Using Conditional Random Fields in Detection of New Intentions for Software Service Evolution

Haihua Xie, Jingwei Yang, Carl K. Chang, *Fellow, IEEE*

Abstract—The capability to accurately and efficiently obtain users' new requirements is critical for software evolution, so that timely improvements can be made to systems to adapt to rapidly changing environments. However, current software evolution cycles are often undesirably long because the elicitation of new requirements is mostly based on poor system performance or delayed user feedback and slow-paced manual analysis of requirements engineers. In this paper, we propose a general methodology that employs Conditional Random Fields (CRF) as the mathematical foundation to provide quantitative exploration of users' new intentions that often indicate their new requirements. By using the CRF method, more accurate desire inference, the precondition for new intention detection, can be achieved compared with other statistical methods. Our methodology is targeted to context-aware software environments, and can discoverer new requirements sooner and considerably shorten software evolution cycles. An experiment on a research library system has been conducted to demonstrate how to apply our methodology to drive system evolution. Finally, we discuss possible threats to validity of our methodology and experiment.

Index Terms—Conditional Random Fields, context awareness, desire inference, human intention, new intention detection, requirements, service, situation, software evolution

1 Introduction

THE goal of software system evolution is to adapt to ever-changing user requirements and operating environments [1]. Evolvability is an essential quality requirement for most software systems [2], [3] because of the nature of user demands [4] and unpredictable environmental changes for modern-day real-world systems [5]. In order to prolong the productive lifetime of software systems, it is necessary to explicitly address evolvability during the entire software lifecycle [6]. The inability to evolve will cause software systems to become progressively less satisfactory, and eventually obsolete [7]. In practice, the target of cost-effective evolution puts strong demands on software engineers to change software systems with major modifications or enhancements in a timely manner [8].

How to make prompt and effective changes to software systems is a big challenge in software evolution [9]. Changes typically start with new requirements that specify new user needs or new system environments. Traditionally, these new requirements are elicited based on delayed user feedback or business needs and are handled by manual analysis [10], a process that fails to keep up with the need for fast-paced software evolution. For software systems with enormously large user bases, new individual requirements constantly emerge and accumulate. To remain competitive in the business, new software development techniques such as "Agile Methods" have been proposed to incrementally develop a working product and deliver it iteratively and frequently (weeks rather than months) [11]. For example, for the Android app "k9mail," its developers release a new version of the application every two weeks; such practices are becoming more and more common in industry [12].

To achieve fast evolution, new approaches to eliciting user requirements are much needed. In recent years, most studies on elicitation of new requirements focus on observing historical defects [13], [14] or analyzing delayed user feedback [15], [16], while little work has been done on exploring human intentions that often drive system evolution [17]. Because service environments are becoming more and more context aware [18], [19], it is possible to observe user behaviors and environmental contexts in real time and analyze or predict users' mental states for acquiring their demands more quickly [20]. To speed up the analysis process, using manual approaches alone is no longer in favor. In fact, as technology advances, applications of mathematical methods in elicitation of new requirements are becoming feasible and even necessary, so that semiautomatic evolution processes can become plausible.

A frontier work of human-intention-driven service evolution is Situ [20], a framework that aims to support rapid and iterative service requirements analysis of real-world systems. Situ builds a Hidden Markov Model (HMM) to deduce desires of an individual user from given observations (a user's actions and environmental context values), and further analyze a user's potentially new intentions for driving service evolution. However, Situ encounters difficulties in desire inference because HMM is not able to encode the causal relations among actions, context values and desires, nor conquer the complexity of desire transitions, due to: 1) in such an HMM, the current

• *The authors are with the Department of Computer Science, Iowa State University, 226 Atanasoff Hall, Ames, IA 50011. E-mail: {haihxie, jwyang, chang}@iastate.edu.*

desire is supposed to be independent of neighboring observations [21]. As a result, the relations of a desire and its previous and following observations cannot be reflected. However, in reality, previous context values may influence a user's current desire, and following actions and context values may be determined by the current desire; 2) in such an HMM, the probabilities of desire transitions are stationary [21]. However, in reality, a user's desire change usually depends on current context values, which may be quite dynamic. Because of the disadvantages mentioned above, Hidden Markov Models are not effective to accurately infer users' desires, not to mention perform detection of new intentions. Therefore, in terms of computability, Situ framework still leaves a lot to be desired.

In this paper, we present a general methodology that applies Conditional Random Fields (CRF) as the mathematical foundation to provide quantitative exploration of users' new requirements [22]. CRF is a class of statistical modelling methods often used for encoding known relationships between observations and constructing consistent interpretations [23]. Here CRF is used to build the mathematical model for desire inference, which is to infer users' desires based on runtime observations of their actions and environmental context values.[1] Since intuitively, desire inference can be regarded as labeling an observation sequence with desires, CRF is chosen in our methodology because it is proven to be more accurate for labeling or parsing of sequential data than traditional statistical methods such as HMM and MEMM (Maximum Entropy Markov Model) [24]. Simply speaking, CRF is more suitable for our study because the relations among actions, context values and desires can be better reflected in a CRF model.

This paper makes the following three contributions:

1. We propose a CRF-based method to effectively infer users' desires based on observations of their actions and relevant environmental context values.
2. We propose three feasible methods to explore users' new intentions based on the results of desire inference.
3. We put forward a new software evolution cycle based on the intention detection methodology that we recommend. To demonstrate and validate our methodology, we conduct a two-round exploratory experiment on a user base of 120 participants.

This paper is organized as follows: Section 2 briefly reviews the literatures on system evolution, requirements elicitation, and human desire & intention, and introduces Conditional Random Fields (CRF) and its applications. Section 3 presents the basic concepts and the methodology of building the CRF model and the process of desire inference and new intention detection using the CRF model. Section 4 explains the first round experiment by demonstrating through each step in our methodology in details, and presents and analyzes the results. We also briefly present the second round experiment results, and mainly focus on evaluating the effectiveness of our methodology on enabling software evolution. Section 5 discusses some potential threats to validity. Section 6 concludes the paper with some speculations.

[1]It is actually a common practice in the software and Internet-based e-commence, originally enabled by the technologies such as Doubleclick and exemplified by companies such as Google, Amazon, and Alibaba. However, systematic academic studies with a sound theoretical foundation are still rare.

2 Background And Related Work

In this section, we first discuss software evolution and requirements elicitation techniques. Then we review the Situ framework, and introduce Conditional Random Fields and its applications.

2.1 Software Evolution & Requirements Elicitation

Generally, software evolution refers to the study and management of the process of making changes to software over time [25]. Sometimes, software evolution is defined based on software maintenance, per IEEE's definition [26]: *the process of modifying the software system or component after delivery to correct faults, improve performance or other attributes, or adapt to a changed environment*. The target of software evolution or maintenance is to implement possible major changes to the system to ensure the reliability and flexibility of the system [27]. As a large software system continues to evolve, the complexity of the system will grow [28], meaning that more effective and efficient evolution methods are much needed.

Proposals for change are the drivers for system evolution, and change identification usually continues throughout the system lifetime. The driving forces of the four maintenance activities summarized by Lientz and Swanson [29] are:

- Adaptive maintenance: adapt to changes in the system environment.
- Perfective maintenance: adapt to new user requirements.
- Corrective maintenance: patch system drawbacks.
- Preventive maintenance: prevent problems in the future.

Among the above four problems, the incorporation of new user requirements is the core problem for software evolution and maintenance [9]. Therefore, the capability to accurately and efficiently obtain users' new requirements is a critical issue to be addressed.

The traditional requirements elicitation process can be considered as a mutual learning process between the requirements engineer and the customer [30]. The knowledge of users' requirements can be obtained from interview [31], feedback [10] or observation of customer activities at their workplace [32]. As user requirements are usually implicit and unpredictable [33], this process mainly depends on a requirements engineer performing subjective analysis and using judgment, so it is usually time-consuming and results in inaccurate requirements. Oftentimes, a cycle of elicitation, modification, development, and deployment takes a long time to complete, usually several months [20]. Researchers are now facing the steep challenge of shortening such undesirably long evolution cycles. New technologies that can enable auto-

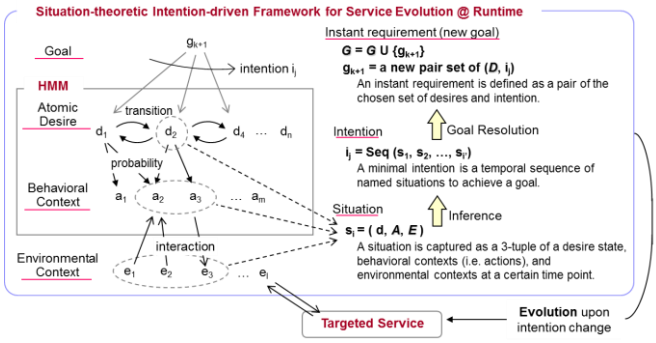

Fig. 1. Situ: Framework for service definition and intention inference with runtime software evolution [20].

matic or semi-automatic requirements elicitation and analysis are needed to realize rapid software evolution.

2.2 The Situ Framework

Situ [20] is the first general approach proposed for human-intention-driven service evolution in context-aware service environments. Situ is also the first computational framework that allows people to model and detect human intentions by inferring human desires, as they are often largely hidden, and capturing the corresponding context values through observations. Equipped with a prediction mechanism, Hidden Markov Model (HMM), in the process of intention detection, Situ envisions a solution path to make instant definition of individualized services at runtime possible, and significantly shorten service evolution cycle.

In order to model and reason human intentions, Situ defines situation as a time-stamped status that includes a user's desire[2], as well as a user's actions and relevant context values. The actions performed can be regarded as the external reflection of a human internal mental state, or so-called "desire." Some context value changes are side effects of human actions that are externally observable. Therefore, through observing a user's actions and context values, it is possible to infer a user's desire at each observation time-point using an applicable mathematical mechanism [34]. Furthermore, a user's intention for achieving a certain desire can be obtained by connecting the situations with the same desire as a sequence. Thus, intention is a path in a scenario, similar to that of studies in robotics [35]. Formally, situation at a time t is defined as a 3-tuple $\{d, A, E\}$, in which d is the inferred or predicted user's desire, A is a set of actions for achieving a goal which d corresponds to, and E is a set of context values. An intention is expressed as $I = seq(S_1, S_2, ..., S_k)$, in which I is an intention and $S_1, S_2, ..., S_k$ is a temporal sequence of situations connected through a unique desire d.

As shown in Fig. 1, instant requirements can be reasoned through goal resolution of the captured intentions. Situ argues that intention change often results in a new goal for the user, requiring the modification of existing features or new pathways to engage new development, such as creating a new functionality. In this way, the system can be enhanced to satisfy the user's new requirements. Since the software evolution process in Situ is semiautomatic, Situ is considered capable of shortening the software evolution cycle, so that the critical and timely service individualization[3] can be made.

Although Situ suggested a new path for service evolution, many practical problems need to be addressed. For example, how to build an effective model for desire inference and detection of intention changes, how to resolve users' new goals and new requirements, and how to make corresponding changes to the system to adapt to the newly elicited requirements, etc. Furthermore, the Situ framework still needs large-scale experimental validation to show its capability and practicality.

2.3 Conditional Random Fields and its Applications

CRFs (Conditional random fields) are a class of statistical modeling methods used to encode known relationships between observations and construct consistent interpretations [23]. They are often used for labeling or parsing of sequential data, such as natural language text [36], [37] or biological sequences [38], [39], which are quite similar to observed sequential data in our work. There are several types of CRF models, and the one adopted in this paper is linear-chain CRF [40].

The general application of the CRF method is sequential labeling, which is to give labels for a sequence of input data. For example, CRF can be used for Part-of-speech (POS) tagging [41], in which the goal is to label each word in a sentence with a POS tag such as *ADJECTIVE*, *ADVERB*, *NOUN*, etc. Before doing this, a set of feature functions are defined to encode the relations between word (data) and POS tag (label). The general format of a feature function is: $f(S, i, l_i, l_{i-1}) = 1$ or 0, in which:

- S is a sequence of input data.
- i is the ordinal number of current data in S.
- l_i is the label for the ith observation in S.
- l_{i-1} is the label for the (i-1)th observation in S.
- The output is 1 when certain relations specified by the function are satisfied among S, l_i and l_{i-1}, otherwise the output is 0.
- Each feature function is associated with a weight that indicates its labeling reliability.

In order to give the best labeling for an input sequence S, the CRF model calculates the score of every possible sequence-labeling L by adding up the weighted feature functions over all data in the sentence:

$$value(L \mid S) = \sum_{j=1}^{u} \sum_{i=1}^{v} w_j f_j(S, i, l_i, l_{i-1}) \qquad (1)$$

(w_j is the weight associated with feature function f_j. u is the number of feature functions in the *CRF* model and v is the length of S.)

Then, these values are transformed into probabilities $p(L|S) \in [0,1]$ by exponentiation and normalization:

$$p(L|S) = \frac{exp[value(L|S)]}{\sum_{L'} exp[value(L'|S)]} = \frac{exp[\sum_{j=1}^{u} \sum_{i=1}^{v} w_j f_j(S,i,l_i,l_{i-1})]}{\sum_{L'} exp[\sum_{j=1}^{u} \sum_{i=1}^{v} w_j f_j(S,i,l'_i,l'_{i-1})]} \qquad (2)$$

[2]Desires denote what users want from the human perspective, while goals encode what users want from the system perspective. They can be mapped, although most likely not one-to-one.

