2017

# A situation-centric, knowledge-driven requirements elicitation approach

Jingwei Yang
*Iowa State University*

**A situation-centric, knowledge-driven requirements elicitation approach**

by

**Jingwei Yang**

A dissertation submitted to the graduate faculty

in partial fulfillment of the requirements for the degree of

DOCTOR OF PHILOSOPHY

Major: Computer Science

Program of Study Committee:
Carl K. Chang, Major Professor
Samik Basu
James A. Davis
Simanta Mitra
Johnny S. Wong

Iowa State University

Ames, Iowa

2017

DEDICATION

To

My dear Mom and Dad, Xiuyan and Hai

My dear wife, Yanni

My dear son, Jason

TABLE OF CONTENTS

# NOMENCLATURE

| | |
|---|---|
| RE | Requirements Engineering |
| KB | Knowledge Base |
| DIKW | Data-Information-Knowledge-Wisdom |
| MTL | Multi-strategy, Task-adaptive Learning |
| CRF | Conditional Random Fields |
| CoRE | Cooperative Research Environment |

# ACKNOWLEDGMENTS

I would like to thank my advisor, Prof. Carl K. Chang, and my committee members, Drs. Basu, Davis, Mitra, and Wong, for their guidance and support throughout the course of this research.

In addition, I would also like to thank my friends, colleagues, the department faculty and staff for making my time at Iowa State University a wonderful experience. I want to also offer my appreciation to those who were willing to participate in my experiment, without whom, this thesis would not have been possible.

# ABSTRACT

Human factors have been increasingly recognized as one of the major driving forces of requirement changes. We believe that the requirements elicitation (RE) process should largely embrace human-centered perspectives, and this work focuses on changing human intentions and desires over time. To support software evolution due to requirement changes, *Situ* framework has been proposed to model and detect human intentions by inferring their desires through monitoring environmental contexts and human behavioral contexts prior to or after system deployment. Earlier work on *Situ* reported that the technique is able to infer users' desires with a certain degree of accuracy using the Conditional Random Fields method. However, new intention identification and new requirements elicitation still primarily depends on manual analysis.

This work attempts to find a computable way to identify users' new intentions with limited help from human oracle. We discuss the feasibility of implementing the concept of Data-Information-Knowledge-Wisdom (DIKW) to bridge the gap between requirements and data pertaining to user behaviors and environmental contexts, and propose a situation-centric, knowledge-driven requirements elicitation approach using the Multi-strategy, Task-adaptive Learning (MTL) method and the Strategic Rationale (SR) model. A case study shows that the proposed approach is able to identify users' new intentions, and is especially effective to capture alternatives of low-level tasks. We also demonstrate how these newly identified intentions can be fused to the existing domain knowledge network using the SR model, and harvest high-level wisdom, in terms of new requirements and design insights.

*Keywords* – contexts, DIKW, human intention, requirements, situation, SR model, the MTL method, user behaviors

# CHAPTER I

# INTRODUCTION

Requirements elicitation (RE) is an essential phase in system development because the quality of the elicitation results will be directly reflected in the final product. In order to properly manage this phase, various methods have been studied and practiced [1]. However, system specification still cannot describe the system-to-be completely and correctly. One major problem is how to handle changes in user's needs and environmental contexts. Unfortunately, most of the traditional RE techniques fall short in handling changes in terms of effectiveness and efficiency [2]. In recent years, along with the outburst of the development and application of different data analytic techniques, requirements engineering researchers started looking into the possibility of acquiring requirements and subsequent changes from different data sources, such as historical multi-modal user behaviors, system log information, error reports, etc. Although some promising results have been shown, attempts to systematically solve the problem have not been successful.

To handle changes, the strategy shared by many traditional methods [2] is to wait until feedback or new business needs emerge from users, and then manually analyze them. However, software nowadays evolves at a rather fast pace [3], and manual strategies cannot keep up. To address this problem, new requirements elicitation approaches have been proposed, but most of them focus on observing historical system defects [4], [5], or analyzing users' delayed feedbacks [6], [7]. A more prompt way to gather information that can lead to new requirements is much needed.

Furthermore, we believe that one major driving force for software evolution is human intention. A frontier work of human-intention-driven software evolution, *Situ* [8], was proposed

to support rapid and iterative requirements analysis as a general framework. Based on our review of requirements elicitation techniques, the main research focus in RE techniques is now shifting from the previous system-centered perspective towards to a human-centered one. During this transition, data, more specifically user behavioral and environmental context data, has become more important and valuable as the means of expressing human goals and context attributes. However, there is a huge gap between data and requirements, as commonly agreed in the fields of information system research [9], [10]. Hence, we propose to apply the Data-Information-Knowledge-Wisdom (DIKW) Hierarchy [9] to bridge such gap and present a divide-and-conquer approach to partition the huge gap into multiple smaller ones, so that each of them can be handled separately with existing methods and techniques. In other words, we will implement the concept of DIKW in a computable way to elicit requirements from data.

