A Statistical Analysis Approach to Predict User's Changing Requirements for Software Service Evolution

Haihua Xie , Jingwei Yang , Carl K. Chang , Lin Liu

PII: S0164-1212(17)30135-8
DOI: 10.1016/j.jss.2017.06.071
Reference: JSS 9995

To appear in: *The Journal of Systems & Software*

Received date: 9 July 2016
Revised date: 19 March 2017
Accepted date: 25 June 2017

Please cite this article as: Haihua Xie , Jingwei Yang , Carl K. Chang , Lin Liu , A Statistical Analysis Approach to Predict User's Changing Requirements for Software Service Evolution, *The Journal of Systems & Software* (2017), doi: 10.1016/j.jss.2017.06.071

Highlights:

- A CRF(Conditional Random Fields)-based method for inferring users' goals.
- A method to explore users' emerging intentions based on goal inference results.
- A software evolution process based on detection of users' emerging intention.
- An exploratory study on an online service with a user base of 120 participants.

A Statistical Analysis Approach to Predict User's Changing Requirements for Software Service Evolution

Haihua Xie, Jingwei Yang, Carl K. Chang and Lin Liu

H. Xie, J. Yang, and C. K. Chang are with the Department of Computer Science, Iowa State University, 226 Atanasoff Hall, Ames, IA 50011. E-mail: {haihxie, jwyang, chang}@iastate.edu.
L. Liu is with the School of Software, Tsinghua University, Beijing, China, 100084. E-mail: linliu@tsinghua.edu.cn.

Abstract—Evolution is inevitable for almost all software, and may be driven by users' continuous requests for changes and improvement, the enablement of technology development, among other factors. The evolution of software services can be seen as the evolution of system-user interactions. The capability to accurately and efficiently observe users' volatile requirements is critical to making timely system improvements to adapt to rapidly changing environments. In this paper, we propose a methodology that employs Conditional Random Fields (CRF) as a means to provide quantitative exploration of system-user interactions that often lead to the discovery of users' potential needs and requirements. By analyzing users' run-time behavioral patterns, domain experts can make prompt predictions on how users' intentions shift, and timely propose system improvements or remedies to help address emerging needs. Our ultimate research goal is to speed up software service evolution to a great extent with automated tools, knowing that the challenge can be undoubtedly steep. The evolution of an online research library service is used to illustrate and evaluate the proposed approach in detail.

Index Terms—Conditional Random Fields, human intention detection, goal inference, requirements, service, software evolution

1 Introduction

THE goal of software system evolution in this fast changing, modern world is to effectively adapt to highly volatile user requirements and operating environments [1]. Hence, the ability to rapidly evolve is considered an essential quality requirement for today's software systems [2]. Due to the unpredictable nature of user needs and environmental changes for systems that are socially embedded and human centric, making accurate and timely evolution decisions poses unparalleled technical challenges [3]. For a given software system, the lack of evolvability will lead to progressively decreasing satisfactory performance until it eventually becomes obsolete [4]. In order to keep up with the growing needs and desires of users, it is necessary to make evolution a planned task during the entire software lifecycle [5].

In practice, the trigger for software evolution in a human-centric system is the human desire for achieving something that cannot be achieved with the existing system. It typically starts with requirements that specify emerging user needs or new settings of system environments. Traditionally, these emerging requirements are elicited by user feedback collection conducted offline with certain latency. For software systems with a massive user base, user requirements can be diverse, and changes are often highly fast-paced. To remain competitive in business, new software development lifecycle models, such as agile software development, have been adopted to develop a work product quickly, with small, incremental releases [6]. 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, especially in the mobile application domain [7]. However, many questions naturally emerge, such as "what functions should be released in the next iteration?" or "what are the next most desired features in the queue?" Such questions are mostly answered only by intuition or the best educated guesses of product managers.

To make more rational evolution decisions, new approaches to eliciting evolving user requirements are needed. Recent studies on elicitation of emerging requirements focus on observing historical defects or analyzing delayed user feedback [8], [9], while little attention has been paid to exploring human intentions at runtime to drive system evolution [10]. In today's context-aware service-oriented data-driven environments, it is possible to observe user behaviors and environmental changes in real time and analyze or predict users' potential mental states to understand their true demands [11, 12, 13]. In fact, using sensor-laden data collection and appropriate mathematical models, elicitation of emerging requirements by employing some form of automated tools may become viable.

A previous work on human-intention-driven service evolution is the Situ framework [13], which aims to support rapid and iterative service requirements analysis of real-world systems. Situ builds a Hidden Markov Model (HMM) to deduce goals of an individual user from given observations, user's actions and environmental context values, and further analyze a user's potentially emerging intentions for driving service evolution. However, Situ encounters difficulties in goal inference because HMM is not able to sufficiently encode the causal relations among actions, context values and goals, nor conquer the complexity of goal transitions [14].

In order to accurately infer users' goals and effectively discover users' emerging intentions, a methodology that applies Conditional Random Fields (CRF) as the mathematical foundation is proposed to provide quantitative exploration of users' emerging requirements [15]. CRF is a class of statistical modeling methods used for encoding known relationships between observations and constructing consistent interpretations [16]. Here, CRF is used to build the mathematical model for goal inference, which is to infer users' goals based on runtime observations of their actions and environmental context values. By nature, goal inference [17] can be regarded as the process of labeling an observation sequence with goals. CRF is proven more capable for labeling or parsing sequential data than traditional statistical methods such as HMM (Hidden Markov Model) and MEMM (Maximum Entropy Markov Model) [18]. Thus, it is more suitable for modeling the relations among actions, context values and goals, as can be evidenced from our case study.

Based on our survey, there are three major contributions presented in this paper:

1. The CRF-based method is systematically employed to infer users' goals based on observations of their actions and relevant environmental context values.
2. Three different methods are proposed to explore users' emerging intentions based on the inference results of the goal analysis engine.
3. A software evolution process based on such intention detection approaches is recommended. The methodology is illustrated and validated with an exploratory study of an online service system with a user base of 120 participants.

To substantiate the aforementioned contributions, some of the inference and analysis techniques are partially supported by automated tools. That is, we do not claim a fully automatic system to support software evolution. Manual efforts are still required in some steps, although partial automation owing to taking advantage of such tools (e.g. the CRF-based modeling and computation environment) does help us tackle a genuine human-centric dimension in software evolution.

This paper is organized as follows: Section 2 briefly reviews the literatures on system evolution, requirements elicitation, Conditional Random Fields (CRF) and other statistical modelling methods. Section 3 presents the basic concepts and the methodology built on top of the CRF model based goal inference and intention detection approach. Section 4 introduces a case study on an example online library system by running through each step of the proposed methodology, and summarizes the findings from the study. Section 5 discusses limitations and potential threats to validity. Section 6 concludes the paper and discusses future directions of this line of research.

2 Background And Related Work

Related work in the literature includes work on software evolution, requirements elicitation, and statistical modelling methods.

2.1 Software Evolution

Software evolution refers to the study and management of the process of making changes to software over time [19]. The purpose of software evolution is best characterized per IEEE's definition: *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* [20]. 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 [21]. As a large software system continues to evolve, the complexity of the system will grow, meaning that more effective and efficient evolution methods are much needed [22].