[3] For example, individualization is key to effective and prompt care for elderly people developing into dementia.

The sequence-labeling L with the largest $p(L|S)$ will be chosen as the labeling for the sentence S. To reduce the computation complexity, the Viterbi Algorithm [42] is applied in computing $p(L|S)$, and the Limited-memory BFGS [43] is a common algorithm used for estimating the weights of feature functions in CRF model training.

The CRF method and its extensions or variants are widely used in pattern recognition, machine learning and other domains which deal with structured data [24]. CRF has also been applied to intrusion detection [44] and intent understanding from search behaviors of using search engines [45]. As such, CRF turns out to be a very useful technique to support our research objectives.

3 Detection Of New Intention Using CRF

In this section, we present a methodology of new intention detection for software evolution. First of all, we explain the application fields of our methodology, which are human-centric context-aware domains. Our methodology is targeted to different types of software systems, e.g., dynamic websites, location-based mobile applications, smart homes, etc., as long as they are human-centric and context-aware. We use the term "human-centric" to generalize the common nature of various application systems in which humans play a central role in driving system evolution, and the term context-aware to emphasize the physical properties of the system that is sensor-laden for monitoring user actions and system statuses. Differentiated from existing work [46], [47] that is system/application-specific, our methodology targets at the human-centric characteristics of a class of systems, making itself generally applicable, flexible, and adaptive in a wide range of applications. By characterizing and formalizing the nature of the target application domains, we aim at investigating the following problems:

1. To embody the human-centric characteristics, what essential entities and relations need to be formalized? How can we formalize them?
2. To infer human desires, what kinds of states, i.e., data shall be captured? What should be the logical process of inference?
3. How do we choose a suitable mathematical method to infer human desires based on observations?
4. Based on observations and inferred desires, how can we filter and formalize possible new intentions? And how can we further elicit new requirements to enable software evolution?

3.1 Human-Centric Context-Aware Domain

The methodology for new intention detection for system evolution is generally applicable to the human-centric context-aware domains. To formally describe and characterize such domains, a first-order logic based language, SiSL [48] (situation-centric specification language), is introduced to specify the entities and essential relations.

First of all, three assumptions are proposed to stipulate the application scope:

Assumption 1. *The domain of discourse of SiSL is a single-agent domain*[4]*. The agent is believed to be rational, and has desires and corresponding intentions to achieve certain goals. If there are multiple users in the system environment, when constructing the domain for a specific user A, only A's own desires and actions will be included, while other users' desires and actions will not be. In our discussion regarding the inference process, by default, the subject of all actions and desires is the unique agent in the domain.*

Assumption 2. *The unique active agent is in only one application of the system in a given time period. The agent has only one desire at a time instant. Meanwhile, the process of desire transition is memoryless (Markov property), i.e., the conditional probability distribution of future desires depends only upon the present desire, not on the sequence of desires that preceded it.*

Assumption 3. *The domain of discourse [49] of SiSL is a sensor-laden computer application domain. There shall be some sensors deployed by the system to capture the status of the agent, system, and the environment. The set of context values and actions defined in the SiSL-specified domain knowledge base are limited by the sensors' observation capability. The inference of predefined desires and detection of new intentions will be performed based on the observations of actions and context values.*

According to Assumption 1, each SiSL domain has one unique agent in the system, in which the desire and actions at a time instant shall belong to the same agent, so that the relationships between the desire and actions can be specified. This assumption rules out the complexity of cases in which one user's desire is directly influenced by other users' actions. E.g. user A changes his/her desire by observing other users' actions. However, other users' influences can be indirectly reflected in relevant context values. Here is an example: Two users, A and B, are living in a smart home environment. In the domain of user A's smart home system, it is assumed that user B's actions have no direct influence on user A's desire; however, they can trigger some context value changes, which may have influence on user A's desire and actions. Such indirect influence can be described by SiSL.

According to Assumption 2, the agent has only one desire at an instant, i.e., the agent does not have parallel or concurrent desires. Note that in our discussion, the desires should be consistent with system goals, i.e., they should be relevant to the system domain. Assumption 2 is also compliant with the definition of situation and intention (which will be introduced later in this section): there is only one desire in a situation, and intention can be inferred by observing a sequence of situations that share the same desire. Additionally, we assume that the transition of desires should satisfy the Markov property, in order to reduce the complexity of the domain while still keeping it rich enough for desire inference. Later in section 4.6, our experiment evaluation results show that this assumption is in fact appropriate.

Assumption 3 can be considered a prerequisite of the

[4] For example, a customer who signs in to a browser and purchases merchandise items featured on Amazon.com is a single agent. On the other hand, multi-player gaming involves multiple agents.

TABLE 1
LEXICON FOR BASIC THEORIES IN THE SISL ONTOLOGY

Entities	*Action(a)*	*a is an action*
	Desire(d)	*d is a desire*
	Object(o)	*o is an object*
	Situation(s)	*s is a situation*
	SitSeq(q)	*q is a situation-sequence*
Situation	*ActionIn(a, s)*	*a is the agent's action in situation s*
	DesireIn(d, s)	*d is the agent's desire in situation s*
	Before(s_1, s_2)	*situation s_1 happens before situation s_2*
	After(s_1, s_2)	*situation s_1 happens after situation s_2*
Situation-sequence	*SituationIn(s, q)*	*s is a situation in situation-sequence q*
	Initial(s, q)	*s is the initial situation of situation-sequence q*
	Final(s, q)	*s is the final situation of situation-sequence q*
	Prev(s_1, s_2, q)	*s_1 is the previous situation of situation s_2 in situation-sequence q*
	Next(s_1, s_2, q)	*s_1 is the next situation of situation s_2 in situation-sequence q*

application of our methodology. Since actions and context values are used as visible states for inferring invisible states (desires), they have to be observable. In our discussion, the term "sensor" represents all general monitoring mechanisms. For example, it can be a hardware sensor device, or a monitoring module hard-coded in the system. It is usually believed that more effective monitoring mechanisms can usually help capture more useful data, and the richer the data, the better our methodology works.

After delimiting the scope of the domains in which our methodology will be applied, we define the basic concepts in SiSL as follows:

Definition 1. *There are five sorts of entities for reasoning about users' behaviors and internal mental states in software system domains – action, desire, situation, situation-sequence and object. Any entity in the domain must belong to exactly one of the five sorts.*

The concepts of the five sorts of entities are described as follows with key predicates shown in TABLE 1:

1. Situation: catch-all status of the domain at a time instant. More specifically, a situation describes the time-stamped status of the agent (behaviors and desire) and the system environment.
2. Action: agent's behavior performed in the system's environment, particularly their operations on the system interface. For example, in a manuscript management system to support a conference, an agent's operations *click button "submit"* and *click link "view paper information"* are considered as actions, and they can be represented in the form of functions: *click(btn_Submit)* and *click(link_ViewPaperInfo)* respectively. Here *click* represents the action itself and *btn_Submit* and *link_ViewPaperInfo* are objects involved in these actions. To represent an action that occurs in a situation, a special predicate *ActionIn* is defined. For example, if *ActionIn(a_i, s_j)* is true, it means action a_i occurs in situation s_j.
3. Desire: the condition or status of entities that the agent would like to achieve. For example, *upload a paper* or *download a review* can be considered as a desire. To represent an agent's desire in a situation, a special predicate *DesireIn* is defined. For example, if *DesireIn(d_i, s_j)* is true, it means agent's desire is d_i in situation s_j. By recalling Assumption 1, there must be only one desire d for which *DesireIn(d, s_j)* is true. Since the desire is hidden in the agent's mind, it needs to be inferred.
4. Situation-sequence: A sequence of situations. Due to the limitations of first-order logic [49], the definition of a sequence has to rely on the definitions of predicates such as *Initial()*, *Final()*, *Next()*, *Previous()*, etc. The entity of situation-sequence is necessary because it is essentially a different concept from the entity of situation, and it is the key to defining intention.
5. Object: any entity other than desire, action, situation, or situation-sequence in the domain, such as button "*Submit*", number "*1*", etc.

Based on the definition of the five basic entities, another two important concepts, context and intention, can be defined as follows:

6. Context: any information that is used to characterize the status of objects in the domain of discourse. Context can be regarded as relations between objects and situations. A context is defined as a function or a predicate (a special function), with values in its range called context values. For example, a functional context *numOfSubmittedPapers(s_n)* represents the information "*the number of submitted papers in situation s_n*", and its context value can be *0*, *1*, *2*, …, any non-negative integer; a predicate context *LoggedOn(x, s_1)* represents the information "*x has logged on in situation s_1*", and its context value can be "*true*" or "*false*" only.
7. Intention: an execution path for achieving a certain desire. In SiSL, an intention is represented as a situation-sequence in which all situations are chronologically linked while containing the same desire.

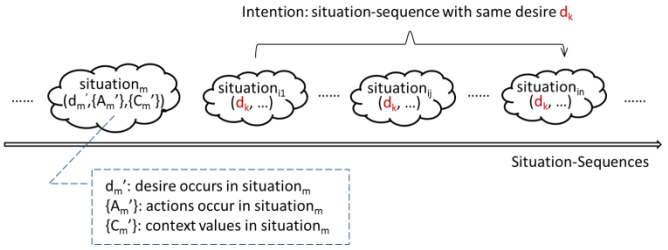

Fig. 2. Relations of different entities in the domain.

Fig. 2 illustrates the relations of different types of entities in the human-centric context-aware domain. In essence, our methodology's highlight is to explore the relation between environmental visible states and human internal states. To achieve this goal, a set of fundamental axioms are introduced to specify the relations between

action and desire, context value and action, desire and context value, respectively.

Axiom I. *Action determinism: The set of possible actions in any situation is determined by the desire and context values in the current situation.*

Axiom II. *Context value determinism: The values of some contexts in future situations are determined by the actions and context values in the current situation, while other context values are not influenced by any action, depending on the nature of those actions and contexts.*

Axiom III. *Desire satisfaction determinism: A desire is satisfied when a set of context values are reached in the current situation.*

According to Axiom I, actions performed can be regarded as an external reflection of a human internal mental state (i.e., desire) in a specific circumstance. For example, an agent wants to log into his email inbox. When the network speed is good, he usually chooses the normal view; when the webpage stays on buffering for a long time, he may switch to a simplified view to speed up the process, or may keep on waiting. In both cases, the agent wants to satisfy the same desire, however, he has two different possible actions, and his decision to choose which action to perform should conform to certain probability distribution depending on the context value (the network speed).

Axiom II provides an explanation of relations between actions and context values. It states that some context value changes are the results of agent's actions. These context value changes are directly or indirectly influenced by the actions, while other context values may not be influenced by any action, such as the time of the day. To build a complete domain knowledge base, the context values for all contexts, including those can be influenced by actions and those cannot, should be specified. While in practice, such context value determinisms can be encoded into a mathematical model.

Axiom III indicates that desire changes may depend on the context values in the current situation, because if the ideal context values are reached, i.e., the agent's desire is satisfied, a new desire is more likely to emerge, otherwise the desire is less likely to change.

Given the above axioms, we can establish certain determinative connections from desire to action, to context value, and then to desire changes. Furthermore, to completely describe the domain, the determinative relations from each desire to each action, from each action to each context value, and from each context value to each desire, all should be specified. However, in practice, agent's action selections and desire changes sometimes may be random and may not strictly conform to our axioms, because humans are such complex beings, and sometimes unpredictable. From a statistical point of view[5], based on long-time observations and with the help of suitable mathematical methods, we can still build a computational model to reflect the patterns of agent's action selections and desire changes in those commonly seen situation sequences, which should approximately satisfy Axiom I, II and III, as we believe. Hence, the Axioms proposed above can be used as the criteria for selecting our observation means and mathematical methods.

[5] More discussions can be found in section 5.2.

3.2 Desire Inference and New Intention Detection

As the precondition for new intention detection, desire inference is to infer an agent's desire at each observation time point. According to the axioms introduced in last section, it should be possible to infer an agent's desires based on his/her actions and current context values. Intuitively, HMM is a feasible method because it is used to predict hidden states (desire) from observable variables (actions and context values), and it is also considered capable of determining human mental states from human actions [21]. In fact, we initially started with HMM and eventually chose a more judicial method, CRF, which we now consider a better fit for our methodology. More detailed comparison between these two methods will be introduced in section 3.3. But, before delving into the depth of selecting mathematical models, it is necessary to design the observation data first.

To accurately infer agent's desires, we propose to observe the following states within the observation capability:

1. Agent's actions within the system domain, especially operations on the system interface.
2. Those contexts whose values may influence agent's actions.
3. Those contexts whose values can be changed by agent's actions.
4. Those contexts whose values can indicate whether agent's desire is satisfied or not.
5. Other context values which may not be considered as of any significance, but can be easily captured. The motivation to include these data is to construct raw data for future analyses, since the data may become valuable and relevant at some point.

After determining the content to observe, we further suggest the instant at which an observation should be made:

1. Action trigger: each time when the agent performs an action that is supposed to be observed.
2. Context-value-change trigger: each time when there is a change of the context value being monitored.