There are 3 research objectives for this work:

***Objective 1***: To investigate and evaluate the feasibility of applying DIKW for requirements elicitation, and the feasibility of using situation as the key linkage to connect different layers of DIKW.

***Objective 2***: To propose a general situation-centric, knowledge driven requirements elicitation approach on the basis of the result of **Objective 1**, by applying existing data analytic techniques, machine learning methods, and requirements modeling techniques.

***Objective 3***: To carry out case studies and demonstrate and evaluate the approach proposed in **Objective 2** on functional requirements elicitation. We also seek to further fine-tune the approach to improve elicitation results.

The proposed research will yield a general approach to take historical user behavioral and environmental context data as input, and produce a set of potential new requirements for

evolving the system. Furthermore, we hope that the proposed approach can be extended to the applications on runtime system decision making, when the corresponding knowledge base is available. Although in this work the notion of wisdom is embodied as requirements, it can also be interpreted as design rationales and beliefs, or other high-level software engineering intellectual artifacts. In essence, the proposed approach can be considered as one way to incrementally accumulate the body of knowledge of the specific system and user domain through data analysis. And we expect it intersects with the fields of software engineering, data analytics, and knowledge engineering [41], and eventually becomes a general architecture to utilize techniques across the above-mentioned fields.

The rest of this dissertation is organized as follows: Chapter 2 briefly introduces the DIKW Hierarchy and presents an overview of related work cutting across the area of requirements elicitation and goal model, human intention modeling, and the Multi-strategy, Task-adaptive, learning (MTL) method. Chapter 3 discusses some fundamental issues about the research problem and proposes a situation-centric approach to identifying new user intentions from user behavioral and environmental context data using the MTL method. Chapter 4 presents a case study of an online library system on new intention identification using our proposed method. Chapter 5 illustrates how to use a strategic rationale model to transform the newly identified intentions into the form of a strategic model, and how to fuse them with the existing knowledge network to harvest new design insights. Chapter 6 discusses threats to validity and conclude this work.

# CHAPTER II

# BACKGROUND & RELATED WORKS

## 2.1 Requirements Elicitation and Goal Models

The traditional requirements elicitation process can be considered an interactive mutual learning process between the requirements engineer and the customer [11]. The knowledge of users' requirements can be obtained from interview [12], feedback [2] or observation of customers' activities at their workplace [13]. As users' requirements are usually implicit and unpredictable [14], this process mainly depends on requirements engineers' subjective analysis and judgment, so it is usually time-consuming and results in inaccurate requirements. Oftentimes, a cycle of elicitation, modification, development, and deployment takes a long time to complete, usually several months [1]. Researchers are now facing the steep challenge of shortening such undesirably long evolution cycles to build timely patches and updates required by modern-day users. New technologies that can enable automatic or semi-automatic requirements elicitation and analysis are needed in order to realize rapid software evolution.

Goal models ([12], [15], [16], [33]) have been found to be effective in precisely and accurately capturing large numbers of alternative sets of low-level tasks, operations, and configurations that can fulfill high-level stakeholder goals. The characterization of a large space of such alternatives has been shown to be useful for evaluating alternative designs during the analysis process [17], for customizing designs to fit individual user characteristics [18], or even for coping with the large space of configurations of common desktop applications [19]. In this work, goal model will not be used directly. Instead, the concept of "goal" is embedded within the

*Situ* framework, and it is still an essential element to establish linkage from data to information, and further to knowledge.

## 2.2 The DIKW Hierarchy

The DIKW Hierarchy, also known variously as the "DIKW Pyramid", the "Wisdom Hierarchy", the "Knowledge Hierarchy", the "Information Hierarchy", and the "Knowledge Pyramid", refers loosely to a class of models for representing structural and/or functional relationships between data, information, knowledge, and wisdom. The implicit assumption is that data can be used to create information; information can be used to create knowledge; and knowledge can be used to create wisdom. As Ackoff [9] explains, each of the higher types in the hierarchy includes the categories that fall below it (see Figure 1).