Proposals for change are the drivers for system evolution, and change identification usually continues throughout the system's lifetime. The driving forces of the four maintenance activities as summarized by Lientz and Swanson [23] are hereby paraphrased:

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

Among the above four problems, the incorporation of user's emerging requirements is the core problem for software evolution and maintenance that pertains to our research [5]. Therefore, the capability to accurately and efficiently obtain users' emerging intentions, thus emerging requirements, is a critical issue addressed in this paper.

2.2 Requirements Engineering

The traditional requirements elicitation process can be considered an interactive learning process between the requirements engineer and the customer [24]. The knowledge of users' requirements can be obtained from interview, feedback or observation of customer activities at their workplace [25]. As user requirements are usually implicit and unpredictable [26], this process mainly depends on a requirements engineer performing subjective analysis and using judgment. As such, 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 for a sizable system development [13]. Researchers are now facing the steep challenge of shortening such undesirably long requirements engineering (RE) cycles by applying new technologies and methods that can speed up the process of requirements elicitation and analysis.

In the RE literature, one major related work is the work on goal-oriented requirements engineering (GORE) [27, 28, 29]. The GORE approach uses graphs of goals and intentions to support goal refinement, alternatives exploration, contributions identification, etc. It maps out the functional and non-functional needs of an evolving system, which is complementary to the goal models in general. Goals are the global design objectives of the system designers to be elaborated into formal system specifications and temporal constraints [27]. Non-functional requirements are represented as soft goals to be operationalized in different designs, whose satisfac-

tion dictates the evaluation criteria of given design alternatives [28]. When a multi-agent perspective is undertaken, intentions and commitment of actors in the system-social setting is modeled in order to understand their mutual dependencies and vulnerabilities [29]. Based on the goal-oriented paradigm, there is a common assumption across different approaches, i.e., the subjective goals and intentions of actors remain unchanged, adaptation and evolution mainly take place when there are different ways to operationalize those goals or when there are changes in environmental conditions [30, 31]. Furthermore, run-time monitoring of requirements [32] is to identify violation of the goal conditions, rather than goal changes. The fundamental assumption of this paper is different. Instead of considering goals and objectives as the most stable substance of requirements elicitation, we observe for changes of user goals and intentions.

Another related work is contextual requirements [25]. In current contextual requirements elicitation practices, location, identity, activity and time related information at the workplace of the customer are used to directly capture the service context [25]. In Situ [13], context value changes are consequences of human actions that are externally observable, and the actions performed can be regarded as the external reflection of a human internal mental state, or so-called “desire” which is mostly manifested as system goals in several Situ related case studies and in this paper. Therefore, the primary argument is that it is possible to infer a user’s goal at each observation time-point using an applicable mathematical mechanism through observing a user’s actions and context values [33]. Furthermore, instant requirements can be reasoned through goal resolution of the captured intentions. In Situ, the observation-goal, goal-action relationship may change according to physical context such as time, location of system deployment. Thus, the usage of context is two fold: to model and capture the state of the system run-time settings, and to learn and infer the current and evolving goals of the system actors.

2.3 Statistical Modelling Methods

In order to infer human mental states based on observations, a dynamic mathematical model capable of structured learning should be considered. HMM, a dynamic Bayesian network that provides a framework to predict hidden states from observable variables is considered able to determine human mental states from human actions [14]. Two classes of modeling elements, hidden states and visible states, exist in an HMM. A hidden state in HMM can be readily mapped to a human mental state, and a visible state can be mapped to an observation of human actions and context values. An HMM encodes the relations between hidden states and observations using a probability matrix. The current hidden state can be inferred by an HMM through statistical reasoning based on the previous hidden state and the current observation.

However, HMM is not able to encode the causal relations among actions, context values and goals, nor conquer the complexity of goal transitions, due to: 1) in such an HMM, the current goal is supposed to be independent of neighboring observations. As a result, the relations of a goal and its previous and following observations cannot be reflected. However, in reality, previous context values may influence a user’s current goal, and following actions and context values may be determined by the current goal; 2) in such an HMM, the probabilities of goal transitions are stationary. However, in reality, a user’s goal change usually depends on current context values, which may be quite dynamic. Because of the shortcomings mentioned above, Hidden Markov Models are not effective to accurately infer users’ goals, not to mention performing detection of emerging intentions.

Another class of statistical modelling methods, Conditional Random Fields (CRF), is often used to encode known relationships between observations and construct consistent interpretations [16]. 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 [38], 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).

Besides the applications of labeling or parsing of sequential data, such as natural language text [34], [35] or biological sequences [36], [37], the CRF method and its extensions or variants are widely used in pattern recognition, machine learning and other domains dealing with structured data [18]. CRF has also been applied to intrusion detection [39] and intent understanding from search behaviors of using search engines [40].

CRF is more suitable for our study because the relations among actions, context values and goals can be better reflected in a CRF model. Using feature functions instead of a probability matrix to represent such relations, CRF is able to overcome the shortcomings in HMM because of the flexibility of feature functions. Meanwhile, CRF has been 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) [18]. Thus, CRF turns out to be a very useful technique to support our research objectives.

3 Detection Of Emerging Intention for System Evolution Using CRF

This section presents a method to detect user's emerging intentions for software evolution. First of all, our methodology is mainly applicable to the human-centric context-aware application domains such as 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 features of 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 status. Related work in the literature such as [41] and [42] are very much system and application-specific. On the contrary, our methodology focuses on the human-centric characteristics of a class of systems of interest. By characterizing and formalizing the nature of such target application domains, we aim at investigating the following problems:

1. To infer human goals, what kinds of states, i.e., data, shall be captured? What should be the logical process of inference?

2. Based on observations and inferred goals, how do we capture and formulate possible emerging intentions? And, how do we further elicit emerging requirements to enable software evolution?

3.1 Running Example

An example of system improvement is used in this paper to assist us in explaining the concepts and methods in our methodology. The system is a manuscript management system, called MyReview [43], and it is used by academic conferences for authors to submit and manage their paper manuscripts. On the system, an author can register an account, login the system, submit a manuscript and edit his personal information.

To improve system usability, we are trying to capture users' emerging requirements through observations and analysis of their behaviors. For example, if users often tried to upload files with formats that were not supported by the system, we can come up with an emerging requirement that the system should support more file formats. A more complicated example is, when the user is accessing the system from a far-away country where the connection is slow and unstable, he may simplify his operations, such as skipping opening the page of instructions when submitting a manuscript. A corresponding improvement to the system might be to provide a simplified version of the system interface for users with an unstable connection. Such requirements can be ob-

tained by asking users to give feedback for the system, but a more efficient and sometimes necessary (not in this case but for, say, safety-critical systems) way is to observe and analyze user behavior in real-time (e.g., operation on the interface) and environmental context values (e.g., network speed), and accordingly improve the system. Meanwhile, to reduce human efforts, it is essential to single out meaningful behaviors and information for analysis from a huge set of observation data set. Choosing observation contents for analysis and eliciting emerging requirements from observations are the main concerns of our approach.