The above principles to determine observation contents and observation frequency are given as general guidelines, as each individual system may have its own characteristics and limitations, and shall be carefully handled on a case-by-case basis. However, even for the same system/domain, different observation setups may capture different sets of raw data, which may lead to different inference results. In order to get the most out of our methodology, it is recommended to experiment with different data collection mechanisms and methods, and select the one with the best performance in regard to meeting our needs.

After setting up the monitoring mechanism, now we introduce the principles of desire inference and new in-

tention detection, as well as related concepts as follows (A step-by-step methodology is given in section 3.3):

1. Observation: An observation is the observed status of the system domain at a time point, including a set of actions and context values that are time-stamped. Desire inference and new intention detection will be performed based on observations. Based on the observation contents and frequency discussed above, a number of applicable and necessary sensors should be deployed in the system to capture an agent's actions and relevant context values. Some actions and context values cannot or will not be captured due to observation capability or non-necessity, hence an observation may contain only a part of the status of the domain.
2. Desire Inference: In desire inference, the inference model takes a sequence of observations as the input and outputs the agent's desire at each observation time point. As we mentioned at the beginning of this section, a statistical inference model is used in this process. Feasible candidates can be HMM, CRF, etc.
3. Intention Inference: After the agent's desire is inferred for each observation, the intention for achieving a certain desire can be obtained by connecting the situations with the same desire into a sequence that is destined to reach a goal state. The underlying rationale is because of our definition of intention in section 3.1.
4. New intention detection: A new intention is a situation-sequence pattern which has not been predefined in the domain knowledge base, i.e., an agent's new behavior sequence pattern for achieving a desire. We believe that in theory a new intention will arise in two cases: either the agent has a new desire, or the agent has a new strategy for achieving a predefined desire. In practice, new intentions will also be detected when the agent cannot properly perform operations due to system flaws, which may cause their divergent behaviors. All of these cases are useful for system evolution, because they can imply an agent's new requirements or system drawbacks due to deviation from the known set of user's intentions. Therefore, the ultimate goal of our methodology can be reached, that is to detect an agent's new intentions for driving system evolution.

3.3 New Intention Detection Using the CRF Method

The Conditional Random Fields (CRF) method is applied in our research to build the computational model for desire inference and new intention detection. More specifically, we use linear-chain CRF (refer to section 2.3). Compared with HMM, linear-chain CRF has more advantages on labeling sequential data and is more suitable for our research due to the following reasons:

1. According to Axiom I, the agent's actions are usually determined by their current desire and current context values. Such determinative relations can be reflected in a CRF model through defining feature functions that encode the known relations between the attributes of the current observation of context parameters and the current desire. However, in HMM, Axiom I cannot be well reflected because the current action is only determined by the current desire according to the state-observation emission relations [21].
2. According to Axiom II, a context value in a later observation may depend on context values and an agent's actions in a former observation. In CRF, such relations can be encoded in some feature functions to better infer the current desire. However, in HMM, the relations between consecutive observations cannot be reflected because the observations are supposed to be independent (Output-Independent Assumption [21]).
3. According to Axiom III, the satisfaction of a desire depends on current context values. As an agent usually moves on to fulfill a new desire when their current desire is duly satisfied, desire transitions also depend on current context values. Such desire satisfaction determinism and desire transition principle can also be reflected in a CRF model using feature functions that encode the known relations among the attributes of the previous observation, the previous desire and the current desire. In HMM, according to the Stationarity Assumption [21], the desire transition probabilities are stationary regardless of the current context values, so the desire transition principle cannot be well represented.

TABLE 2
COMPARISON BETWEEN HMM AND CRF IN DESIRE INFERENCE

	HMM	**Linear-CRF**
Axiom I	Not supported because of state-observation emission relations	Supported by defining feature functions to reflect Action Determinism
Axiom II	Not supported because of Output-Independent Assumption	Supported by defining feature functions to reflect relations between consecutive observations
Axiom III	Not supported because of Stationarity Assumption	Supported by defining feature functions to reflect Desire Transition based on context values
Assumption 2	Supported because of Markov Property	Supported because of linear property

Based on the above reasons (also shown in TABLE 2), linear-CRF is chosen as the mathematical foundation for our methodology of desire inference and new intention detection. The general format of feature functions in the linear-chain CRF model in this paper is formulated as $f_n(O, i, d_i, d_{i-1})$, in which:

- O is a sequence of observations.
- i is the ordinal number of current observation in O.
- d_i is the inferred desire for the ith observation in O.
- The output of f_n is 1 when certain relations speci-

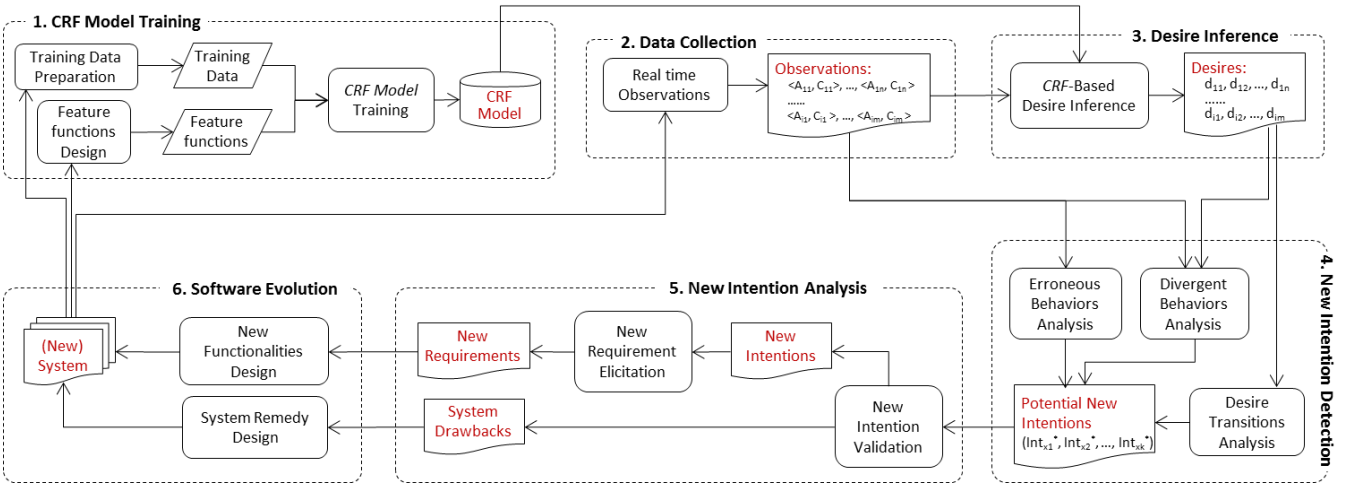

Fig. 3. Methodology for Desire Inference and New Intention Detection.

fied by the function are satisfied among O, d_i and d_{i-1}, otherwise the output is 0.

The fundamental task of desire inference using CRF is to find the desire sequence D^* with the largest labeling score (highest probability), and use D^* as the inference result for the input observation sequence. Technically this task is very similar to the one that formula (2) (in section 2.3) works for, which is to predict word labels for a sentence. We can modify formula (2) by replacing its parameters with our own, and have formula (3) as the core mathematical model for desire inference for observation sequence $O = <o_1, \ldots, o_n>$:

$$D^* =< d_1^*, \ldots, d_n^* >= \underset{D}{\operatorname{argmax}} \left(\frac{\exp\left[\sum_{j=1}^{m} \sum_{i=1}^{n} \lambda_j f_j(o, i, d_i, d_{i-1})\right]}{\sum_{D'} \exp\left[\sum_{j=1}^{m} \sum_{i=1}^{n} \lambda_j f_j(o, i, d_i', d_{i-1}')\right]} \right) \quad (3)$$

- λ_n is the weight associated with f_n.

The complete process of applying our methodology to detect new intentions for software evolution is depicted in Fig. 3. The detailed description of each step's task is:

Step 1: Constructing the CRF model. To achieve the two goals in our study: 1) to accurately infer the agent's desires predefined in the domain knowledge base; 2) to detect the agent's new or unexpected intentions; a linear-chain CRF model that encodes users' known behavior patterns for achieving desires should be built as the metrics for outlier detection [50]. We propose to use supervised learning [51] to train the CRF model.

Step 1.1: Training Data Collection: The input training data consists of a set of sequences of observation, and each of the sequence shall follow the form of $<o_1^*, d_1^*>$, $<o_2^*, d_2^*>$, …, $<o_m^*, d_m^*>$, in which each observation o_i^* is labeled with a desire d_i^*. Since the CRF model is used as a standard reference model to infer desires, it must be built upon the existing domain knowledge, including known or pre-defined desires, behavior patterns, etc. To make it happen, the observation sequence $<o_1^*, o_2^*, \ldots, o_m^*>$ in our training data shall be collected from observing user behaviors that are expected to conform to the system design, e.g., existing use case scenarios and sequence diagrams, etc., and the desires associated with each observation $<d_1^*, d_2^*, \ldots, d_m^*>$ shall be accurately reported by users[6] or manually labeled by domain experts.

Step 1.2: Defining Feature Functions: Based on domain experts' knowledge, feature functions can be defined to reflect the relations between observations and desires. If using existing tools, such as CRF++ [52], Flex-CRF [53], etc., we need to design feature templates which specify the form of feature functions (to be introduced in section 4.4 with examples.)

Step 1.3: Training: With the training data prepared in Step 1.1, and the feature functions designed in Step 1.2, we can start training the CRF model. Existing tools are available for building the CRF model, each with its own technical merits. Alternatively, researchers and developers can design and write their own algorithm based on their own preference.

The main task in this step is to construct the CRF model and prepare it for upcoming inference work. To get the best inference result, it is recommended to iterate the steps from 1.1 to 1.3, and to use different training data sets with different feature functions configured for training, so that an approximately-optimal model can be acquired.

Step 2: Data Collection and Pre-processing: Observe an agent (real user)'s actions and relevant context values when they operate on the system, and generate observation sequences $<o_{11}, \ldots, o_{1n}>$, $<o_{21}, \ldots, o_{2m}>$, …;

Step 3: Desire Inference: Input the observation sequences prepared in Step 2 into the CRF model acquired in Step 1, and infer the agent's desire at each observation time point.

Step 4: New Intention Detection: As introduced in section 3.2, a new intention is a situation-sequence pattern which has not been predefined in the domain knowledge base. Considering the various causes of new intentions, and for the purpose of system improvement, we propose three methods for detecting new intentions based on the results of desire inference:

[6] In our IRB-approved experiment, this is done in a "think-aloud" fashion.

- **Method I: Analysis of divergent behavior**

Detect users' divergent behaviors through desires inferred with low confidence (probability). Divergent behaviors usually occur when a user does not follow an expected way to operate the system (a predefined intention), which may indicate their dissatisfaction on the system or expression of new desires. For example, in a ticketing system, a user might think the process is too tedious so he tries different ways to skip some steps or starts over to find if there is any faster entry. His behaviors may appear irregular and cannot be well interpreted by the CRF model, which will result in low confidence when labeling desires. Therefore, these divergent behaviors can be singled out for analyzing users' new requirements.

- **Method II: Analysis of desire transition**

Obtain a new intention based on desire transitions. If two desires often appear consecutively, a new intention can be considered to make the desire transition smoother. Accordingly, the system can be modified to simplify users' operations. For example, again in the ticketing system, if users often move on to print the ticket right after finishing buying a ticket, a link that directs the users to the printing page can be added on the confirmation page of "*buying a ticket*". This kind of desire transition can be obtained based on the results of desire inference with a high confidence level.

- **Method III: Analysis of erroneous behavior**

Find new desires and system drawbacks from users' erroneous behaviors. When users have new desires which are not supported by the current system, their behaviors may trigger error reports. For example, in the same ticketing system, if a user tries to input a phone number in a format that is not supported by the system, an error report will be generated. If such error report occurs time and time again, it may indicate that many users actually want to input their phone number in a non-supported format, which can become a new intention. Additionally, some of the detected users' erroneous behaviors can be false positive, which means these behaviors are actually normal, while system drawbacks/defects make them look "erroneous". So these behaviors coupled with related error reports can be a good starting point to detect system drawbacks and defects, and to further improve the system.

3.4 System Evolution Process

The detected potentially new intentions and system drawbacks will be further analyzed by domain experts to determine whether they are indeed new intentions or system drawbacks. This step cannot be done automatically, and must be manually performed by domain experts. The system drawbacks may not be too difficult to identify, while the new intentions are usually implicit and hard to decide. Domain experts can make some assumptions of the new intentions and then verify them based on their expertise, or they can directly ask the users for evaluation if possible. After this validation and verification process is done, the current system shall be redesigned to satisfy users' new intentions, or be retrofitted with necessary remedies to overcome those revealed drawbacks/defects.