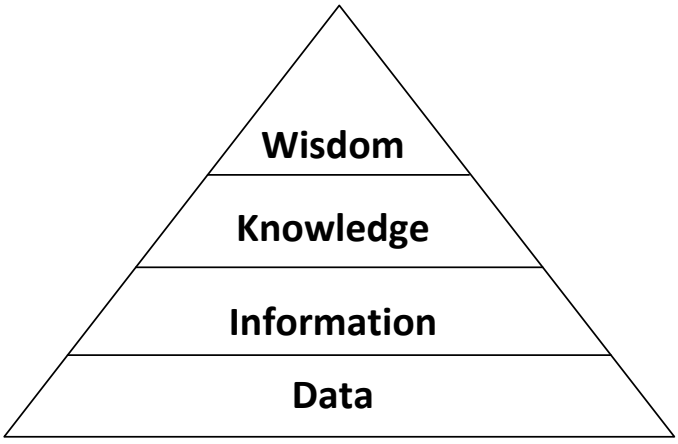


**Figure 1.** DIKW Hierarchy

The definitional role of the DIKW Hierarchy positions it as a central model of information management, information systems and knowledge management. It sets the stage for disambiguating data from knowledge and sets definitional boundaries for what data, information and knowledge are. However, there has been significant debate over the years on the nature and

definition of information, both before and since Ackoff's paper, e.g. [20–23]. The main reason for the existence of such a debate is due to the difficulty in reaching a uniform distinction between the definitions of information and knowledge at general theoretical and philosophical levels. But, when dealing with a specific domain, such definitional confusion can be overcome by providing domain-specific definitions for each concept. In our work, this strategy has been adopted. With this approach, the DIKW Hierarchy provides an intuitive way to divide and conquer the problem of knowledge acquisition and application when each concept (in DIKW) is carefully and clearly defined in the domain. Also, what is equally important is that the transitions from each lower layer to a higher layer within the DIKW Hierarchy can help us interpret the nature of each sub-learning-problem.

### 2.3 Human Intention Modeling

In the fields of Philosophy, Cognitive Science [24], and Artificial Intelligence [25], [26], the definition of human intention has been well studied. However, the concept of human intention has not been rigorously investigated through a computational perspective until just a decade ago. *Situ* [8] is a general approach proposed for human-intention-driven service evolution in context-aware service environments. *Situ* as a computational framework allows people to model and detect human intentions by inferring human desires, as they are often largely hidden, and capturing the corresponding context values through observations. In the early phase of *Situ* research, Hidden Markov Model (HMM) [27] was adopted for desire inference and intention detection. However, it was recently recognized that HMM is not able to provide the optimal inference accuracy due to its model limitations [38], [39].

Recently, the Conditional Random Field (CRF) method [36] was proposed for automatic desire inference, and its capability has been demonstrated for better inference accuracy compared to HMM. Figure 2 shows the latest development of the *Situ* framework, in which the relations between desire, actions, and environmental context variables have been further investigated, and domain knowledge and characters has been converted into mathematical models as feature functions in CRF.

The current *Situ* framework remains incomplete in terms of computation ability, as it still relies on manual analysis to identify new human intentions from the inferred desires. Our work using the MTL method seeks to extend *Situ*'s ability to automatically detect new intentions from inferred desires with limited help from human oracle.