3.2 Human-Centric Context-Aware Domain

Our methodology for detection of emerging intention in support of system evolution is generally applicable to human-centric context-aware systems. First of all, three assumptions are made to stipulate the application scope:

Assumption 1. *The application domain is a sensor-laden computer application domain. There are sensors deployed by the system to capture the status of the users, system, and the environment. The inference of predefined goals and detection of emerging intentions will be performed based on the observations of users' actions and environmental context values.*

Assumption 2. *Each user in the domain is believed to be rational, and has desires and corresponding behaviors to achieve certain goals. The observed users' behaviors are individually identifiable, so that the goal inference model can be built for each user based on the observation of his own behaviors and relevant environmental context values.*

Assumption 3. *The user is in only one application of the system in a given time period, i.e., the user is achieving only one goal at a time instant. Meanwhile, the process of goal transition accords with Markov property, i.e., the conditional probability distribution of future goals depends only upon the present goal, not on the sequence of goals that preceded it.*

Assumption 1 can be considered a prerequisite of the application of our methodology. Since actions and environmental context values are used as visible states for inferring invisible states (goals), they have to be observable. In our discussion, the term "sensor" represents all possible monitoring mechanisms. For example, it can be a hardware sensor device, or a monitoring software 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.

According to Assumption 2, the relationships between goal and action can be specified because the causal relationships between users' actions and goals are not always clear and can be inferred. Meanwhile, when there are multiple users in a domain, an inference model can be built for each user because the observed actions of different individuals can be distinguished.

According to Assumption 3, a user has only one goal at an instant, i.e., the user does not have parallel or concurrent goals. Note that in our discussion, user goals should be consistent with system goals, i.e., they should be relevant to the system domain. In addition, the transition of goals should satisfy the Markov property, in order to reduce the complexity of the domain while still keeping the model rich enough for goal inference.

After delimiting the scope of the domains in which our methodology will be applied, the basic concepts are defined as follows:

1. Action: user's behavior exerted in the system's environment, particularly their operations on the system interface.

2. Goal: the condition or status in the system's environment that the user would like to achieve. By recalling Assumption 3, there must be only one goal in any situation.

3. Context: any information that is used to characterize the status of entities in the domain of discourse.

4. Situation: catch-all status of the domain at a time instant. More specifically, a situation characterizes the time-stamped status of the user (actions and goal) and the system (contexts).

5. Intention: an execution path for achieving a certain goal. An intention is represented as a sequence of situations in which all situations are chronologically linked while containing the same goal. The intention sequence converges at situation S_k wherein a corresponding system goal has been satisfied.

In essence, our methodology's highlight is to explore the relation between the visible environmental states and the hidden human internal states. Actions performed can be regarded as an external reflection of a human internal mental state (i.e., desire to achieve a goal), and some context value changes are the results of user actions. Meanwhile, user goals are often influenced by context value changes. Given the above domain characteristics, we can establish certain determinative connections from goal to action, to context value, and then to goal changes. Furthermore, to completely describe the domain, the determinative relations from each goal to each action, from each action to each context value, and from each context value to each goal, all should be specified. However, in practice, user action selections and goal changes may sometimes be random and may not strictly conform to the above characteristics. This is because humans are such complex beings, and oftentimes unpredictable. From a statistical point of view, based on long-time observations and with the help of appropriate mathematical methods, a computational model can be built to reflect the patterns of user action selections and goal changes in commonly seen situation sequences.

3.3 Goal Inference and Detection of Emerging Intentions

As the precondition for detection of emerging intentions, goal inference is to infer a user's goal at each observation time point. According to the attributes introduced in the last section, it should be possible to infer a user's goals based on his/her actions and current context values. To accurately infer user goals, we propose to observe the following states within the observation capability:

1. user actions within the system domain, especially operations on the system interface,

2. those contexts whose values may influence user actions,

3. those contexts whose values can be changed by user actions,

4. those contexts whose values can indicate whether user goal is satisfied or not,

5. other context values that may not be considered of any significance, but can be easily captured. The motivation to include this data is to construct raw data for future analyses, since the data may potentially become valuable and relevant at some point.

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

1. action trigger: each time when the user 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 goal inference and emerging intention detection, as well as related concepts as follows, whereas a step-by-step methodology will be given later in section 3.4:

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. Goal inference and emerging intention detection in the human domain will be performed based on observations. Based on the observation content and frequency discussed above, a number of applicable and necessary sensors should be deployed in the system to capture a user's actions and relevant context values. Some actions and context values cannot or will not be captured due to insufficient observation capability or non-necessity, hence an observation may contain only a part of the status of the system domain.

2. Goal Inference: In goal inference, the inference model takes a sequence of observations as the input and outputs the user's goal at each observation time point. A statistical inference model is used in this process, and feasible candidates can be HMM, CRF, etc.

3. Intention Inference: After the user's goal is inferred for each observation, the intention for achieving a certain goal can be obtained by connecting the situations with the same goal into a sequence that is destined to reach a target state. The underlying rationale is because of our definition of intention in section 3.2.

4. Detection of emerging intention: An emerging intention is a situation-sequence pattern that has not been predefined or captured in the domain knowledge base, i.e., a user's new behavior sequence pattern for achieving a goal. In theory, an emerging intention will arise in two cases: either the user has a new goal, or the user has a new strategy for achieving an existing goal. In practice, emerging intentions will also be detected when the user 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 a user's emerging requirements or system drawbacks due to deviation from the known set of user intentions. Therefore, the ultimate goal of our methodology can be reached, that is, to detect a user's emerging intentions for driving system evolution.

3.4 Emerging Intention Detection Using the CRF Method

The Conditional Random Fields (CRF) method is applied in our research to build the computational model for goal inference and detection of emerging intention. More specifically, we choose linear-chain CRF [38] as the mathematical foundation for our methodology. As the prerequisite of goal inference, a set of feature functions are defined to encode the relations between goal and observable states (user actions and context values). The general format of feature functions in the linear-chain CRF model in this paper is:

$$f(O, i, g_i, g_{i-1}) = \begin{cases} 1 & when\ certain\ relations\ specified\ by \\ & f\ are\ satisfied\ among\ O, g_i \text{ and } g_{i-1} \\ 0 & otherwise \end{cases}$$

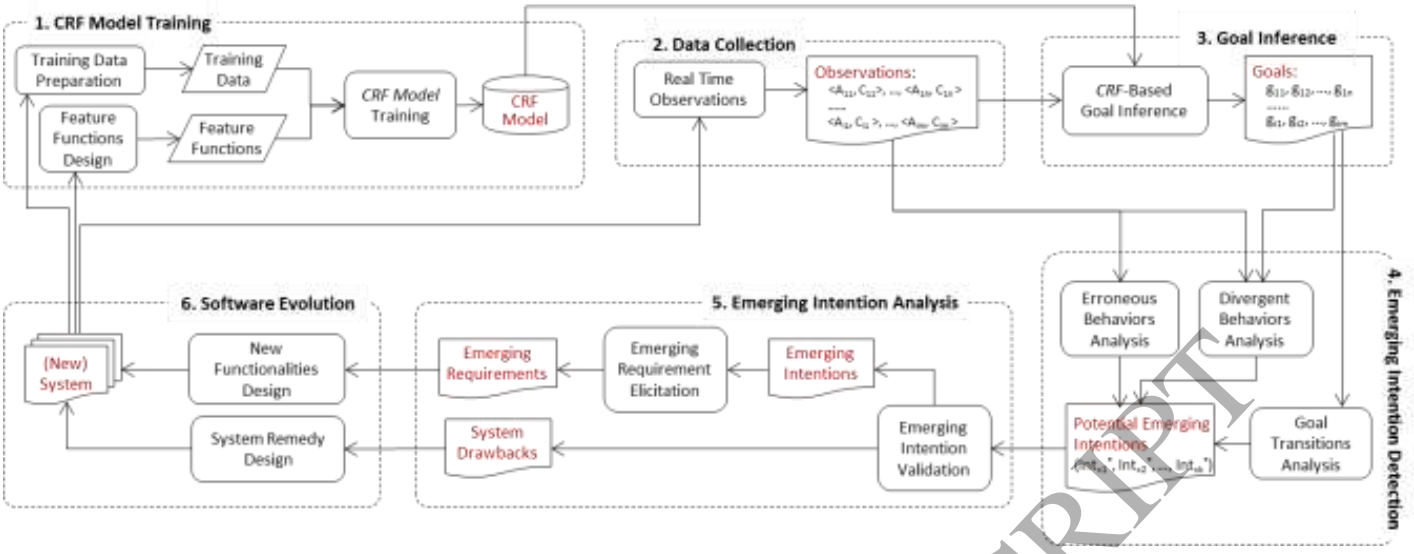

Fig. 1. Workflow of Goal Inference and Emerging Intention Detection for Software Evolution.