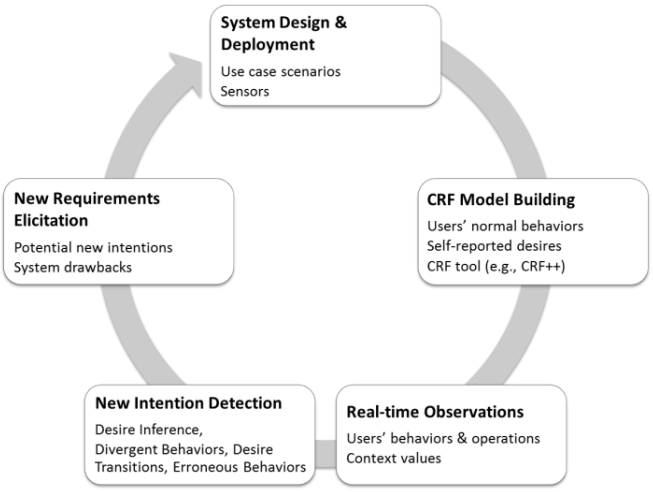

Fig. 4. Process of system evolution driven by new intention detection using the CRF method.

The complete process of system evolution based on new intention detection using the CRF method is shown in Fig. 4.

In summary, our methodology is supposed to be more efficient and effective for system evolution than traditional approaches for the following reasons:

1. Our methodology is based on observations of users' real-time context-aware behaviors, while traditional approaches often depend on delayed user feedback, system defect reports or system performance logs, etc. [16], [14], [54]. Therefore, our methodology can shorten the time to obtain useful information in order to analyze new requirements.
2. Observations of actions and relevant context values contain richer and more meaningful information for new requirements elicitation because they are the direct reflection of users' internal mental states that occur during their operation on the system. On the contrary, user feedback is usually not as accurate and often implicit, while other resources such as system defect reports and system performance logs may not reflect users' new requirements on the system effectively.
3. The application of the mathematical method linear-chain CRF enables accurate and rapid inference of users' desires, making our methodology more efficient than traditional manual analysis. The three methods proposed in section 3.3 for new intention detection can be automated as well. Hence, they can obtain useful information quickly and speed up the process of new requirements elicitation.
4. The new requirements discovered in our methodology are directly elicited from user's unexpected behaviors in the operational environment, thus the system can be pertinently improved to adapt to those user behaviors. Therefore, the system evolution path is supposed to be more appropriate and effective.

4 Experiment On A Research Library System

As stated in the previous section, we believe that our methodology is beneficial for efficiently and effectively discovering new requirements and shortening the software evolution cycle. In this section, we present an exploratory experiment on a dynamic web-based system to demonstrate and validate our methodology. We call it "exploratory" because there is no similar work for us to compare with our methodology.

First of all, we made the following hypotheses before conducting the experiment:

1. Based on observations of user's actions and relevant context values, it is possible to accurately infer user's desires using a CRF model. We expect the inference accuracy will be 90% or higher and it should be higher than that by using HMM.
2. It is possible to detect some new intentions based on the results of desire inference using the three methods introduced in section 3.3.
3. New user requirements or system drawbacks can be revealed based on the detected new intentions. Rapid system evolution can be achieved based on the revealed new user requirements or system drawbacks.

To validate the above hypotheses, our experiment can be divided into two sub-experiments and one case study as follows:

1. First-round experiment: An experiment to validate that high desire inference accuracy can be achieved by using CRF.
2. New-intention-detection case study: A case study to demonstrate our methodology for new intention detection and new requirements elicitation.
3. Second-round experiment: An experiment to validate the system improvement between two versions of the system.

4.1 Experiment Platform – the CoRE System

The experimental system is an online library system, called Cooperative Research Environment (CoRE). It was modified based on an open-source web application, MyReview [55], for managing the process of paper submission and paper review. The original system has served many academic conferences, and our modification still keeps its basic functionalities. Therefore, our experiment is expected to emulate users' operations on a real-world system. Another reason for choosing a web-based system to conduct our experiment is that it is easier to get participants' self-reported desires, which are used to validate our methodology.

CoRE has been designed and developed as a research community for people to share their thoughts and views on academic papers. Users can upload research papers, submit comments for papers, and view paper information, etc. Fig. 5 is the use case diagram of the CoRE system.

To monitor users' behaviors and capture related context values, an embedded program is deployed in CoRE as a sensor. Users' operations, paper and comment submissions, and the contents on the web pages will be recorded and stored in the database. During each experiment session, users report/select their current desires from a dropdown list containing a set of expected desires. Examples of desire options are "*Upload a paper*", "*Submit a comment*", "*View a paper information*", etc., and "*Not in the list*". The users can also correct their desire selections in the post-session questionnaire in case that they forgot to report desires or reported an incorrect one during the experiment. Fig. 6 shows the interface of the page for uploading a paper in the CoRE system, on which there is dropdown menu in the right-side panel for users to report their desires during the experiment.

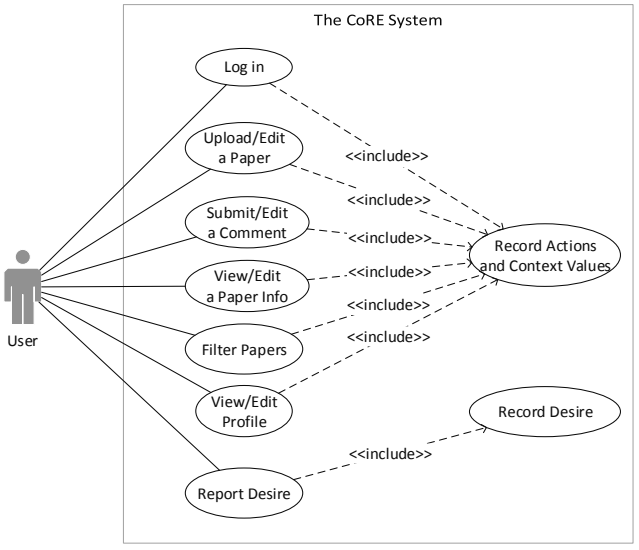

Fig. 5. The use case diagram of the *CoRE* system.

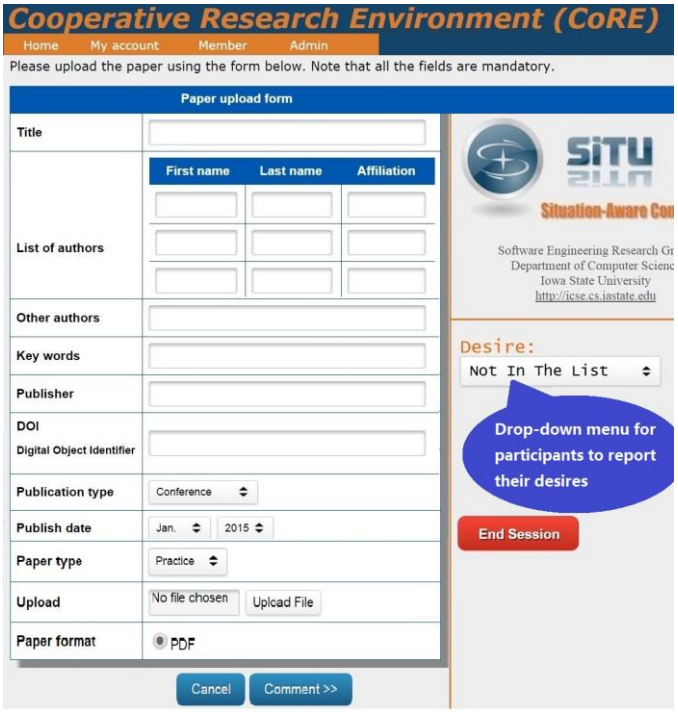

Fig. 6. Interface of the page for uploading a paper in *CoRE* Version I.

4.2 Procedure of an IRB Approved Experiment

Due to the nature of our experiment that involves human subjects, and to stay compliant with federal regulations set forth by the Department of Health and Human Services and the Food and Drug Administration, all of the principle investigators in our experiment have completed the National Institutes of Health (NIH) Web-based training course "Protecting Human Research Participants", and they have also closely worked with our local Institutional Review Board (IRB) for approval for conducting our experiment, which was granted before our experiment was implemented.

More than 120 people participated in our experiment. Each participant was required to study the user manual, and had a chance to do some test operations on the system to get a preliminary understanding about it. Participants' actions, self-reported desires, and relevant context values were recorded as the experiment raw data. In order to show our methodology's ability to enable and speed up the evolution process of the CoRE system, the experiment was done over two rounds (as two sub-experiments) to emulate one software evolution cycle. The whole procedure is described as below:

1. Deploy the initial system: CoRE Version I.
2. Run the first-round experiment for 30 days: invite participants, collect data on participants' actions, desires and context values.
3. Shut down CoRE Version I. Apply our proposed methodology on the raw data captured in Round 1, using CRF to infer participants' desires. Then carry out New-intention-detection case study, analyze and elicit users' new requirements, and further revise the system accordingly.
4. Deploy the enhanced system: CoRE Version II.
5. Run the second-round experiment for 30 days: invite some new participants, collect data records of participants' actions, desires and context values.
6. Shut down CoRE Version II. Apply our proposed methodology on the raw data captured in Round 2, using CRF to infer participants' desires.
7. Evaluate the effectiveness of our methodology on the evolution of CoRE from Version I to II.

During each sub-experiment, participants could enter CoRE multiple times, and each time was recorded as one session. In each session, participants followed the procedure as follows:

1. Visit experiment website and log into experiment.
2. Answer pre-session questionnaire about their familiarity with the system.
3. Start a session: Log into CoRE and start operating on the system. Participants were free to carry out any operation, but needed to report their desire on each webpage by choosing a predefined desire in a dropdown list.
4. End a session: Participants could end a session at any time. Then they were directed to a post-session questionnaire, where they gave some feedback on the system, and also had a chance to correct their reported desires if necessary.

Observation in our experiment was action triggered. During each experiment session, the embedded monitoring program in the system took a snapshot of the participant's action and certain system context information when he/she performed an operation on the system.

TABLE 3
EXAMPLE RAW DATA RECORD

Record Item	Data
Time	2014-06-23 12:11:20
loginID	User020
Action	click(Btn_Login)
Page	Page_Login
Content	[Login ID]Test001 [Password]112233 [Message]Invalid password
Desire	Filter Papers

Each raw data record has the following attributes, with an example shown in TABLE 3:

1. Time: the time point when the participant performs an operation.
2. Participant's login ID.
3. Action: including mouse click on a button or a link, or selection on a dropdown menu.
4. The current webpage where the action occurs.
5. Contents on the webpage (user's submitted input, system's responses to the user's action including exceptions and error messages).
6. Participant's self-reported desire.

4.3 Data Collection and Preprocessing

This step corresponds to Step 2 in our methodology (in section 3.3). In our experiment, Step 2 and Step 1 in our methodology were done in a reverse order, because several domain experts and system designers were also participants, and their own data records were mixed with others and were later filtered out as training data for building the CRF model. For other applications, if there is a good historical data source available that is suitable for training purposes, one can certainly follow all steps of our methodology in the normal order.

There are 10,063 raw data records and 585 experiment sessions captured in the first-round experiment, and 10,524 raw data records and 582 experiment sessions captured in the second-round experiment. A raw data record contains all the attributes shown in TABLE 3. A record of an experiment session is the data sequence that starts when the user logs into the experiment, and ends when the user logs out.

As the accuracy of participants' self-reported desires is critical for validating the results of desire inference, those raw data records with inaccurate self-reported desires are not usable and are considered as noise. To remove noise, raw data records were checked and filtered, data noise in a unit of sessions were removed based on the following principles:

1. If most (>50%) self-reported desires are "Not in the list", which is the default value in the desire selection dropdown list. We will assume that in this case the participant forgot to report his desire at all or most of the time in the experiment.
2. If the answer is "no" for the question "Did you se-

TABLE 4
SAMPLE TRAINING DATA

Observation	Time Interval	Desire
clickMenuAllPapers	30s	ViewAllPapers
clickPaperInfos&PaperID	m	ViewAPaperInfo
clickFilter&FilterCategory	60s	FilterPapers

TABLE 5
PARTITION OF TIME INTERVALS

Actual duration of the interval	Time Interval Coding
0 ~ 5s	5s
5s ~ 10s	10s
10s ~ 20s	20s
20s ~ 30s	30s
30s ~ 60s	60s
60s ~ 60mins	m
>= 1 hour	h

lect the desire every time when you had a new desire?" in the post-session questionnaire.

3. After filtering based on (1) and (2) was done, we had the rest of the data records manually checked by domain experts and system designers, and evaluated in the perspective of whether the participants' self-reported desires were reasonable or not. Those obviously wrong ones should and have been removed because they cannot be used to validate our desire inference results. An example is that a participant might have forgotten to report his desire (change) when he accomplished his previous task and started a new one.

After preprocessing and noise filtering, the final usable data set of the first-round experiment had 6880 data records and 369 experiment sessions, and the final data set of the second-round experiment had 6931 data records and 361 experiment sessions.

4.4 Building the CRF Model

According to Step 1 in section 3.3, a CRF model was built to encode users' known behavior patterns. The training data records for building the CRF model were collected from observing user behaviors that were expected to conform to the system design. In our experiment, the training data was selected based on the following criteria:

1. The records belonged to system designers and those experienced participants whose behaviors were conducted to achieve certain expected desires. By "expected", it means those behaviors shall conform to the use case scenarios designed for CoRE, including normal cases, exceptions, and alternatives.
2. The training data set covered all of the expected desires.
3. The self-reported desires recorded in the training data set were accurate.