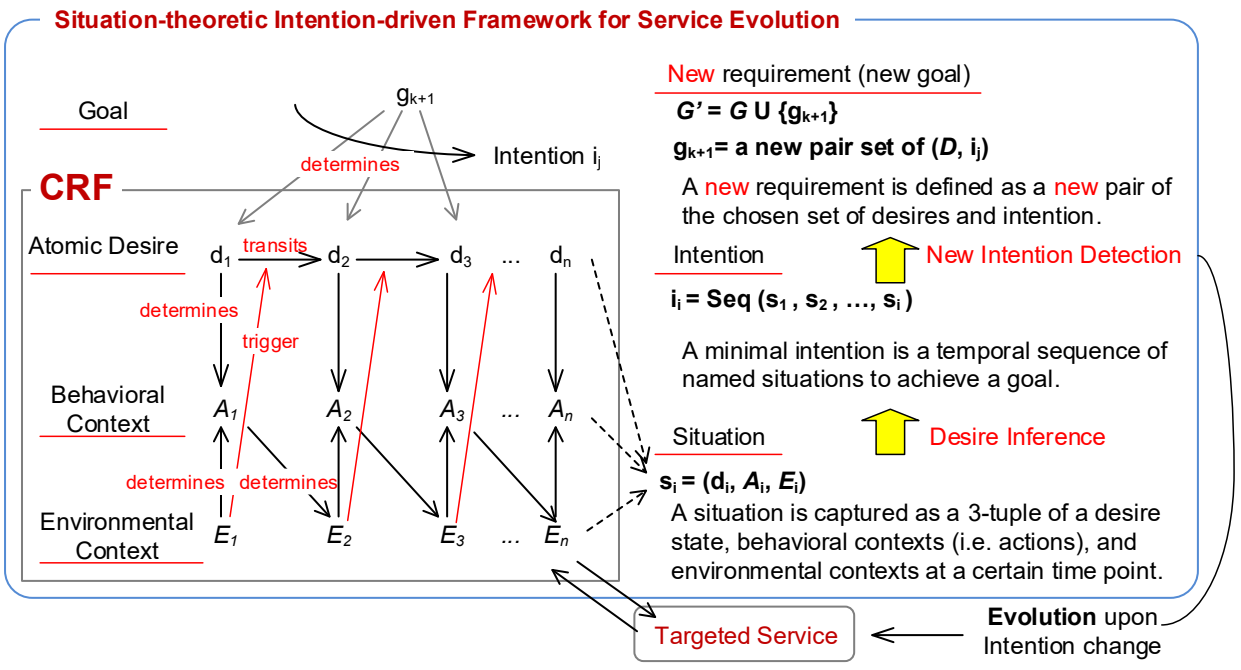


**Figure 2.** Using CRF to infer human desires in *Situ*

## 2.4 The MTL Method

With the growing understanding of capabilities and limitations of single-strategy learning methods, there has been an increasing interest in multi-strategy learning methods that employ two or more inference types and/or computational mechanisms [28], [29]. Multi-strategy learning methods have potential for much greater competence, that is, the ability to solve a much wider range of learning problems than single-strategy methods, because they take advantage of the complementarity of individual learning strategies (see Table 1).

In this work, we adopt the MTL method [30], because theoretically, it only requires incomplete and partially correct knowledge base (KB) with at least one positive example, which is suitable for the application scenarios of our work, considering system evolution as the process of correcting and perfecting system requirements. Also, this method has been applied to elicit requirements for agile development [31], showing some potential for further application with other requirements modeling techniques, such as goal models [12], [15], [16], and [33]. In terms of learning, the CRF method is essentially a type of analogy & case-based method, while other types of learning methods that MTL includes may serve as complementary roles to help discover new knowledge, i.e., new intentions.

**Table 1.** Requirements and results of different types of single-strategy learning method

| Learning Type | Application Requirements | Type of Result |
|---|---|---|
| **Empirical induction** | requires many input examples and a small amount of background knowledge | a hypothetical generalization of several input examples |
| **Explanation-based** | requires one input example and a complete background knowledge | an operational generalization of an input example |

**Table 1 continued.**

| Learning Type | Application Requirements | Type of Result |
|---|---|---|
| **Analogy & case-based** | require background knowledge analogous with the input | new knowledge about the input |
| **Abduction** | requires causal background knowledge related to the input | new background knowledge |

# CHAPTER III

# A SITUATION-CENTRIC APPROACH TO IDENTIFYING NEW INTENTIONS USING THE MTL METHOD

In this chapter, we first discuss the feasibility of bridging the gap between user behavioral and environmental context data and requirements, with situation as the key linkage to connect different layers. Then, an intuitive research idea is informally described. Based on that, in 3.3, we discuss the necessary modification to the existing MTL method. In 3.4, a situation-centric approach using the MTL method is proposed to identify new user intentions [48].

## 3.1 Feasibility

### 3.1.1 Feasibility of eliciting requirements from data

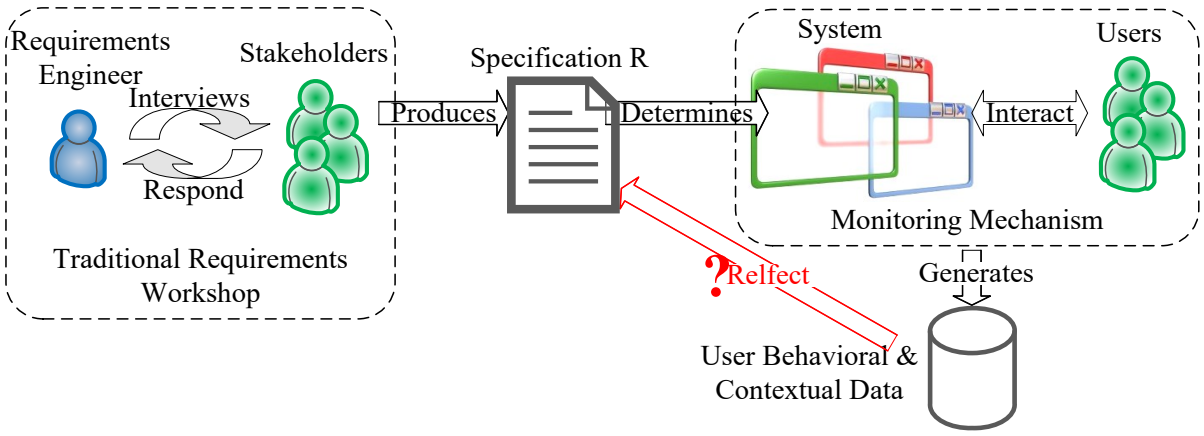


**Figure 3.** Potential linkage between data and requirement specification

Figure 3 depicts the potential linkage between data and system requirements in a typical software lifecycle. As the starting point of the whole process, a requirements workshop produces system specification through stakeholders' input and requirements engineers' elicitation efforts,

which is later implemented in the system design. After deployment, users interact with the system through instrumented operations, and such a process can generate a huge amount of data. In our analysis, it is a time-series user behavioral data with corresponding contextual parameters. These data show historical operational status within the scope containing system, users, and related contextual variables, i.e. historical facts. Assuming that the system has been correctly designed and implemented (the system is in compliance with its specification), the interaction between the system and its users shall also be consistent with what has been anticipated in the requirements, which is what the data can and should reflect. So, theoretically, data can be viewed as visible states that gives certain hints on the original requirements specification.