in which:

- *O* is a sequence of observations.
- *i* is the ordinal number of current observation in *O.*
- g_i is the inferred goal for the *ith* observation in *O.*
- g_{i-1} is the inferred goal for the *(i-1)th* observation in *O*.
- Each feature function is associated with a weight that indicates its labeling reliability.

Feature functions can be defined to reflect the relations between observations and goals based on domain experts' knowledge. Example feature functions of the MyReview system domain can be found in section 4.4.

The fundamental task of goal inference using CRF is to find the goal sequence *G** with the largest labeling score (highest probability), and use *G** as the inference result for the input observation sequence *O*. To do this, the CRF model calculates the score of every possible goal sequence *G* by adding up the weighted feature functions over all data in the observation sequence:

$$value(G \mid O) = \sum_{j=1}^{u} \sum_{i=1}^{v} w_j f_j(O, i, g_i, g_{i-1}) \qquad (1)$$

(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 *O*.)

Then, these values are transformed into probabilities *p*(*G*|*O*) between *0* and *1* by exponentiation and normalization, and the goal sequence *G* with the largest *p*(*G*|*O*) will be chosen as the inference result for the observation sequence *O*. That is:

$$G^* =< g_1^*, \ldots, g_n^* >= \operatorname{argmax}_G \left(\frac{\exp\left[\sum_{j=1}^{m} \sum_{i=1}^{n} w_j f_j(O, i, g_{i-1}, g_i)\right]}{\sum_{G'} \exp\left[\sum_{j=1}^{m} \sum_{i=1}^{n} w_j f_j(O, i, g'_{i-1}, g'_i)\right]} \right) \quad (2)$$

To reduce the computation complexity, the Viterbi Algorithm is applied in computing *value*(*G*|*O*), and the Limited-memory BFGS is a common algorithm used for estimating the weights of feature functions in CRF model training.

The complete process of applying our methodology to detect emerging intentions for software evolution is depicted in Fig. 1. The detailed description of the task of each step is:

Step 1: Constructing the CRF model. In order to accurately infer the user's goals predefined in the domain knowledge base and to detect the user's emerging or unexpected intentions, a linear-chain CRF model that encodes users' known behavior patterns for achieving goals should be built as the metrics for outlier detection [44]. We propose to use supervised learning [45] 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 sequences shall follow the form of $<o_1^*, g_1^*>, <o_2^*, g_2^*>, \ldots, <o_m^*, g_m^*>$, in which each observation o_i^* is labeled with a goal g_i^*. Since the CRF model is used as a standard reference model to infer goals, it must be built upon existing domain knowledge, including known or pre-defined goals, 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 goals associated with each observation $<g_1^*, g_2^*, \ldots, g_m^*>$ shall be accurately reported by users 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 goals. Feature templates specifying the form of feature functions are 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.

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

Step 3: Goal Inference*:* Input the observation sequences prepared in Step 2 into the CRF model acquired in Step 1, and infer the user's goal at each observation time point.

Step 4: Detection of Emerging Intention*:* As introduced in section 3.3, an emerging intention is a situation-sequence pattern that has not been predefined in the domain knowledge base. In our methodology, the formulation of emerging requirements is performed based on the results of emerging intention detection. Considering the various causes of emerging intentions, and for the purpose of system improvement, three semi-automatic methods for detecting emerging intentions based on the results of goal inference is proposed.

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

Detect users' divergent behaviors through goals 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 misunderstanding of required operational procedures, dissatisfaction with the system, or most importantly, expression of emerging desires. For example, in the MyReview system, a user might think the submission 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 behavior may appear irregular and cannot be well interpreted by the CRF model, which will result in low confidence when labeling goals. Therefore, these divergent behaviors can be singled out for analyzing users' emerging requirements.

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

Obtain an emerging intention based on goal transitions. If two goals often appear consecutively, an emerging intention can be considered to make the goal transition smoother. Accordingly, the system can be modified to simplify user operations. For example, again in the MyReview system, if users often move on to check the submitted paper right after finishing the submission of a manuscript, a link that directs the users to the paper information page can be added on the confirmation page of "*submitting a paper.*" This kind of goal transition can be obtained based on the results of goal inference with a high confidence level.

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

Find emerging desires and system drawbacks from users' erroneous behaviors. When users have emerging desires that are not supported by the current system, their behaviors may trigger error reports. For example, in the same MyReview system, if a user tries to upload a file in a format that is not supported by the system, an error report will be generated. If such an error report occurs time and time again, it may indicate that many users actually want to upload a file in a non-supported format, which can become an emerging 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." 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.

The intentions and issues inferred above will be further analyzed by domain experts to determine whether they are indeed emerging intentions or system drawbacks. This step cannot be done automatically, and must be manually performed by domain experts. Therefore, our methodology is used as a filter to reduce the human expert's input space in emerging requirements elicitation. The system drawbacks may not be too difficult to identify, while the emerging intentions are usually implicit and hard to decide. Domain experts can make some assumptions of the emerging intentions and then verify them based on their expertise, or they can directly ask the users for evaluation if necessary. After this validation and verification process is done, the current system shall be redesigned to satisfy users' emerging intentions, or be retrofitted with necessary remedies to overcome those revealed drawbacks or defects.

4 Case Study On A Research Library System

As stated in the previous section, our methodology can support discovering emerging requirements and hopefully shortening the software evolution cycle. This section presents an exploratory case study on a dynamic web-based system to demonstrate and validate our methodology. It is "exploratory" because our methodology takes a unique approach to goal inference via applying CRF to this research domain.

First of all, the following research questions are asked before conducting the case study:

1. Based on observations of user actions and relevant context values, is it possible to accurately infer user's goals using a CRF model? What is the inference accuracy?

2. Is it possible to detect emerging intentions based on the results of goal inference using the three methods introduced in section 3.4?

3. Is it possible to reveal user's emerging requirements or system drawbacks based on the detected emerging intentions? Is it possible to achieve rapid system evolution based on the revealed emerging user requirements or system drawbacks?

To answer the above questions, our case study can be divided into three sub-studies as follows:

1. First-round case study (goal inference study): a study to assess the performance of goal inference by using CRF

2. Detection of emerging intention study: a study to demonstrate the effectiveness of our methodology for emerging intention detection and emerging requirements elicitation

3. Second-round case study (system evolution study): a study to validate the system improvement between two versions of the system.