After being processed from the raw data, a training data set was acquired for building the CRF model. There were 2930 data records and 158 experiment sessions in the training data set of the first-round experiment. A sample of these training data is given in TABLE 4.

TABLE 6
FEATURE TEMPLATES

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

Each training data record had three elements: observation, time interval and desire.

- Observation: includes the action and context values which are captured simultaneously. The format is *action&contextvalues*, e.g., in the observation *clickFilter&FilterCategory*, *clickFilter* is an action and *FilterCategory* is a context value.
- Time interval: the time interval between two consecutive observations. A time interval is calculated based on the time point in each data record. The partition of different time intervals is given in TABLE 5.
- Desire: user's self-reported desire.

We used an open-source tool called CRF++ [52] to build our CRF model based on the training data. According to Step 1.2 in 3.3, we needed to design the feature functions for our CRF model, and in CRF++, they can be automatically generated based on feature templates that specify their formats. The feature templates used in our experiment are shown in TABLE 6, in which each entry denotes one template. In each template, special macros in the format of *%x[row, col]* were used to specify a token in the input data, where *row* specifies the relative position from the current focusing token and *col* specifies the absolute position of the column. The above feature templates were acquired after experimenting with the data for the highest inference accuracy. Below are some brief explanations of some key feature templates and why they are good for building the CRF model:

1. *U01~U06*: the unigram templates specify the relation between the observations and the current desire. *%x[-1,0]*, *%x[0,0]* and *%x[1,0]* represent the previous, the current and the next observation respectively. An example feature function generated by *U02: %x[-1, 0]/%x[0, 0]* based on data records in TABLE 4 is:

$$f_1(O, n, d_{n-1}, d_n) = \begin{cases} 1 & if\ d_n = \text{ViewAPaperInfo} \\ & O_n = \text{clickPaperInfos\&PaperID} \\ & O_{n-1} = \text{clickMenuAllPapers} \\ 0 & Otherwise \end{cases}$$

The above feature function can be interpreted as: if the current observation O_n is *click the link "View Paper Infos" of a paper* and its previous observation O_{n-1} is *click the menu option "All Papers"*, the current desire d_n is supposed to be *View a paper's information* with a labeling confidence w_1 associated with f_1.

Because of the nature of the domain of CoRE, we can improve the accuracy of desire inference by considering

TABLE 7
EXAMPLE SEGMENT IN THE TRAINING DATA SET

Observation	Time Interval	Desire
clickLogin&LoginGood	30s	EditProfile
clickMenuMyProfile	10s	EditProfile
clickSubmit&ProfileUpdated	30s	EditProfile

TABLE 8
EXAMPLE SEGMENT IN THE TRAINING DATA SET

Observation	Time Interval	Desire
clickLogin&LoginGood	10s	ViewProfile
clickMenuMyProfile	10s	ViewProfile
clickMenuAllPapers	30s	ViewAllPapers

the neighboring observations when designing feature templates. For example, there is a segment in the training data set as shown in TABLE 7, in which observation *clickLogin&LoginGood* is labeled as *EditProfile* because the following observations show that the user updated his profile. Another segment is shown in TABLE 8, in which the observation *clickLogin&LoginGood* is labeled as *ViewProfile* because the following observations show that the user did not update his profile so he probably just viewed the profile. This kind of relationship between observations and desires shown in the above two examples can be encoded into the CRF model based on template *U06:%x[0,0]/%x[1,0]/%x[2,0]*, which takes the current, next and after next observations into account for inferring the current desire.

2. *B01~B04*: the bigram templates, different from the unigram ones, consider the previous desire for inferring the current desire. These four bigram templates specify the relations among observations, time intervals, current desire and previous desire. An example feature function generated by *B02: %x[0, 0]/%x[0, 1]* based on data records in TABLE 4 is:

$$f_2(O,n,d_{n-1},d_n)=\begin{cases}1 & if\ d_n=\text{ViewAPaperInfo}\\ & \quad d_{n-1}=\text{ViewAllPapers}\\ & \quad O_n=\text{clickPaperInfos\&PaperID}\\ & \quad TimeInterval(O_n)=\text{m}\\ 0 & Otherwise\end{cases}$$

The above feature function can be interpreted as: if the current observation O_n is *click the link View Paper Infos of a paper* and the previous desire d_{n-1} is *View all papers*, the current desire is supposed to be *View a paper's information,* which is not consistent with the previous desire because the time interval between two observations is *m* (which is long).

Because of the nature of the domain of CoRE, we can make the inference results more accurate by adding the factor of time intervals into feature templates. TABLE 9 shows two example segments in the training data set (separated by dotted lines).

The first segment in TABLE 9 shows that if the user's desire is to *Upload a paper*, the time he stays on the page *Upload Paper* will be long (more than 1 minute), and if the user just wants to do some *Test*, the time interval is most likely short. This kind of relationship among observations, time intervals and desires can be encoded in the feature functions generated by template *B02*, which takes the current observation and current time interval into account for inferring the current desire.

TABLE 9
EXAMPLE SEGMENT IN THE TRAINING DATA SET

Observation	Time Interval	Desire
clickMenuUploadPaper	10s	UploadPaper
clickSubmit&NoFile	m	UploadPaper
……	…	…
clickMenuUploadPaper	10s	Test
clickSubmit&NoFile	20s	Test

TABLE 10
EXAMPLE SEGMENT IN THE TRAINING DATA SET

Observation	Time Interval	Desire
clickMenuMyPapers	10s	UploadPaper
clickMenuUploadPaper	5s	UploadPaper
clickSubmit&PaperInfoGood	m	UploadPaper

For a similar reason, time gaps between observations are important to desire inference, as careless or unintentional errors are usually corrected shortly after they are made. An example of this case (shown in TABLE 10) shows that the participant's desire was to *Upload a paper*, but he clicked link *My uploaded papers* first by mistake and then clicked link *Upload a paper* in 5 seconds. The accidental erroneous operation will still be correctly labeled if considering the short time interval between the current and the next operation. This kind of relations can be represented in template *B01*, which encodes the relations among the current time interval, previous desire and current desire. With the template *B01* and data records in TABLE 10, a feature function will be generated which labels consecutive observations with a same desire if the time interval is short.

In summary, each feature template shown in TABLE 6 has its own meaning in describing certain properties of the domain of CoRE. During the course of the experiment, it took us several iterations to fine-tune these feature templates so that the domain of CoRE could be properly characterized and depicted. Based on the prepared training file and template file, a CRF model can be built by CRF++.

4.5 Desire Inference using Hidden Markov Model

To validate our hypothesis on the advantage of using CRF for desire inference over HMM, we conducted a HMM-based inference experiment as a comparison. The set of training data and test data used to build HMM is the same as that mentioned in the previous section. However, the column "Time Interval" (in TABLE 4) is not used because only one class of visible states can be used to build the HMM, so only the column "Observation" has been used. More specifically, the emission probability Matrix *O* (introduced below) in HMM specifies the relations between invisible states (desires) and one class of visible states (observations), and the two types of proba-

bility matrixes, Π and T, can only describe desires.

Jahmm [56], a Java implementation of HMM related algorithms, is adopted as the computation tool. Because the Baum-Welch algorithm applied in Jahmm for parameter estimation can only find a local minimum of its optimum function, a critical step in building an HMM model is to provide an initial guess of values in the following three basic probability matrixes:

1. Π : initial probability distributions of desires. E.g., $P(d_i)$ represents the probability of a user's desire being d_i at time *0* (the beginning of an observation sequence).
2. T: matrix of desire transition probability distributions. $P_T(d_i, d_j)$ represents the conditional probability of a user's desire being d_j at time $t+1$ given that it is d_i at time t, for any $t \geqslant 0$.
3. O: matrix of desire-observation emission probability distributions. $P_O(d_i, o_j)$ represents the conditional probability of the observation being o_j given that the desire is d_i, at any time point.

Based on the domain of CoRE and the training data, the best way that we found through multiple trials to compute the estimation of the three kinds of probabilities is the following:

- $P_\Pi(desire_1) = num(desire_1\_seq)/num(all\_seq)$, in which $num(desire_1\_seq)$ is the number of sequences in which the desire in the first data record is $desire_1$, and $num(all\_seq)$ is the number of all sequences;
- $P_T(desire_1, desire_2) = trans\_num(desire_1, desire_2) / trans\_num(desire_1, anydesire)$, in which $trans\_num(desire_1, desire_2)$ is the number of transitions from $desire_1$ to $desire_2$ for all consecutive data records, and $trans\_num(desire_1, anydesire)$ is the number of transitions from $desire_1$ to any desire;
- $P_O(desire_1, observation_1) = emis\_num(desire_1, observation_1) / emis\_num(desire_1, anyObservation)$, in which $emis\_num(desire_1, observation_1)$ is the number of emissions from $desire_1$ to $observation_1$ in all data records, and $emis\_num(desire_1, anyObservation)$ is the number of emissions from $desire_1$ to any observation.

Then we trained our HMM based on the initial estimation of the model in Jahmm iteratively. The model was more accurate when the number of iterations is higher. The trained HMM was used to perform desire inference on the test data.

4.6 Inference Result Analysis in the First-Round Experiment

TABLE 11
FORMAT OF DESIRE INFERENCE RESULT USING CRF++

Test Data			Inferred Desire/ Inference Probability
Observation	**Time Interval**	**Desire**	
……	……	……	…… # 0.075707
clickMenuAllPapers	30s	ViewAllPapers	ViewAllPapers/0.925086
clickLogin&LoginGood	10s	ViewAllPapers	ViewAllPapers/0.957425
clickMenuAllPapers	10s	ViewAllPapers	ViewAllPapers/0.982439
……	……	……	……

TABLE 12
INFERENCE ACCURACY OF DIFFERENT TEMPLATES

	Template 1 U01, B01~B04	Template 2 U01~U06	Template 3 U01~U06, B01~B04	Jahmm
%Mislabeling[a]	11.2405% (444/3950)	14.0759% (556/3950)	8.9367% (353/3950)	26.6329% (1052/3950)
Avg-P(Des-Infer)[b]	0.808979	0.749702	0.847904	NA
Avg-P(Seq-Infer)[c]	0.234812	0.0971787	0.286588	NA

[a]*Ratio of mislabeled observations to all observations*
[b]*Average inference probability of single observations*
[c]*Average inference probability of experiment sessions*

This step corresponds to Step 3 in our methodology (in section 3.3). Separated from training data, the rest of the data records were used as test data in the first-round experiment, containing 3,950 data records and 211 experiment sessions. The format of the test data was the same as that of the training data. Based on the hand-built CRF model (introduced in section 4.4), desire inference was performed on the test data using CRF++. TABLE 11 shows a sample of the results of desire inference.

Each inference result contains the inferred desire and the inference probability. Meanwhile, the overall inference probability of the output desire sequence for each experiment session is also given, e.g., in TABLE 11, # *0.075707*. By examining the inference result, we found that the overall inference probability of the whole desire sequence has a large correlation to the length of the sequence, while the probability of each single output desire can better reflect the inference accuracy. We also found that when using the same training data and test data but different feature templates, the inference accuracy was different. To show a comprehensive comparison, TABLE 12 lists the inference results of using HMM and CRF

TABLE 13
OCCURRENCE RATE AND MISLABELING RATE OF OBSERVATIONS IN DIFFERENT INFERENCE PROBABILITY RANGES

Inference Probability Range[a]	< 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
%Occurrence[b]	0.10%	1.29%	1.82%	3.22%	3.92%	4.15%	5.19%	6.43%	10.15%	63.72%
%Mislabeling[c]	75%	60.78%	45.83%	44.09%	41.29%	27.44%	19.02%	12.60%	7.48%	0.79%

[a]*The range of the inference probability for an observation*
[b]*The ratio of the number of observations with inference probability in the range to the total number of observations*
[b]*The ratio of the number of mislabeled observations to the total number of observations with inference probability in the range*

models with different types of feature templates.

The results in TABLE 12 show that overall CRF has far higher inference accuracy than HMM, and it gives better inference results (more accurate) when more meaningful feature templates are used and the domain is described more thoroughly.

To identify users' divergent behaviors that often reflect their new intentions (introduced in section 3.3), it was necessary to choose data records that had high probability to indicate such behaviors in an effective way. In our methodology, we believe that observation records corresponding to users' divergent behaviors will most likely be labeled with desires which are not consistent with their real desires because the CRF model does not explain these behaviors very well. However, in practice, since users do not report their real desires, it is not feasible to detect users' divergent behaviors through comparing the inferred desires with the self-reported ones. One alternative way is to look into the inferred desires with low inference probability, since we anticipate that inference accuracy will decrease/increase progressively with inference probability. Based on our analysis on data records, we found that basically there was an inverse relationship between inference probability and mislabeling probability, i.e., an observation is more likely to be mislabeled if its inference probability is lower. As shown in TABLE 13, the mislabeling rate (%Mislabeling) is higher for those observations with lower inference probability. In practice, since we cannot study all data records, we focused on analyzing observations with low inference probability first. In our experiment, we mainly focused on studying data records with a desire inference probability lower than 0.5.