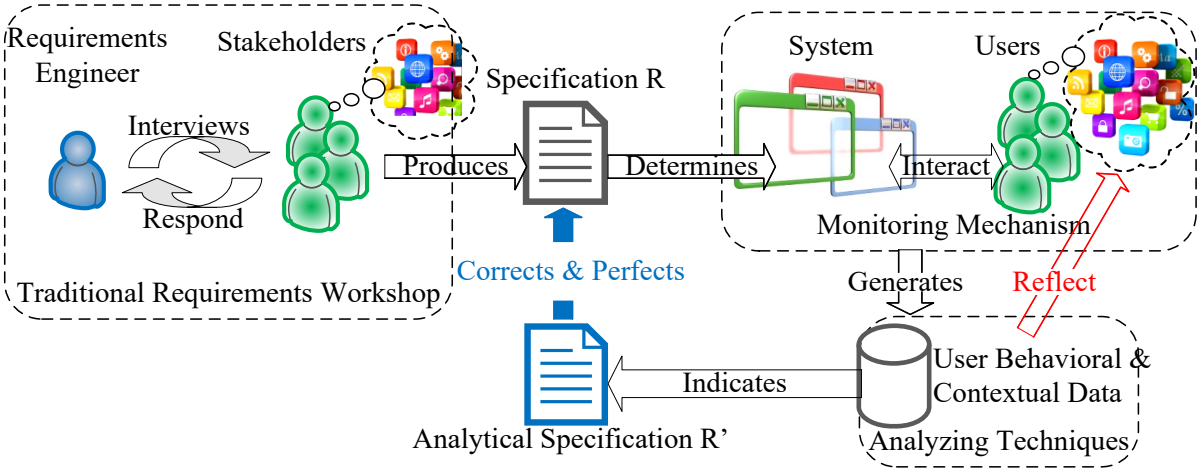


**Figure 4.** System evolution based on user behavioral and contextual data analysis

However, there is a strong yet implicit assumption that the specification must completely and accurately capture what the users need. Unfortunately, in many realistic cases, this assumption does not hold true. A common cause is that stakeholders may not have a clear and concrete idea of their needs, making it difficult for requirements engineer to elicit. Consequently, the linkage between data and requirements may be not as straightforward as is shown in Figure

3. More realistically, as shown in Figure 4, the imagined system in the requirements elicitation phase might not faithfully project the same end users. Such inconsistency can result in changes to the system, i.e., new requirements, driving system evolution. Additionally, another side-effect is that rational users may get to know that the differences between their comprehensive projection of the system and the actual one through their interactions with the system, and may adjust their behaviors accordingly to fulfill their needs. Such adjustments can be captured and reflected in their behavioral data. Analysis on these data is likely to reveal the character of the users' actual needs, and can help refine the current system specification to eventually evolve the system.

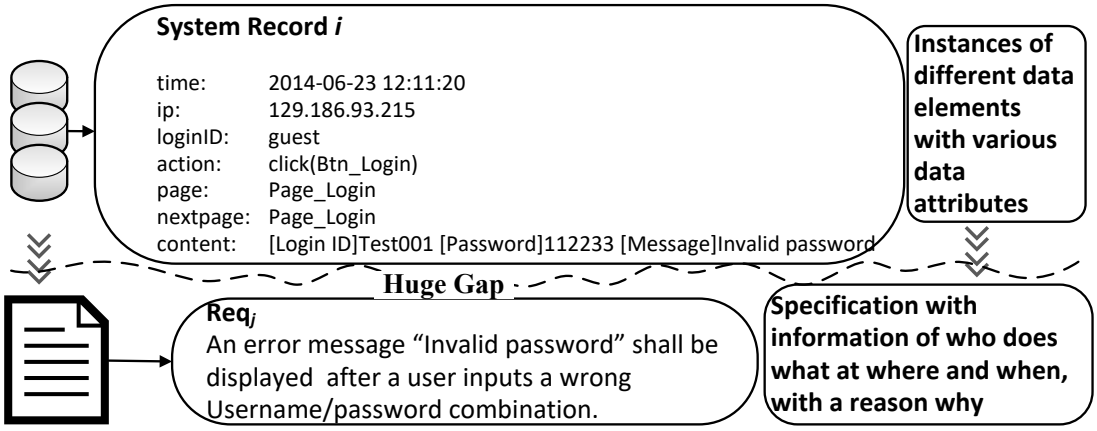


**Figure 5.** The huge gap between raw data and system requirements

Based on the above discussion, the idea of eliciting new requirements from data may sound feasible. However, one problem that has been overlooked so far is the huge gap between raw data and system requirements (shown in Figure 5). In essence, requirements are the abstraction of the "right" system behaviors, while data is the output or side-effect of those system behaviors through interaction with users. There are too many variables between the two sides, not to mention that there are even more relationships to be analyzed. To overcome such data complexity, a common solution is to reduce it by certain learning goals. Also, as discussed in