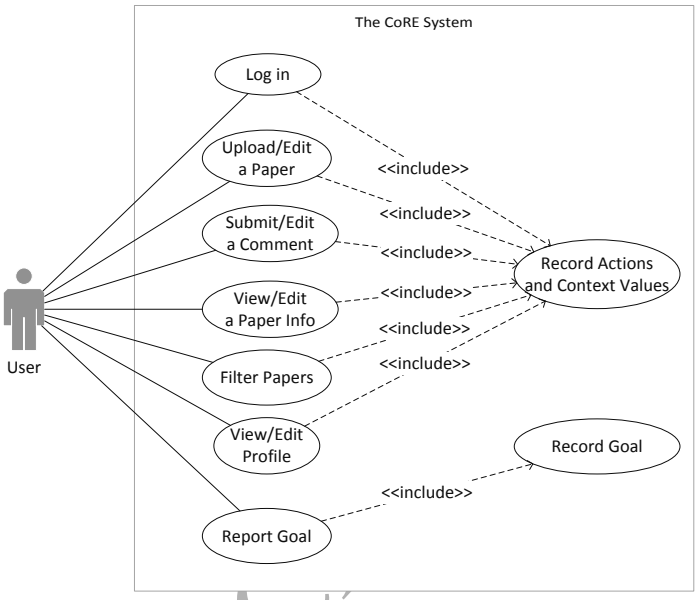

Fig. 2. The use case diagram of the *CoRE* system.

Fig. 3. Interface of the page for uploading a paper in *CoRE* Version I.

4.1 Study Platform – the CoRE System

The experimental system is an online library system, called Cooperative Research Environment (CoRE). It was modified from an open-source web application, MyReview [43], 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 case study is expected to emulate users' operations on a real-world system. Another reason for choosing a web-based system to conduct our case study is that it is easier to get participants' self-reported goals in a "think aloud" fashion, which is 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. 2 is the use case diagram of the CoRE system.

To monitor user behaviors and capture related context values, an embedded program is deployed in CoRE as a sensor. User operations, paper and comment submissions, and the content of the web pages will be recorded and stored in the database. During each session, users report/select their current goals from a dropdown list containing a set of expected goals. Examples of goal options are "*Upload a paper,*" "*Submit a comment,*" "*View paper information,*" etc., and "*Not in the list.*" To ensure the selections are correct, users can also correct their goal selections in the post-session questionnaire in case that they forgot to report goals or reported an incor-

rect one during the study. Fig. 3 shows the interface of the page for uploading a paper in the CoRE system, on which there is a dropdown menu in the right-side panel for users to report their goals during the study.

4.2 Procedure of Case Study

A group of 120 people participated in our study. The participants were randomly selected from people who responded to our invitation with willingness for participation. Among the 120 participants, 44 are PhD students, 63 are Masters students, and 13 are college teachers (lectures or assistant/associate/full professors). 31 participants are female and 89 participants are male. According to their education level, all participants are supposed to be familiar with operations on a web-based system, and most of them have experience of using a paper submission system.

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 goals, and relevant context values were recorded as the raw data. In order to show our methodology's ability to enable and speed up the evolution process of the CoRE system, the case study was done over two rounds (as two sub-studies) 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 study for 30 days: invite participants, collect data on participants' actions, goals and context values.
3. Shut down CoRE Version I. Apply our proposed methodology on the raw data captured in Round 1, and use CRF to infer participants' goals. Then, carry out an Emerging-Intention-Detection case study, analyze and elicit users' emerging requirements, and further revise the system accordingly.
4. Deploy the enhanced system: CoRE Version II.
5. Run the second-round study for 30 days: invite some new participants, collect data records of participants' actions, goals and context values.
6. Shut down CoRE Version II. Apply our proposed methodology on the raw data captured in Round 2; use CRF to infer participants' goals.
7. Evaluate the effectiveness of our methodology on the evolution of CoRE from Version I to II.

During each sub-study, 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 study website and log in to study
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 goal on each webpage by choosing a predefined goal 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 goals if necessary.

Observation in our study was action triggered. During each 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.

Each raw data record has the following attributes:

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 goal

4.3 Data Collection and Preprocessing

This step corresponds to Step 2 (Data Collection and Pre-processing) in our methodology (in section 3.4). In our case study, Step 2 and Step 1 (constructing the CRF model) 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 sessions captured in the first-round case study, and 10,524 raw data records and 582 sessions captured in the second-round case study. A raw data record contains all the six attributes presented in the previous section. A record of a session is the data sequence that starts when the user logs into the study, and ends when the user logs out.

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

1. If most (>50%) self-reported goals are "Not in the list", which is the default value in the goal selection dropdown list. We assume that in this case the participant forgot to report his goal at all or most of the time in the study.
2. If the answer is "no" for the question "Did you select the goal every time when you had a new goal?" 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 goals were reasonable or not. Those obviously wrong ones were removed because they cannot be used to validate our goal inference results. An example is that a participant might have forgotten to report his goal (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 case study had 6880 data records and 369 sessions, and the final data set of the second-round case study had 6931 data records and 361 sessions.

4.4 Building the CRF Model

According to Step 1 (Constructing the CRF model) in section 3.4, 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 study, the training data was selected based on the following criteria:

1. The records belonged to system designers and those experienced participants because their behaviors were to achieve certain expected goals. By "expected", it means that those behaviors should 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 goals.
3. The self-reported goals recorded in the training data set were accurate.

TABLE 1 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]	

TABLE 2 EXAMPLE SEGMENT OF TRAINING DATA

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

TABLE 3 EXAMPLE SEGMENT IN THE TRAINING DATA SET

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

After being processed from the raw data, a training data set was attained for building the CRF model. There were 2930 data records and 158 sessions in the training data set taken from the first-round case study. Each training data record had three elements: observation, time interval and goal.

- Observation: includes the action and context values 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. In our experiment, the time intervals are partitioned into several levels. For example, *5s* represents time intervals less than 5 seconds, *30s* represents those between 20 seconds and 30 seconds, and *m* represents those between 60 seconds and 60 minutes, etc.
- Goal: user's self-reported goal

An open-source tool, CRF++ [46], was used to build our CRF model based on the training data. According to Step 1.2 (Defining Feature Functions) in 3.3, the feature functions for our CRF model is to be designed, and in CRF++, they can be automatically generated based on feature templates that specify their formats. The feature templates used in our study are shown in TABLE 1, 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 goal. *%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]* is:

$$f_1(O, n, g_{n-1}, g_n) = \begin{cases} 1 & if\ g_n = \text{ViewAPaperInfo} \\ & \quad O_n = \text{clickPaperInfo\&PaperID} \\ & \quad O_{n-1} = \text{clickMenuAllPapers} \\ 0 & Otherwise \end{cases}$$

The above feature function can be interpreted as: if the current observation O_n is *click the link "View Paper Info" of a paper* and its previous observation O_{n-1} is *click the menu option "All Papers"*, the current goal g_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, the accuracy of goal inference can be improved by considering the neighboring observations when designing feature templates. For example, there is a segment in the training data set as shown in TABLE 2, in which observation *clickLogin&LoginGood* is labeled as *EditProfile* because the following observations show that the user updated his profile. This kind of relationship between observations and goals shown in the above example can be encoded into the CRF model based on a template which takes the current, next and after next observations into account for inferring the current goal.