4.7 New-Intention-Detection Case Study

This step corresponds to Step 4 in our methodology (in section 3.3) and system evolution process (in section 3.4). Three new intention detection methods have been applied and demonstrated with illustrative examples based on the first-round experiment results shown below:

1. Method I: Analysis of divergent behavior

As introduced in the previous section, we focus on studying observations with low inference probability (< *0.5*). These observations are more likely mislabeled, i.e., the CRF model cannot accurately explain these behaviors so they are probably divergent behaviors. Here we give some examples to demonstrate how to detect users' divergent behaviors and use them for system improvement.

a. New_Intention 1: some users often click button *hide the above selection form* to hide the selection form (see Fig. 7) immediately after entering the page *All Papers*.

As shown in TABLE 14, the observation *clickHideSelection* is labelled with *ViewAllPapers* as the inferred desire with a low probability *0.401886*, and its previous observation is also labelled with a low-probability desire, while its following observation is labelled with a very high-probability desire. In this case, this particular observation (*hiding the selection form*) can be easily singled out as a divergent behavior for analysis. To understand it, we proceeded with the assumption that the user might think the selection form is cumbersome when he tries to view the information of papers because it is too lengthy and takes up too much screen space.

TABLE 14
EXAMPLE SEGMENT IN THE DESIRE INFERENCE RESULTS

Observation	Time Interval	Inference Results
clickMenuAllPapers	10s	ViewAllPapers/0.510326
clickHideSelection	60s	ViewAllPapers/0.401886
clickPaperInfos&PaperID	60s	ViewAPaperInfo/0.986832
clickPaperInfos&PaperID	30s	ViewAPaperInfo/0.995394

TABLE 15
EXAMPLE SEGMENT IN THE DESIRE INFERENCE RESULTS

Observation	Time Interval	Inference Results
……	…	……
clickSubmit&ShortLimitation	60s	SubmitComment/0.499967
clickSubmit&NoCategory	20s	SubmitComment/0.333981
clickSubmit&CommentGood	10s	SubmitComment/0.380183

Two options were considered to solve this problem: 1) make the selection form initially hidden on the page *All Papers* (users can make it visible by clicking the link *show the selection form*); 2) show a simplified selection form initially (users can make it complete by clicking the link *expand the selection form*). To assess which option is better, we did a statistical analysis of users' behaviors after entering the page *All Papers*. There are four kinds of behaviors: *a)* hide the selection form to view the information of a paper; *b)* directly view the information of a paper; *c)* filter papers; *d)* others (exclude users' aimless behaviors). The number of occurrences of each behavior is: *49*, *52*, *67*, *19*, respectively. Because the frequency of behavior *c)* is highest, it is better to keep the selection form while shrinking it to a small size. Therefore, we proposed a modification to the system, that is to simplify the selection form with most frequently searched items (key words in the title and publication type) and add a new link *expand selection form* while initially hiding the selection form (see Fig. 7).

b. System_Drawback 1: As shown in TABLE 15, there are two consecutive error messages corresponding to records in the second and third rows. The first error is *ShortLimitation* and the second one is *NoCategory*. Such cases are unexpected because the user should have selected the category when the first error occurred, otherwise the first error message will also contain *NoCategory*. After looking into the original system design, we learned that the second error occurred because the system cleared users' previous category selection when the first error occurred; however, the user did not notice such a change. The corresponding modification on the system is to let it always keep users' category selection when they submit a comment.

Other examples of users' divergent behaviors detected in our experiment were:

c. System_Drawback 2: Click the button *Submit* twice or more on the page *submit/edit a comment*: The

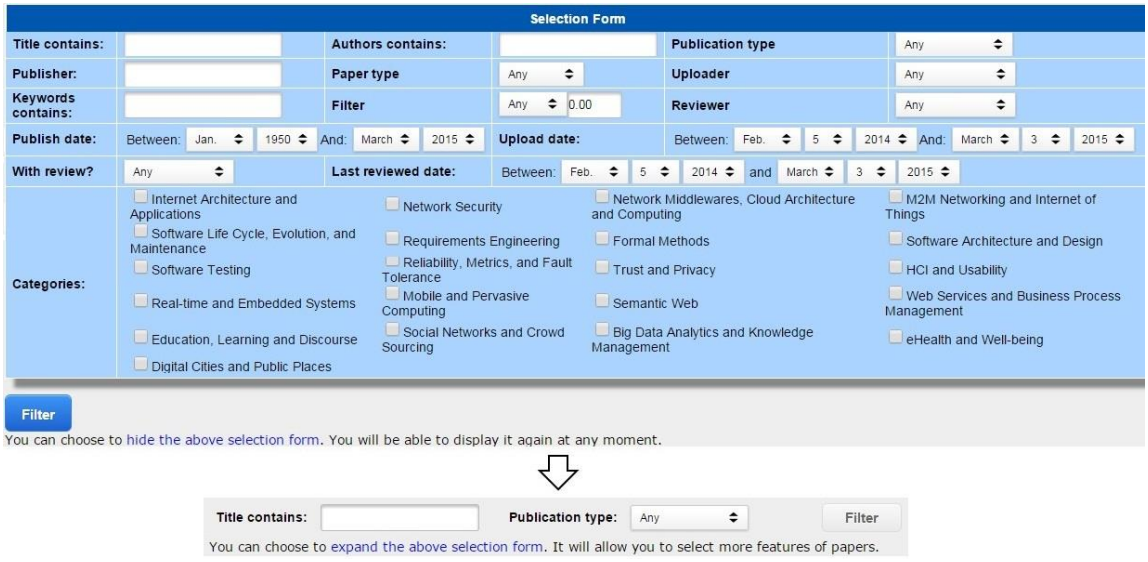

Fig. 7. The improvement of selection form (upper: selection form in *CoRE* Version I, lower: selection form in *CoRE* Version II).

Fig. 8. The new link for viewing the submitted comment and for viewing the uploaded paper.

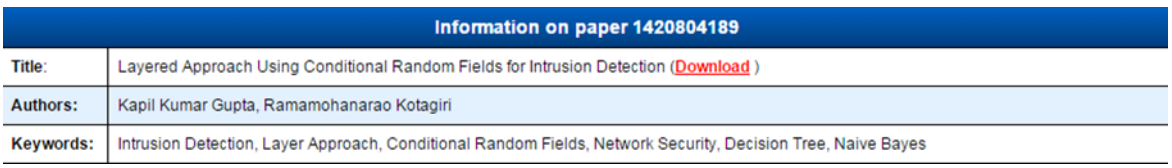

Fig. 9. The new link for downloading a paper.

TABLE 16
DESIRE TRANSITIONS IN THE FIRST-ROUND EXPERIMENT

Desire Transition	Occurrences	Ratio
UploadPaper → ViewMyPaperInfo[a] / UploadPaper→Any[b]	23/50	46%[c]
SubmitComment → ViewMyCommentInfo / SubmitComment→Any	35/52	67.31%
ViewAPaperInfo → DownloadPaper / Any→DownloadPaper	98/160	61.25%

[a]*The desire transition from "UploadPaper" to "ViewMyPaperInfo"*
[b]*The desire transition from "UploadPaper" to any other desire*
[c]*The ratio of desire transition a to desire transition b*

reason why users did this might be because the message for successful submission was not clear. Our proposed modification was to use bright color and bold font to make the message more visible (see Fig.8).

d. System_Drawback 3: Users consecutively input wrong passwords on the login page and the wrong passwords contain a space. One possible reason is that the user may copy the password with an extra space from somewhere. A possible modification is to give a hint that the password shall contain no spaces.

2. Method II: Analysis of desire transition

Another way to find users' potential new intentions is to study the desire transitions. The following TABLE 16 shows some examples of desire transitions in the first-round experiment.

New intentions can be defined for frequently occurring desire transitions to make them smoother, and the system can be accordingly modified to simplify users' operations. For example, for desire transitions *UploadPaper* → *ViewMyPaperInfo* (New_Intention 2) and *SubmitComment* → *ViewMyCommentInfo* (New_Intention 3), new functions can be added to the system to display a link of the submitted paper/comment on the submission page right after the paper/comment is successfully submitted (see Fig. 8). For the desire transition *ViewAPaperInfo* → *DownloadPaper* (New_Intention 4), a link can be added on the page of paper information to allow users to directly download the paper (see Fig. 9).

2. Method III: Analysis of erroneous behavior

If an erroneous behavior appears in the observation set many times (e.g., >= *5 times*), it can be viewed as a common error. A special case is users' aimless behaviors that may trigger errors that are not useful for analyzing users' desires. In this case, CRF is useful to exclude such noise in the data, and the aimless behaviors are usually labelled with desire "test" by the CRF model, which encodes many "test" behaviors that will be ignored for new inten-

tion detection. After removing the noise in the data, the occurrences of users' erroneous behaviors can be counted using simple mathematical methods.

Some examples of common users' erroneous behaviors detected in the first-round examples are:

a. New_Intention 5: No file was uploaded when editing the information of a paper. Such erroneous behavior occurs probably because users do not want to upload a file when editing the paper information, which can be considered a new intention. The corresponding system modification is to allow users to edit the paper information without uploading a file.
b. System_Drawback 4: Number of words in *comment details* is fewer than the minimum limit when submitting a comment. Users often consecutively encountered this error because they probably have no idea how many words they have typed in. One possible improvement can be that the system shall always display the number of words left to meet the minimum limit.
c. New_Intention 6: Chinese/Japanese comments are not accepted even though they contain enough words. This is due to the fact that the word count function in the system only counts spaces in a comment, which is not suitable for comments written in other languages that contain no spaces. One corresponding improvement could be that the system should be enhanced to support non-English comments.
d. System_Drawback 5: Missing key information (normally DOI, keywords) when uploading a paper. One corresponding improvement could be that the system should give clear tips on the uploading page to tell users what information is mandatory before uploading a paper.
e. New_Intention 7: Non-PDF files are uploaded but not accepted by the system. One corresponding improvement could be that the system is enhanced to support non-PDF files.

Based on the new intentions and system drawbacks we discovered, we made corresponding modifications on the system, and the original CoRE system was evolved to its next version – CoRE Verision II.

4.8 Validation of System Improvement in the Second-Round Experiment

To demonstrate and validate the potential improvement of the system, a second-round experiment was done on the evolved system – Core Version II, by following exactly the same process as what we did in the first round, which means that we strictly followed Step 1 to Step 4 in section 3.3 for one more iteration. The only difference was that we recruited some new participants in the second round to make the comparison between two versions fair and even. In this section, we will not focus on the technical execution of our methodology steps due to space limitations. Instead, we will mainly focus on evaluating the improvement of CoRE version II over its predecessor.

Similar to the data analysis in the first-round experiment, a CRF model was built using *2,984* training data records, and was used for desire inference on a test data set with *3,947* records. The results of desire inference in two rounds are shown in TABLE 17, from which we can see that the overall inference probability and inference accuracy in the second-round experiment have been significantly improved compared with those results in the first round.

TABLE 17
OVERALL DESIRE INFERENCE ACCURACY

	%Mislabeling	Avg-P(Des-Infer)	Avg-P(Seq-Infer)
Second Round	3.9524% (156/3947)	0.905067	0.314128
First Round	8.9114% (352/3950)	0.848429	0.28694

TABLE 18
THE USE OF NEW LINKS FOR VIEWING PAPER INFORMATION

Observation	Occurrences	Ratio
Upload Paper → View Paper Info[a] / Upload Paper → Any[b]	28/32	87.5%
Submit Comment → View Paper Info / Submit Comment → Any	82/92	89.13%
Edit Paper → View Paper Info / Edit Paper → Any	6/9	66.67%
Edit Comment → View Paper Info / Edit Comment → Any	16/22	72.73%

[a]*Use the new link to view paper info after uploading a paper*
[b]*Any behavior after uploading a paper*

Some example system improvements are described below:

1. Simplified selection form (New_Intention 1, Fig. 7)

To evaluate the benefit of the simplified selection form in the new system, we did a statistical analysis of users' behaviors after entering the page *All Papers* on which the selection form is located. There are four kinds of behaviors: a) use the simplified selection form to filter papers; b) expand the selection form to filter papers; c) view the information of a paper; d) others (exclude users' aimless behaviors). The number of occurrences of each behavior is: *54*, *41*, *63* and *30*, respectively. Since users used the simplified selection form more often compared with the frequency that they expanded the selection form, we can draw the conclusion that it is more suitable to display the simplified selection form on the page *All Papers* instead of the expanded version.

2. The link of the newly submitted comment in the message of successful submission/editing, and a link of the newly uploaded paper in the message of successful uploading/editing (New_Intention 2&3, Fig. 8).

TABLE 18 shows the use of the new links when users upload/edit a paper or submit/edit a comment, and it can be seen that the new link was frequently used. Meanwhile, consecutive *Submit* operations did not occur at all in the new system, which also demonstrate the effectiveness of the modification.