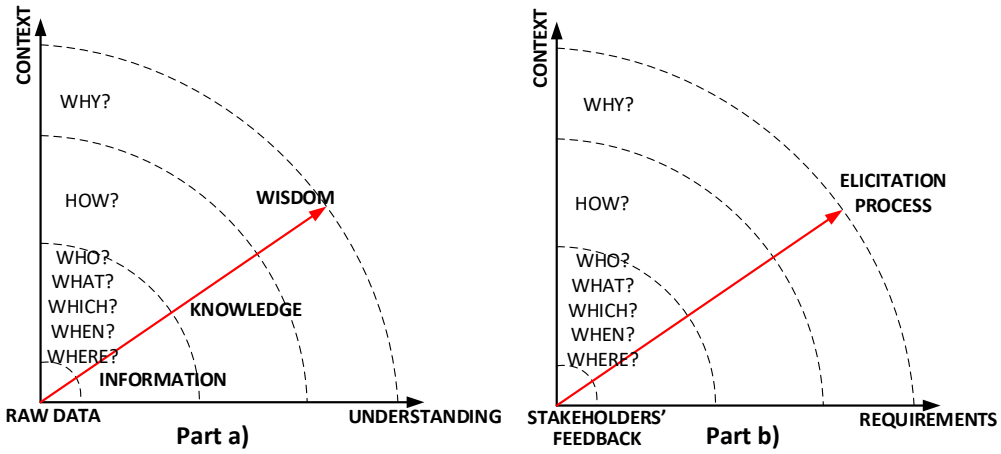


**Figure 6.** Traditional requirements elicitation vs. DIKW structure

section 2.2, the concept of DIKW Hierarchy can provide a natural workflow to divide and conquer the problem.

Figure 6 presents one possible visualization of the DIKW Hierarchy (Part a)). As the understanding of the domain increases, data is processed into information as relations between data, information is synthesized into knowledge as information patterns, and knowledge is explained in terms of wisdom of principles. Interestingly, the type of questions that can be answered in each layer are exactly the key concerns of the RE process [34] (see Part b) of Figure 6). Note that in both figures, when requirements and wisdom continue to emerge, more context factors are explored and involved, which indirectly indicates the possibility of using situation [8] as a key linkage within the DIKW Hierarchy.

### 3.1.2 Feasibility of using situation as linkage within the DIKW Hierarchy

A classic application of the DIKW Hierarchy is the order history mining problem [35], shown in the upper part of Figure 7. In the data layer, many online shopping order records are