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

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

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

Due to the nature of the domain of CoRE, the inference results can be made more accurate by adding the factor of time intervals into feature templates. TABLE 3 shows two example segments in the training data set (separated by dotted lines).

The first segment in TABLE 3 shows that if the user's goal 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 goals 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 goal.

In summary, each feature template shown in TABLE 1 has its own meaning in describing certain properties of the domain of CoRE. During the course of the case study, 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 Goal Inference Result Analysis in the First-Round Case Study

This step corresponds to Step 3 (Goal Inference) in our methodology (in section 3.4). Separated from training data, the rest of the data records were used as test data in the first-round case study, containing 3,950 data records and 211 sessions. The format of the test data was the same as that of the training data. Based on the built CRF model (introduced in section 4.4), goal inference was performed on the test data using CRF++. TABLE 4 shows a sample of the results of goal inference.

TABLE 4 GOAL INFERENCE RESULT USING CRF

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

TABLE 5 INFERENCE ACCURACY OF DIFFERENT TEMPLATES

	Template 1 U01, B01~B04	Template 2 U01~U06	Template 3 U01~U06, B01~B04	HMM
%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 sessions*

Each inference result contains the inferred goal and the inference probability. When using the same training data and test data but different feature templates, the inference accuracy was different. Meanwhile, to show a comparison using different modeling methods, a HMM-based inference study is conducted using the same set of training data and test data that used to build the CRF model. TABLE 5 lists the inference results of using HMM and CRF models with different types of feature templates.

The results in TABLE 5 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 emerging intentions (introduced in section 3.4), it was necessary to choose data records that had high probability to indicate such behaviors in an effective way. In our methodology, the observation records corresponding to users' divergent behaviors will most likely be labeled with goals which are not consistent with their real goals because the CRF model does not explain these behaviors very well. However, in practice, since users do not report their real goals in such a case, it is not feasible to detect users' divergent behaviors through comparing the inferred goals with the self-reported ones. An alternative is to look into the inferred goals with low inference probabilities, as we anticipate that inference

TABLE 6 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*

accuracy will progressively decrease/increase consistent with inference probability. Based on our analysis on data records, 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 6, the mislabeling rate (%Mislabeling) is higher for those observations with lower inference probability. In practice, since it is impossible to study all data records, we focused on analyzing observations with relatively low inference probability first. For example, we studied data records with an inference probability lower than 0.5 in our experiment.

TABLE 7 EXAMPLE OF GOAL INFERENCE RESULTS

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

4.6 Emerging-Intention-Detection Case Study

This step corresponds to Step 4 (Emerging Intention Detection) in our methodology (in section 3.4). Three semi-automated emerging intention detection methods were applied and demonstrated with illustrative examples based on the first-round case study results shown below.

1. Method I: Analysis of divergent behavior

We study 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. Emerging_Intention 1: some users often click button *hide the above selection form* to hide the selection form (see Fig. 4) immediately after entering the page *All Papers*.

As shown in TABLE 7, the observation *clickHideSelection* is labeled with *ViewAllPapers* as the inferred goal with a low probability *0.401886*, and its previous observation is also labeled with a low-probability goal, while its following observation is labeled with a very high-probability goal. 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.

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. 4).

b. System_Drawback 1: As shown in TABLE 8, 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 selections 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 selections when they submit a comment.

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

c. System_Drawback 2: Click the button *Submit* twice or more on the page *submit/edit a comment*: The 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. 5).

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 should contain no spaces.

TABLE 8 EXAMPLE OF THE GOAL 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

TABLE 9 EXAMPLE GOAL TRANSITIONS

Goal 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 goal transition from "UploadPaper" to "ViewMyPaperInfo"*

[b] *The goal transition from "UploadPaper" to any other goal*

2. Method II: Analysis of goal transition

Another way to find users' potential emerging intentions is to study the goal transitions. TABLE 9 shows some examples of goal transitions in the first-round case study.

Emerging intentions can be defined for frequently occurring goal transitions to make them smoother, and the system can be accordingly modified to simplify users' operations. For example, for goal transitions *UploadPaper → ViewMyPaperInfo* (Emerging_Intention 2) and *SubmitComment → ViewMyCommentInfo* (Emerging_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. 5). For the goal transition *ViewAPaperInfo → DownloadPaper* (Emerging_Intention 4), a link can be added on the page of paper information to allow users to directly download the paper (see Fig. 6).

3. 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' goals. In this case, CRF is useful to exclude such noise in the data, and the aimless behaviors are usually labeled with goal "test" by the CRF model, which encodes many "test" behaviors that will be ignored for emerging intention detection. After removing the noise in the data, the occurrences of users' erroneous behaviors can be simply counted and examined.

Fig. 4. The improvement of selection form (upper: selection form in *CoRE* Version I, lower: selection form in *CoRE* Version II).

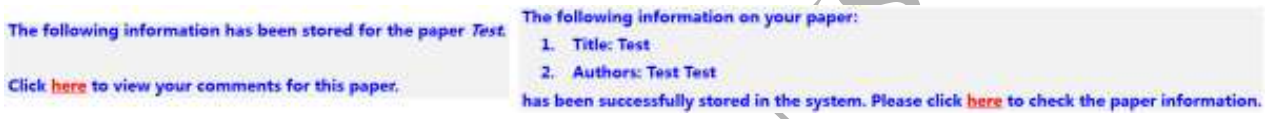

Fig. 5. The new link for viewing the submitted comment and for viewing the uploaded paper.

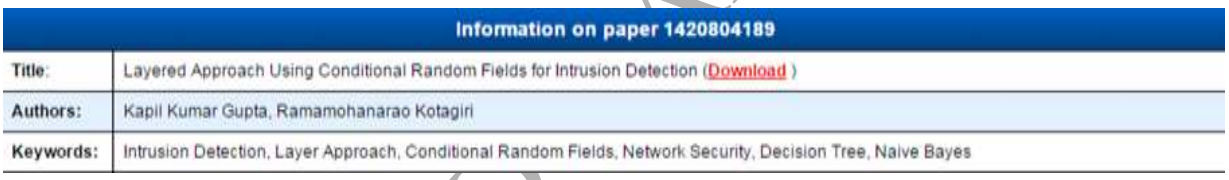

Fig. 6. The new link for downloading a paper.

Two examples of common user erroneous behaviors detected in the first-round case study were:

a. Emerging_Intention 5: No file was uploaded when editing the information of a paper. Such erroneous behavior occurred probably because users did not want to upload a file when editing the paper information, which can be considered an emerging 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 had no idea how many words they had typed in. One possible improvement could be that the system shall always display the number of words left to meet the minimum limit.

TABLE 10 OVERALL GOAL 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% (351/3938)	0.848429	0.28694

TABLE 11 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	6/9	66.67%

Based on the emerging 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 Version II.

4.7 Validation of System Improvement in the Second-Round Case Study

To demonstrate and validate the potential improvement of the system, a second-round case study was done on the evolved system – Core Version II, by following exactly the same process as what we did in the first round. The only difference was that new participants were recruited in the second round to make the comparison between two versions fair and even. This section will evaluate the improvement of CoRE version II over its predecessor.

Similar to the data analysis in the first-round case study, a CRF model was built based on *2,984* training data records, and it was used for goal inference on a test data set with *3,947* records. The results of goal inference in two rounds are shown in TABLE 10, from which we can see that the overall inference probability and inference accuracy in the second-round case study were significantly improved compared with those results in the first round. The reason might be that the new system is more user-friendly so that user behaviors conformed to the designed use case scenarios.