3. *Download* link on *paperinfo* page (New_Intention 4, Fig. 9)

TABLE 19
COMPARISON BETWEEN TWO ROUNDS EXPERIMENTS

Task	Round	# of Occurrence	Avg. Time Spent (s)		Successful		Errors[a]		Consecutive Errors[b]	
			Time	Improvement	# (rate)	Improvement	# (rate)	Improvement	# (rate)	Improvement
Upload Paper	1st	55	387.87	↓55% (Reduced)	38 (69%)	↑41% (Increased)	28 (51%)	↓94% (Reduced)	14 (25%)	↓100% (Reduced)
	2nd	32	173.57		31 (97%)		1 (3%)		0	
Submit comment	1st	42	287.26	↓28%	38 (90%)	↑2%	23 (55%)	↓11%	15 (36%)	↓83%
	2nd	100	207.79		92 (92%)		49 (49%)		6 (6%)	
Edit Paper	1st	16	47.81	↓1%	8 (50%)	↑100%	5 (31%)	↓100%	0	–
	2nd	9	47.33		9 (100%)		0		0	
Edit comment	1st	29	105.38	↓39%	20 (69%)	↑20%	0	–	0	–
	2nd	12	64.25		10 (83%)		0		0	

[a]*The times of at least one error occurs when submitting/editing a paper/comment*
[b]*The number of occurrences of consecutive errors*

The observed user behaviors in the second-round experiment show that users prefer the new link to the old one after viewing the paper information, considering that the total use count of the new link is 146, while that of the old link is 60.

To demonstrate the overall system improvement, we focused on four major user tasks/operations, which are uploading/editing paper, submitting/editing comment, and evaluated users' performance on those tasks in both rounds of experiments. The comparison result is shown in TABLE 19. As we can see, the success rate of four tasks all increased in the second-round experiment, while the time cost and the error occurrence rate, especially the consecutive error occurrence rate, significantly decreased.

Overall speaking, through the evolution from CoRE version I to II, the usability of the system has been improved, which means the evolution based on our proposed methodology was satisfactory.

4.9 Summary of the Experiment

Based on the above demonstration of the execution of the experiment and the analysis of the results, we now can revisit those three hypotheses proposed at the beginning of this section, and see if each of them is valid or not.

1. Desire inference with CRF model:

According to inference accuracy results of the first round experiment (TABLE 12, section 4.6), CRF model was able to deliver highly accurate desire inference results when appropriate feature templates were designed and applied. The actual inference accuracy rate was 91%, which met our expectation (> 90%). Additionally, the average inference probability (confidence) was about 0.85. The combination of high inference accuracy and confidence proves the fact again that CRF model is good at sequential labelling. On the contrary, using the same training and test data sets, HMM fell short on the accuracy as expected, with a less than 74% accuracy rate.

2. Detection of new intention & system drawback:

According to our case study in section 4.7, 7 new intentions and 5 system drawbacks were identified for CoRE Version I using the three newly proposed methods (in section 3.3). As in CoRE Version II, our methods were also effective to detect new intentions in the domain of CoRE. For example, a phenomenon we discovered through desire transition analysis (Method II) is that the user often reviews or downloads paper one by one after filtering papers using the selection form. Since these consecutive *Download Paper* desires occur frequently, we proposed to introduce a compound desire called *Download All Papers* by adding a link for downloading all selected papers at once.

The above results and example also demonstrate that our methodology is able to continuously explore users' new intentions and enhance the system to adapt to the ever-changing users' requirements.

3. Successful and rapid system evolution:

Based on our analysis results in section 4.8, the usability of CoRE was significantly improved in many aspects, so the system successfully evolved from version I to version II. In each round of our experiment, we collected behavioral and contextual information from 60 users for one month, and got around *10,000* raw data records. Then, it took about one month for one domain expert to process and analyze data, infer desires & detect new intentions, and eventually evolve our experiment system. However, in practice, we believe that our methodology can be even more effective, especially for those systems or applications with a large user base such as social network operators. One of the technical merits of our methodology is that we use real system usage data to drive and enable system evolution, so the semi-automated evolution process (in support of domain expert's decision making) should be applicable to real-life situations. As we show in section 4.7, those newly detected intentions and system drawbacks are quite intuitive and self-explanatory. Thus, it will not take much intellectual effort to figure out the corresponding new requirements and system remedies.

In summary, our two-round exploratory experiment was successful, in the sense that all hypotheses were proven valid, and our proposed methodology was demonstrated to be effective.

5 THREATS TO VALIDITY

In this section, we will discuss some potential threats to

the validity of our proposed methodology and experiment from the following perspectives:

1. Threats to construct validity – concerns regarding the design of our methodology and the measurement of our metrics
2. Threats to internal validity – concerns regarding alternate explanations for our results
3. Threats to external validity – concerns regarding the generalizability of our results

5.1 Threats to Construct Validity

One potential concern about the validity of our proposed methodology is the theoretical foundation (section 3.1 & 3.2), in which the computational models for situation and intention have been defined. On one hand, we have carefully and thoroughly characterized and described the human-centric context-aware domains with certain causal relationships among users' desire, actions and relevant context values by making a number of assumptions and axioms (section 3.1). On the other hand, some exceptional cases might not be covered or might newly emerge along with the evolution of these domains. Simply put, any threat to the validity of theoretical foundation can jeopardize our experiment.

There are also a couple of concerns regarding the measurement of our metrics:

1. Interaction of normal operations and reporting desires

To directly validate our desire inference results, we asked participants to report their real-time desires while operating on the system, which is considered a necessary part of our experiment. However, imposing additional tasks (reporting desires) might interfere with participants' normal thinking process and might further influence their understanding of the system. An example would be that the dropdown desire list on each webpage of CoRE contains all the predefined desires related to the system, and participants may be able to learn from it, and consequently adjust their behaviors. Although such phenomenon is undesired, the value of acquiring participants' self-reported desires significantly outweighs its adverse effect.

2. Hypothesis guessing

Due to our local IRB regulations, we needed to fully introduce and explain our experiment to all participants, including our basic experiment motivation and detailed procedure. Although we tried to be very succinct on our experiment goals and technical methods, participants might have been able to guess what hypothesis we wanted to validate, and even what data could help prove the validity of those hypotheses. Such a possibility cannot be eliminated, since the majority of the participants are majors in computer science, computer engineering, or related fields. However, we doubt the possible impact of such phenomenon on the quality of our data, because the CoRE system appears to be a typical online library, which looked intuitive and familiar to most of the participants, and they could finish most of the tasks without thinking too much about the system and the experiment itself. Therefore, we expect the impact of hypothesis guessing, acknowledged as a potential threat, to be minimal in our experiment.

5.2 Threats to Internal Validity

For internal validity, we look into whether there is sufficient evidence to support the conclusions of our experiment, specifically in the following aspects:

1. Design of comparison model (HMM)

To demonstrate that CRF can perform well in desire inference, we designed a comparison model, HMM, and evaluated desire inference results of both methods side by side in TABLE 12 (section 4.6). However, since HMM is designed differently from CRF, it is hard to construct the comparison absolutely fair and even. To be specific, as introduced in section 4.5, one critical step for building a HMM is to design the initial estimation of values in the three matrixes, while the critical step for building a CRF model is to design the feature templates. Although the comparison result shows that CRF outperformed HMM, the possibility of not having the best HMM still cannot be eliminated. What we could do was to try with different HMMs, and pick the best one to compare with CRF. But we cannot know if the one we chose was really the best one. Such an issue commonly exists in many related statistical studies, and we think our way to handle it is acceptable.

2. Prediction of new requirements

The new requirements revealed from the analysis of divergent behaviors were mainly based on our subjective judgement, and were validated through usability analysis between two versions of systems indirectly, instead of directly inquiring the users. Therefore, it is still debatable whether our predicted new requirements were in fact the users' new requirements, or just designer's own preferred system evolution path. However, when evolving a real-world system, engineers may face the same problem, and one way to truly validate newly found requirements is to evaluate the new system through usability analysis. But, it must be admitted that improvement of system usability may be due to different reasons, which might not be the implementation of the specific new requirements. For example, the reason that time spent for uploading a paper (TABLE 19) decreased significantly may be because some previously required information was removed, such as DOI. So, a reasonable conclusion that we can draw for the evolution of CoRE is that its overall system usability has been improved by implementing new requirements and necessary remedies.

3. Arbitrary design of desire granularity

To build the domain knowledge, a prerequisite is to define and enumerate all the anticipated desires. The quality of the predefined desire set is crucial for the accuracy of desire inference. The granularity of desire must be properly designed. If the desire is too fine-grained, it will create too many labels for CRF to choose from, and will probably confuse the CRF model. Or, if the desire is too coarse-grained, it may not be able to generalize the causal relations among desire, actions, and context values. In our experiment, the desires in the domain of CoRE were relatively easy to define. However, we have not tested other domains for the feasibility and the difficulty of designing

a set of appropriate known desires.

4. Statistical conclusion validity

In essence, we use statistical methods (TABLE 13) to identify candidate observations for new intention detection, and we basically depend on the quality and characteristics of sample data (mislabeled records). When the sample space is bigger, our methodology is more likely to work well. However, when the sample space is too small or has too much noise (false positive), our methodology, or any method based on statistical analysis, may not give an ideal result. One example is that for a matured system (domain), there might be only a few new intentions, which correspond to only a few observations. While having such a small target mixed with some noise in the data, it is hard to elicit the real requirements of our interests.

5.3 Threats to External Validity

In our methodology, we described its application domain as a human-centric context-aware domain. Our experiment was conducted on a system that is supposed to be a representative application. However, as we stated in Assumptions 1, 2 and 3, the domain should be context aware, and belong to a single agent who has a single desire at any instant. If an application domain does not have these attributes, our methodology may not be applicable. For example, if users interact frequently, and their actions directly influence each other's desire, their behaviors will be very hard to describe and desire inference is not able to be performed (using our methodology). However, we do believe that our methodology has broad application prospect because many applications can be characterized as human-centric context-aware domains as we described in section 3.1. Some example systems to apply our methodology are listed as follows:

1. The server end of dynamic web-based systems is responsible for processing and responding to user requests, where monitoring programs can be deployed to capture user operations, inputs, submissions and contents on the pages.
2. Mobile applications that can sense users' physical status and environment conditions, and adapt their behavior accordingly. In fact, some research work has been conducted to effectively select services for adaptation according to the user's current context [57].
3. Home automation [58] is another promising application area, especially for smart homes for elderly and disabled [59], in which most user behaviors can be monitored and there is user demand to increase the quality of life without attention of caregivers or institutional care.

6 Conclusions

This paper presents an effective method of applying Conditional Random Fields as the mathematical foundation to infer human desires based on observations of their actions and relevant environmental context values, and further explore their new intentions for driving system evolution. With an exploratory experiment, we show that the accuracy of desire inference is high (> 90%) by using CRF compared with the result of using HMM (75%). Furthermore, the three methods for detecting new intentions (section 3.3) are shown to be effective to obtain users' new potential intentions and further reveal their new requirements on the system or draw attention to system drawbacks that are useful for system evolution. The experiment results verified our hypothesis about rapid discovery of new requirements and system evolution based on our methodology.

However, there are some limitations of our work. Besides the application scope delimited by Assumption 1, 2 and 3 (section 3.1), our methodology is only able to capture users' new functional requirements, not non-functional requirements such as high usability, fast response time, etc. In addition, the analysis of potential new intentions and users' new requirements still requires a lot of human effort and may not consistently give the best result. Therefore, the gap between desire inference and new intention detection presents steep research challenges for us to tackle in the future:

1. The methodology proposed in this paper can only automatically detect users' potential desires that are related to new intentions and system drawbacks. The work of eliciting and verifying new intentions and system drawbacks, as well as designing and implementing new system functionalities would require human effort. From a knowledge engineering perspective, further investigation is plausible as to how to acquire design wisdom from raw-data analytics.
2. In our methodology, we restrict the domain to a single agent domain, which is too restrictive to characterize multiple agent domains, especially in systems where agents frequently interact with each other. To extend the applicability of our methodology, there are some interesting research questions to be answered. For example, "how to characterize the causal relationships among agent's desires, actions, and context values", "can CRF encode such causal relationships among other research questions?", etc.
3. In our experiment, we demonstrated how to apply our methodology to a web-based system. In later sections, we also discussed the feasibility of applying our methodology on different services. To verify that the method is really applicable for other systems, experimental validation is still much needed. Some valuable candidates are mobile applications, smart home systems, etc., which are becoming prevalent nowadays with rapidly changing user requirements.

Acknowledgment

We thank all the colleagues in the Software Engineering Lab for their advice and help on our work. We are also indebted to more than 120 participants who contributed to our experiment work. We would also like to acknowledge the guidance and support from the IRB Committee at Iowa State University.

REFERENCES

[1] K.H. Bennett and V.T. Rajlich, "Software Maintenance and Evolution: a Roadmap," *Proc. Conf. The Future of Software Engineering, ACM*, pp. 73-90, 2000.

[2] I. Borne, S. Demeyer, and G.H. Galal, "Object-Oriented Architectural Evolution," *Object-Oriented Technology ECOOP'99 Workshop Reader Lecture Notes in Computer Science*, vol. 1743, pp. 57-79, 1999.

[3] D. Rowe and J. Leaney, "Evaluating evolvability of computer based systems architectures - an ontological approach," *Proc. Int'l Conf. and Workshop on Engineering of Computer-Based Systems*, pp. 360-367, 1997.

[4] H. Ming, K. Oyama, and C. Chang, "Human-Intention Driven Self Adaptive Software Evolvability in Distributed Service Environments," *Proc. 12th IEEE Int'l Workshop Future Trends of Distributed Computing Systems (FTDCS '08)*, pp. 51-57, 2008.