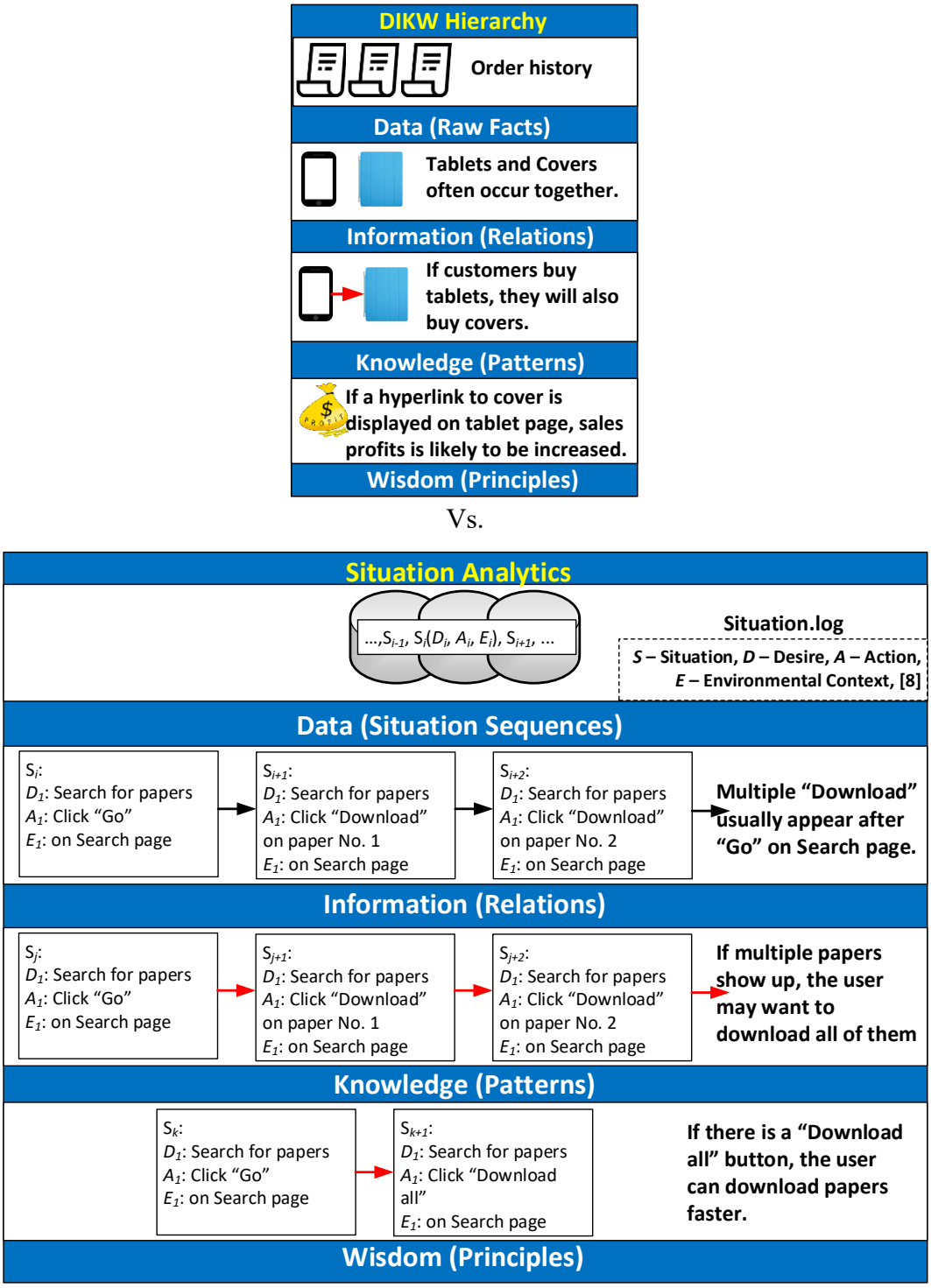


**Figure 7.** An online order history mining problem vs. situation-centric user behavior analysis on "paper download" example

given. However, it is hard to see any relations between them, since each list contains a random list of items. First, these data are processed so that frequently purchased items can be identified; the result shows that "Tablets" and "Covers" often occur together. Next, such relationships are further synthesized into knowledge, and a practical, more meaningful pattern surfaces: "If there are tablets ordered, there will be covers in the same order." Knowledge represented as this kind of pattern can be applied to a future contexts. For example, in a context of pursuing the maximum profits, sellers may want to have "Covers" linked on the "Tablet" page, so that customers can easily find both items and purchase them together. As such, applying new knowledge to an existing knowledge base in order to realize certain goals can be considered wisdom. Similarly, the situation-centric user behavior analysis process (shown in Figure 7, lower part) can be readily mapped to each phase of the above order history mining example. Using the approach discussed above on a "*paper download*" example (within an online library system), raw situation sequences can be processed, and a relation between different situations "multiple *DOWNLOAD* operations usually appear after each *SEARCH* operation" can be found. Based on this information, a situation pattern can be further synthesized as "If multiple papers show up as a search result, users may want to download all of them". When this new knowledge is added into the existing knowledge base, and interacts with other design principles, a wisdom can emerge as "Adding a new button "*DOWNLOAD ALL*" to allow users to download papers more easily".[1]

In fact, we observed that the concept of DIKW had been applied to the *Situ* framework to some extent, through the process of desire inference using the CRF method [36], [38], [39], [40].

[1] A more refined version of this requirement could be "The system shall allow users to be able to select multiple papers including a SELECT-All option, and users shall be able to download them all at once."

As shown in Figure 8, the raw data, a set of observation sequences, are filtered based on relevance of context variables, i.e., non-relevant context variables are filtered out, and raw observations (data) are processed into refined observations, and a set of refined observation sequences (information) can be acquired. Then the CRF method can be applied, and map different parts of each observation sequence with a pre-defined desire. As a result, each observation with the labeled desire together constructs a situation, and those sequences of situation with the same desire can form an intention (knowledge), according to the definition of "intention" in *Situ* [8]. In essence, the whole desire inference process can be considered as an application of the Map-Reduce scheme, and covers the first three layers in the DIKW Hierarchy. Furthermore, to harvest wisdom, it totally depends what the goal is. For example, one may be interested in the transition of desires, and would like to know if there is any desire transition patterns, so that some potential compound desire can be detected. Or, others may be more concerned about the similarity between different situation sequences but labeled with the same intention, and may possibly identify alternative ways to fulfill the same intention. Hence, to make the leap from knowledge to wisdom, it is necessary to set up a specific goal first, and based on that decide a suitable solution. Considering our ultimate goal of this work is to elicit new requirements, an immediate next step naturally becomes identifying the new intentions among all situations sequences.