Some example system improvements are described below:

1. Simplified selection form (Emerging_Intention 1, Fig. 4)

To evaluate the benefit of the simplified selection form in the new system, we did a statistical analysis of user 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, the conclusion 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 (Emerging_Intention 2&3, Fig. 5).

TABLE 11 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 demonstrates the effectiveness of the modification.

3. *Download* link on *paperinfo* page (Emerging_Intention 4, Fig. 6)

The observed user behaviors in the second-round case study show that users preferred 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 case studies. The comparison result is shown in TABLE 12. As we can see, the success rate of four tasks all increased in the second-round case study, while the time cost and the error occurrence rate, especially the consecutive error occurrence rate, significantly decreased.

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

4.8 Summary of the Case Study

Based on the above demonstration of the execution of the case study and the analysis of the results, we can now revisit those three research questions proposed at the beginning of this section for answers.

1. Goal inference with CRF model:

According to inference accuracy results of the first-round case study (TABLE 5, section 4.5), CRF model was able to deliver highly accurate goal inference results when appropriate feature templates were designed and applied. The actual inference accuracy rate was 91%, and 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 labeling. As a comparison, 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 emerging intention & system drawback:

According to our case study in section 4.6, several emerging intentions and system drawbacks were identified for CoRE Version I using the three newly proposed methods (in section 3.4). As in CoRE Version II, our methods were shown to be effective in detecting emerging intentions in the domain of CoRE. For example, a phenomenon we discovered through goal transition analysis (Method II) is that the user often reviews or downloads papers one by one after filtering papers using the selection form. Since these consecutive *Download Paper* goals occur frequently, we proposed to introduce a compound goal called *Download All Papers* by adding a link for downloading all selected papers at once.

The above results and examples also demonstrate that our methodology is able to continuously explore users' emerging intentions and enhance the system to adapt to volatile user requirements.

TABLE 12 COMPARISON BETWEEN TWO ROUNDS CASE STUDIES

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*

1. Successful and rapid system evolution:

Based on our analysis results in section 4.7, the usability of CoRE was apparently improved in some aspects. Our experience shows that the system successfully evolved from version I to version II. To evaluate the efficiency of our methodology, we give a detailed explanation of the level of manual effort and expertise in each step in the case study.

(1) Training Data Collection: This step is to collect behavioral data and context data to train the CRF model. In each round of our case study, we collected behavioral and contextual information from *60* users for one month, and got around *10,000* raw data records. The collection time can be shortened in practice because we only need to collect training data from observations of domain experts (e.g., system designers and experienced users) who are familiar with the system, while in our study we also collected test data from behaviors of participants who were sometimes reluctant to perform tasks on the system.

(2) Defining Feature Functions: This step should be done by domain experts. In our study, we spent only one hour to define and refine feature templates based on analysis of the relations among users' goals, actions and context values. Conceivably, domain experts not familiar with this task will take a longer time to start.

(3) CRF Model Training: This step was done in just seconds with the help of CRF tools.

(4) Data Collection and Pre-processing: This step is to obtain users' operation data in order to analyze and uncover their emerging requirements. In our study, we collected this data in one month based on observations of 60 users' behaviors. The data collection process can be more effective in practice, especially for those systems or applications with a large user base, such as social network operators.

(5) Goal Inference: This step was done in just seconds with the help of CRF tools.

(6) Detection of Emerging Intention & Analysis of Emerging Requirements: This step was done by domain experts in 2 days based on the results of goal inference.

(7) System Improvement: It took about a half of a month for one domain expert to evolve our experiment system based on the seven emerging intentions and five system drawbacks. As we show in section 4.6, 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 emerging requirements and system remedies.

We speculate that our methodology could be more efficient in practical applications because users' operation data can often be collected more quickly. 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.

5 Threats to Validity

This section discusses some potential threats to the validity of our proposed methodology and case study 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

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

1. Design of the experimental system

We designed and developed the CoRE system to carry out our case study to validate our proposed methodology, and conducted a considerable amount of manual work in the process of data pre-processing and emerging requirements analysis. However, we had no control of users' behaviors throughout the experiment, and the method of data pre-processing is rather mechanical and explicit as stated in section 4.3. The design of the feature functions in our CRFs model might appear burdensome, however, it is common to fine-tune the statistic models in any data mining/learning study. In short, we had no control over the trends and characteristics shown by our data. And, the aforementioned mentioned process can be automated in a computable way, which is one of the major technical merits of our methodology. Also, there were a few unexpected emerging requirements coming out of our analysis. For example, System_Drawback 3 (users consecutively input wrong passwords) and Emerging_Intention 5 (no file was uploaded when editing the information of a paper).

2. Interaction of normal operations and reporting goals

To directly validate our goal inference results, we asked participants to report their goals in real-time while operating on the system, which is considered a necessary part of our case study. However, imposing additional tasks (reporting goals) might have interfered with participants' normal thinking process and might have further influenced their understanding of the system. An example would be that the dropdown goal list on each webpage of CoRE contained all the predefined goals related to this application, and participants may have been able to learn from it, and consequently adjusted their behaviors. Although such phenomenon is undesired, the value of acquiring participants' self-reported goals significantly outweighs its adverse effect.

3. 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 explaining our experiment goals and technical methods, participants might have been able to guess what hypothesis 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 were not much concerned with the possible impact of such a 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. They could finish most of the tasks without thinking too much about the system and the experiment itself. Therefore, the impact of hypothesis guessing, acknowledged as a potential threat, should be minimal in our study.

5.2 Threats to Internal Validity

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

1. Prediction of emerging requirements

The emerging requirements revealed from the analysis of divergent behaviors were mainly based on our subjective judgment, and were validated through usability analysis between two versions of systems indirectly, instead of directly inquiring to the users. Therefore, it is still debatable whether our predicted emerging requirements were in fact the users' emerging requirements, or just the 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 emerging requirements. For example, the reason that time spent for uploading a paper (TABLE 12) decreased significantly may be because some previously required information was removed, such as DOI. Still, we can draw a reasonable conclusion that as CoRE evolved, its overall system usability was improved by implementing emerging requirements and necessary remedies using the semi-automated approach as described above.

2. Arbitrary design of goal granularity

To build the domain knowledge, a prerequisite was to define and enumerate as many anticipated goals as possible. The quality of the predefined goal set is crucial for the accuracy of goal inference. The granularity of goal must be properly designed. If the goal 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 goal is too coarse-grained, it may not be able to generalize the causal relations among goal, actions, and context values. In our case study, the goals 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 goals.

3. Statistical conclusion validity

As statistical methods were used to identify candidate observations for emerging intention detection, it basically depended on the quality and characteristics of sample data (mislabeled records). When the sample space is larger, 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 emerging 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 emerging requirements of interest.

5.3 Threats to External Validity

The intended application domain is a human-centric context-aware domain. As we stated in Assumptions 1, 2 and 3, the domain should be context aware, and operated by a single user who has a single goal at any instant. If an application domain does not present such attributes, our methodology requires further studies to be feasible. However, we believe that our methodology has broad application prospects because many applications can be characterized as human-centric context-aware domains as described in section 3.2. 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 [47].

3. Home automation [48] is another promising application area, especially for independent living in smart homes for elderly and disabled [49], in which most user behaviors can be monitored and there is increasing user demand to increase the quality of life without attention of caregivers or institutional care.