[5] A. Iliasov and A. Romanovsky, "CAMA: Structured Communication Space and Exception Propagation Mechanism for Mobile Agents," *Proc. ECOOP Workshop Exception Handling in Object Oriented Systems: Developing Systems That Handle Exceptions*, pp. 75-87, 2005.

[6] H.P. Breivold, I. Crnkovic, R. Land, and S. Larsson, "Using dependency model to support architecture evolution," *Proc. 4th International ERCIM Workshop on Software Evolution and Evolvability (Evol'08) at the 23rd IEEE/ACM International Conference on Automated Software Engineering (ASE'08)*, pp. 82-91, 2008.

[7] M.M. Lehman, J.F. Ramil, P. Wernick, D.E. Perry, and W.M. Turski, "Metrics and Laws of Software Evolution—the Nineties View," *Proc. Fourth Int'l Software Metrics Symposium*, pp. 20-32, 1997.

[8] H.P. Breivold, I. Crnkovic, and M. Larsson, "Software architecture evolution through evolvability analysis," *Journal of Systems and Software*, vol. 85, Issue 11, pp. 2574–2592, November 2012.

[9] C.L. Nehaniv and P. Wernick, "Introduction to software evolvability," *Proc. Third Int'l IEEE Workshop on Software Evolvability*, pp. 6-7, 2007.

[10] S. Robertson and J. Robertson, *Mastering the Requirements Process: Getting Requirements Right (3rd Edition), Chapter 14: Requirements and Iterative Development*, Addison Wesley, 2012.

[11] M. Marschall, "Transforming a Six Month Release Cycle to Continuous Flow," *Proc. Assoc. Geographic Information Laboratories Europe Conf. (AGILE '07)*, pp. 395-400, 2007.

[12] App Annie, "Decision-making platform for the entire mobile app economy," Retrieved from https://www.appannie.com, 2015.

[13] J. Gorinsek, S. Van Baelen, Y. Berbers, and K. De Vlaminck, "Managing Quality of Service during Evolution Using Component Contracts," *Proc. ETAPS 2003 Workshop Unanticipated Software Evolution (USE '03)*, pp. 57-62, 2003.

[14] O. Saliu and G. Ruhe, "Supporting Software Release Planning Decisions for Evolving Systems," *Proc. 29th IEEE/NASA Software Eng. Workshop (SEW-29)*, pp. 14-26, 2005.

[15] C. Salinesi and A. Etien, "Compliance Gaps: A Requirements Elicitation Approach in the Context of System Evolution," *Proc. 9th International Conference on Object-Oriented Information Systems (OOIS 2003)*, pp. 71-82, 2003.

[16] W. Jiang, H. Ruan, L. Zhang, P. Lew, and J. Jiang, "For User-Driven Software Evolution: Requirements Elicitation Derived from Mining Online Reviews," *Proc. 18th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining (PAKDD 2014)*, pp. 584-595, 2014.

[17] C. Chang, K. Oyama, H. Jaygarl, and H. Ming, "On Distributed Run-Time Software Evolution Driven by Stakeholders of Smart Home Development," *Proc. Second Int'l Symp. Universal Comm. (ISUC '08)*, pp. 59-66, 2008.

[18] R. Laddaga, "Self Adaptive Software—Problems and Projects," *Proc. Int'l Workshop Software Evolvability (SE '06)*, pp. 3-10, 2006.

[19] J. Xia, C. Chang, T. Kim, H. Yang, R. Bose, and S. Helal, "Fault-Resilient Ubiquitous Service Composition," *Proc. Third IET Int'l Conf. Intelligent Environments (IE '07)*, pp. 108-115, 2007.

[20] C. Chang, H. Jiang, H. Ming, and K. Oyama, "Situ: A Situation-Theoretic Approach to Context-Aware Service Evolution," *IEEE Trans. on Services Computing*, vol. 2, no. 3, July-September, 2009.

[21] L. Rabiner, "A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition," *Proc. IEEE*, vol. 77, no. 2, pp. 257-286, 1989.

[22] H. Xie, C. Chang, "Detection of New Intentions from Users Using the CRF Method for Software Service Evolution in Context-Aware Environments", *Proc. 39th Annual IEEE Computer Software and Applications Conference*, 2015 (Accepted).

[23] C. Sutton and A. McCallum, *An Introduction to Conditional Random Fields for Relational Learning*. MIT Press, 2006.

[24] J. Lafferty, A. McCallum and F. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," *Proc. of 18th International Conference on Machine Learning*, Morgan Kaufmann, pp. 282-289, 2001.

[25] T. Mens and S. Demeyer, *Software Evolution*. Springer-Verlag Berlin Heidelberg, 2008.

[26] IEEE Std. 610.12-1990, *IEEE Standard Glossary of Software Engineering Terminology*, IEEE, New York, 1991.

[27] P.I. Okwu and I.N. Onyeje, "Software Evolution: Past, Present and Future," *American Journal of Engineering Research (AJER)*, vol. 03, Issue 05, pp. 21-28, 2014.

[28] Liguo Yu and Alok Mishra, "An Empirical Study of Lehman's Law on Software Quality Evolution," *International Journal of Software and Informatics*, vol. 7, Issue 3, pp. 469-481, 2013.

[29] B.P. Lientz and E.B. Swanson, *Software Maintenance Management, A Study of the Maintenance of Computer Application Software in Data Processing Organizations*. Addison-Wesley, Reading MA, 1980.

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

[31] H. Xie, L. Liu, and J. Yang, "i*-Prefer: Optimizing Requirements Elicitation Process Based on Actor Preferences", *Proc. 24th ACM Symposium on Applied Computing*, pp. 347-354, 2009.

[32] Thomas Keller, "Contextual Requirements Elicitation - An Overview," Seminar in Requirements Engineering, Department of Informatics, University of Zurich, 2011.

[33] G.E. Kniesel and R.E. Filman, "Unanticipated Software Evolution," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 17, Issue 5, pp. 307–377, 2005.

[34] G. Weiss, *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, 2000.

[35] R. Kelley, A. Tavakkoli, C. King, M. Nicolescu, M. Nicolescu, and G. Bebis, "Understanding Human Intentions via Hidden Markov Models in Autonomous Mobile Robots," *Proc. Third ACM/IEEE Int'l Conf. Human Robot Interaction (HRI '08)*, pp. 67-374, 2008.

[36] A. McCallum and W. Li, "Early results for named entity recognition with conditional random fields, feature induction and web-enhanced lexicons," *Proc. the seventh conference on Natural language learning at HLT-NAACL (CONLL '03)*, volume 4, pp 188-191, 2003.

[37] F. Peng, F. Feng, and A. McCallum, "Chinese segmentation and new word detection using conditional random fields," *International Conference on Computational Linguistics (COLING)*, pp. 562–568, 2004.

[38] Y. Liu, J. Carbonell, P. Weigele, and V. Gopalakrishnan, "Protein fold recognition using segmentation conditional random fields (SCRFs)," *Journal of Computational Biology*, vol. 13, no. 2, pp. 394–406, 2006.

[39] K. Sato and Y. Sakakibara, "RNA secondary structural alignment with conditional random fields," *Bioinformatics*, vol. 21, pp. 237–242, 2005.

[40] L. Getoor and B. Taskar, *An Introduction to Conditional Random Fields for Relational Learning (Edition 1)*. MIT Press, pp. 93–127, 2007.

[41] E. Chen, "Introduction to Conditional Random Fields," Retrieved from http://blog.echen.me/2012/01/03/introduction-to-conditional-random-fields/, 2012.

[42] T. Cohn, "Efficient inference in large conditional random fields," *Proc. the 17th European conference on Machine Learning (ECML'06)*, pp. 606-613, 2006.

[43] H.M. Wallach, "Efficient training of conditional random fields," Master's thesis, University of Edinburgh, 2002.

[44] K.K. Gupta, B. Nath, and R. Kotagiri, "Layered Approach Using Conditional Random Fields for Intrusion Detection," *IEEE Trans. on Dependable and Secure Computing*, vol. 7, Issue 1, pp. 35 – 49, 2010.

[45] Y. Shen, J. Yan, S. Yan, L. Ji, N. Liu, and Z. Chen, "Sparse hidden-dynamics conditional random fields for user intent understanding," *Proc. the 20th international conference on World wide web (WWW '11)*, pp. 7-16, 2011.

[46] O. Brill and E. Knauss, "Structured and Unobtrusive Observation of Anonymous Users and their Context for Requirements Elicitation," *Proc. IEEE 19th International Requirements Engineering Conference*, pp. 175–184, 2011.
[47] A. Alkhanifer and S. Ludi, "Towards a Situation Awareness Design to Improve Visually Impaired Orientation in Unfamiliar Buildings: Requirements Elicitation Study," *Proc. IEEE 22nd International Requirements Engineering Conference*, pp. 23-32, 2014.
[48] H. Xie, C. Chang, H. Ming, and K. Lu, "The Concepts and Ontology of SiSL: A Situation-Centric Specification Language", *Proc. Computer Software and Applications Conf. Workshops* (*COMPSACW '12*), pp. 301-307, 2012.
[49] L.T.F. Gamut, "Logic, Language, and Meaning," Intensional Logic and Logical Grammar, vol. 2, Chicago, IL: University of Chicago Press, 1991.
[50] V.J. Hodge and J. Austin, "A Survey of Outlier Detection Methodologies," *Artificial Intelligence Review*, vol. 22, Issue 2, pp. 85-126, 2004.
[51] S.B. Kotsiantis, "Supervised Machine Learning: A Review of Classification Techniques," *Proc. of the 2007 conference on Emerging Artificial Intelligence Applications in Computer Engineering: Real Word AI Systems with Applications in eHealth, HCI, Information Retrieval and Pervasive Technologies*, pp. 3-24, 2007.
[52] T. Kudo, "CRF++: Yet Another CRF toolkit," Retrieved from http://taku910.github.io/crfpp, 2014.
[53] X.H. Phan, L.M. Nguyen, and C.T. Nguyen, "FlexCRFs: Flexible Conditional Random Fields," Retrieved from http://flexcrfs.sourceforge.net, 2014.
[54] V. Prabhakaran, A.C. Arpaci-Dusseau, and R.H. Arpaci-Dusseau, "Analysis and Evolution of Journaling File Systems," *Proc. the USENIX Annual Technical Conference (ATEC '05)*, pp. 8-23, 2005.
[55] P. Rigaux, "The MyReview System," Retrieved from http://myreview.sourceforge.net.
[56] J.M. François, "An implementation of Hidden Markov Models in Java," Retrieved from https://code.google.com/p/jahmm, 2014.
[57] G.S. Thyagaraju and U.P. Kulkarni, "Design and Implementation of User Context aware Recommendation Engine for Mobile using Bayesian Network, Fuzzy Logic and Rule Base," *International Journal of Computer Applications,* vol. 40, No.3, pp. 47–63, 2012.
[58] Abi Research, "1.5 Million Home Automation Systems Installed in the US This Year," Retrieved from https://www.abiresearch.com/press/15-million-home-automation-systems-installed-in-th, 2015.
[59] P. Harmo, T. Taipalus, J. Knuuttila, J. Vallet, and A. Halme, "Needs and solutions - home automation and service robots for the elderly and disabled," *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2005)*, pp. 3201-3206, 2005.

Haihua Xie received the PhD degree in computer science from Iowa State University in 2015. Before that, he received the B.S. in Electrical and Information Science and Technology from Nanjing University of Posts and Telecommunications in 2006, and the M.E. in Software Engineering from Tsinghua University in 2009. His research interests include data analytics, context / Situation awareness, requirements engineering, software service evolution.

Jingwei Yang is a PhD candidate in the Department of Computer Science at Iowa State University. He received the B.S. in Automation from Zhejiang University in 2006, and the M.E. in Software Engineering from Tsinghua University in 2009. His research interests include software evolution, knowledge discovery, and requirements engineering.

Carl K. Chang is Professor of Computer Science, Professor of Human-Computer Interaction and Director of Software Engineering Laboratory at Iowa State University. He received a PhD in computer science from Northwestern University in 1982, and worked for GTE Automatic Electric and Bell Laboratories before joining the University of Illinois at Chicago in 1984. He joined Iowa State University in 2002 as Department chair of Computer Science, and completed three terms as its chairman in an eleven-year sprint. His research interests include requirements engineering, net-centric computing, situational software engineering and successful aging. Chang is 2004 President of IEEE Computer Society. Previously he served as the Editor-in-Chief for IEEE Software (1991-94). He received the Computer Society's Meritorious Service Award, Outstanding Contribution Award, the Golden Core recognition, and the IEEE Third Millennium Medal. In 2006 he received the prestigious Marin Drinov Medal from the Bulgarian Academy of Sciences, and was recognized by IBM with the IBM Faculty Award in 2006, 2007 and 2009. From 2007-1010 he served the Editor-in-Chief of IEEE Computer, the flagship publication of IEEE Computer Society. He is the 2012 recipient of the Richard E. Merwin Medal from the IEEE Computer Society. Most recently he was elected the 2014 Distinguished Alumnus by the National Central University in Taiwan, and received the 2014 Overseas Outstanding Contribution Award from China Computer Federation. Chang is an IEEE and AAAS Fellow, and member of the European Academy of Sciences.