As users' desires and intentions for using such applications may change and evolve often, our methodology is meant to speed up the evolution cycle so that users' changing and emerging needs can be better and more rapidly satisfied.

6 Conclusions

This paper presents a methodology for applying Conditional Random Fields as the mathematical foundation to infer human goals based on observations of their actions and relevant environmental context values, and further explore users' emerging intentions for driving system evolution. Through this particular exploratory case study, we show that the accuracy of goal inference is high (> 90%). Furthermore, the three methods for detecting emerging intentions (section 3.4) are shown to be effective to obtain users' potentially emerging intentions and further reveal their emerging requirements on the system or draw attention to system drawbacks that are useful for system evolution. The study results verified our expectations about the rapid discovery of emerging requirements through automated inference, and evolved the system based on our methodology. In sum, our

methodology has been shown to be able to improve existing functionalities on the basis of an existing system, and it is particularly suitable for eliciting users' functional requirements and improving the usability of a system.

However, there are some limitations of our work. Besides the application scope delimited by Assumptions 1, 2 and 3 (section 3.2), our methodology is only able to capture users' emerging functional requirements, not non-functional requirements such as high usability, fast response time, etc. In addition, the analysis of potential emerging intentions and users' emerging requirements still require a lot of human effort and may not consistently give the best results. Therefore, the gap between goal inference and emerging 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 emerging intentions and system drawbacks. The work of eliciting and verifying emerging intentions and system drawbacks, as well as designing and implementing new system functionalities would still require human intervention. From a knowledge engineering perspective, further investigation is plausible as to how to acquire design wisdom from raw-data analytics.

2. According to the rationale of goal inference, our methodology might not perform well in systems where users frequently interact with each other. For example, if users interact frequently, and their actions directly influence each other's goal, their behaviors will be very hard to describe and individualized goal inference cannot be performed using our methodology. To extend the applicability of our methodology, there are some interesting research questions to be answered. For example, "how can we characterize the causal relationships among multiple users' goals, actions, and context values?" and "can CRF encode such causal relationships catered to other research questions?", etc.

3. In our case study, 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 to other types of systems, more experimental validation is needed. Some valuable candidates are mobile applications, smart home systems, etc., which are becoming prevalent nowadays with rapidly changing user requirements.

Acknowledgments

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 case study work. We would also like to acknowledge the guidance and support from the IRB Committee at Iowa State University.

References

[1] V. Rajlich, "Software evolution and maintenance," *Proc. the on Future of Software Engineering*, pp. 133-144, 2014.

[2] E.B. Charrada, A. Koziolek, and M. Glinz, "Supporting requirements update during software evolution," *Journal of Software: Evolution and Process*, vol. 27, no. 3, pp. 166-194, March 2015.

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

[4] 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.

[5] H.P. Breivold, I. Crnkovic, and M. Larsson, "Software architecture evolution through evolvability analysis," *Journal of Systems and Software,* vol. 85, no. 11, pp. 2574–2592, November 2012.

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

[7] N. Seyff, G. Ollmann, and M. Bortenschlager, "AppEcho: a user-driven, in situ feedback approach for mobile platforms and applications," *Proc. the 1st International Conference on Mobile Software Engineering and Systems*, pp. 99-108, 2014.

[8] 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.

[9] 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.

[10] 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.

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

[12] 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.

[13] 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.

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

[15] 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*, pp. 71-76, 2015.

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

[17] C.L. Baker, R.R. Saxe, and J.B. Tenenbaum, "Bayesian Theory of Mind: Modeling Joint Belief-Desire Attribution", *Proc. of the thirty–third annual conference of the cognitive science society*, pp. 2469–2474, 2011.

[18] 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.

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

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

[21] P.I. Okwu and I.N. Onyeje, "Software Evolution: Past, Present and Future," *American Journal of Engineering Research (AJER)*, vol. 03, no. 05, pp. 21-28, 2014.

[22] Liguo Yu and Alok Mishra, "An Empirical Study of Lehman's Law on Software Quality Evolution," *International Journal of Software and Informatics*, vol. 7, no. 3, pp. 469-481, 2013.

[23] 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.

[24] R. Rowel and K. Alfeche, *Requirements Engineering: A good practice guide*, John Wiley and Sons, 1997.

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

[26] G.E. Kniesel and R.E. Filman, "Unanticipated Software Evolution," *Journal of Software Maintenance and Evolution: Research and Practice*, vol. 17, no. 5, pp. 307–377, 2005.

[27] A. Dardenne, A. van Lamsweerde, and S. Fickas, "KAOS: Goal-directed requirements acquisition," *Science of Computer Programming*, vol. 20, no. 1-2, pp. 3-50, 1993.

[28] L. Chung, B.A. Nixon, E. Yu, and J. Mylopoulos, "NFR: Non-Functional Requirements in Software Engineering," *International Series in Software Engineering*, vol. 5, 2000.

[29] E.S.K. Yu, "Towards modelling and reasoning support for early-phase requirements engineering," *Proc. of the 3rd IEEE International Symposium on Requirements Engineering*, pp. 226-235, 1997.

[30] Vítor E. Silva Souza, Alexei Lapouchnian, Konstantinos Angelopoulos, and John Mylopoulos, "Requirements-driven software evolution," *Computer Science - Research and Development*, vol. 28, no. 4, pp. 311-329, 2013.

[31] Fabiano Dalpiaz, Paolo Giorgini, and John Mylopoulos, "Adaptive socio-technical systems: a requirements-based approach," *Requirements Engineering*, vol. 18, no. 1, pp. 1-24.

[32] Yiqiao Wang, Sheila A. McIlraith, Yijun Yu, and John Mylopoulos, "Monitoring and diagnosing software requirements," *Automated Software Engineering*, vol. 16, no. 1, pp. 3-35, 2009.

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

[34] 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)*, vol. 4, pp. 188-191, 2003.

[35] 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.

[36] 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.

[37] K. Sato and Y. Sakakibara, "RNA secondary structural alignment with conditional random fields," *Bioinformatics,* vol. 21, no. 2, pp. 237–242, 2005.

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

[39] 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, no. 1, pp. 35 – 49, 2010.

[40] 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.

[41] 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.

[42] 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.

[43] P. Rigaux, "The MyReview System," Retrieved from http://myreview.sourceforge.net, 2016.

[44] V.J. Hodge and J. Austin, "A Survey of Outlier Detection Methodologies," *Artificial Intelligence Review*, vol. 22, no. 2, pp. 85-126, 2004.

[45] 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.

[46] T. Kudo, "CRF++: Yet Another CRF toolkit," Retrieved from http://taku910.github.io/crfpp, 2014.

[47] 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.

[48] 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.

[49] 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.

Authors' Biography:

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 served as editor-in-chief for IEEE Software from 1991-1994, and editor-in-chief for Computer from 2007-2010. Chang is a Fellow of the IEEE, a Fellow of AAAS, and a member of the European Academy of Sciences. His research interests include software engineering, services computing and successful aging.

Lin Liu is Associate Professor at the School of Software, Tsinghua University, China. She received her Ph.D. in Computer Science from the Institute of Mathematics, Chinese Academy of Sciences in 1999. Her interests are in the areas of requirements engineering, services engineering, and data and knowledge engineering. Her research emphasizes concepts and techniques for modelling and analyzing the social and human perspectives of software systems.