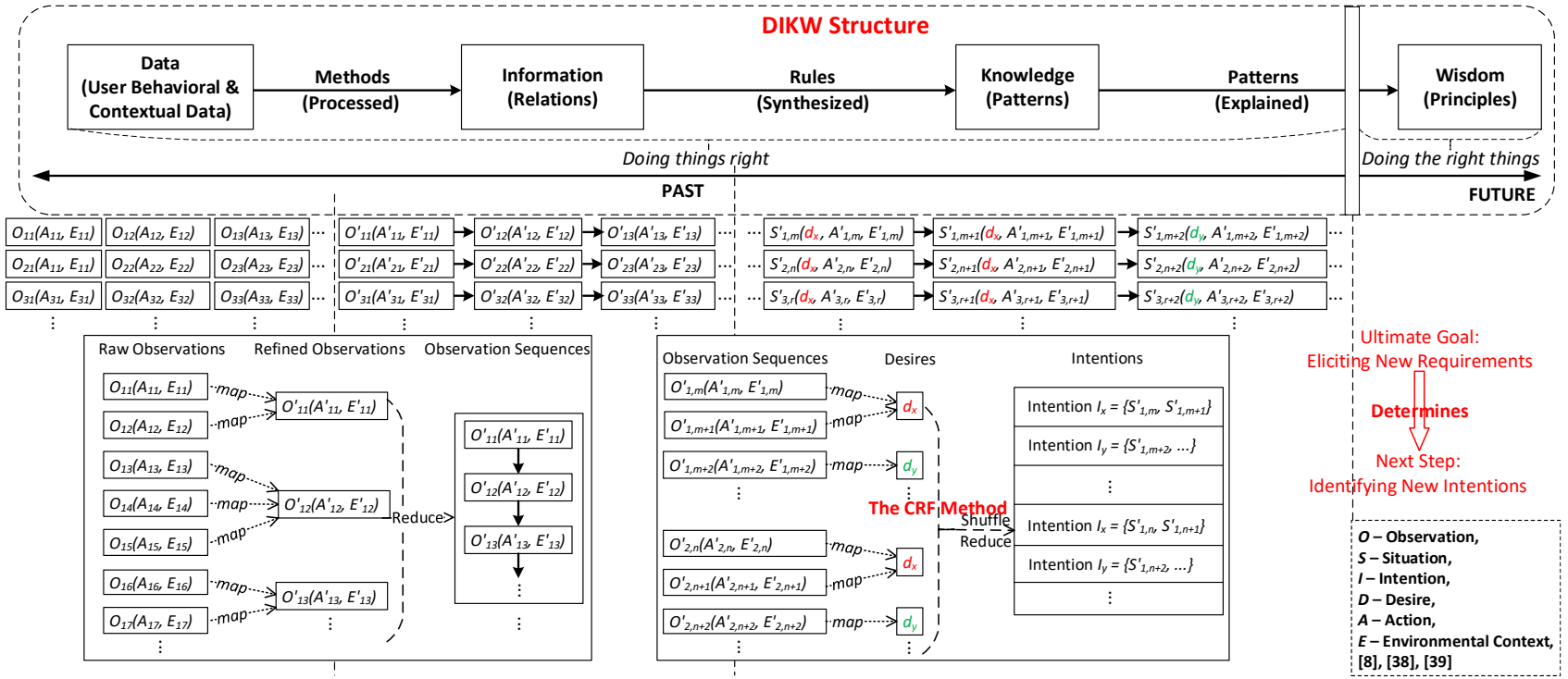


**Figure 8.** Applying *Situ* within DIKW Hierarchy – A CRF Method

### 3.2 An Intuitive Idea

As discussed in the previous section, an immediate goal that may be achievable is to infer new intentions based on the current desire inference result by using the CRF method, since in the current *Situ* framework, new intentions are still manually identified [38], [39]. Figure 9 presents an intuitive idea of the whole workflow powered by an MTL engine. The application scenario is to refine the requirements of an existing system or a prototype, which has been set up with a monitoring mechanism to capture user behavioral and relevant environmental context data. As reported in [38], [39], the CRF method has been applied to the raw data (in form of situations), and successfully labeled each situation with an appropriate pre-defined desire. Noise introduced by the CRF method (i.e. inaccurate inference results) is allowed. The initial knowledge base is generated based on existing knowledge of the current system. To be specific, a set of intention decomposition rules are extracted from the specification of the current system. Moreover, known positive and negative examples are selected from the training data set used to train the CRF model. Note that these known examples must be completely correct and noise-free.

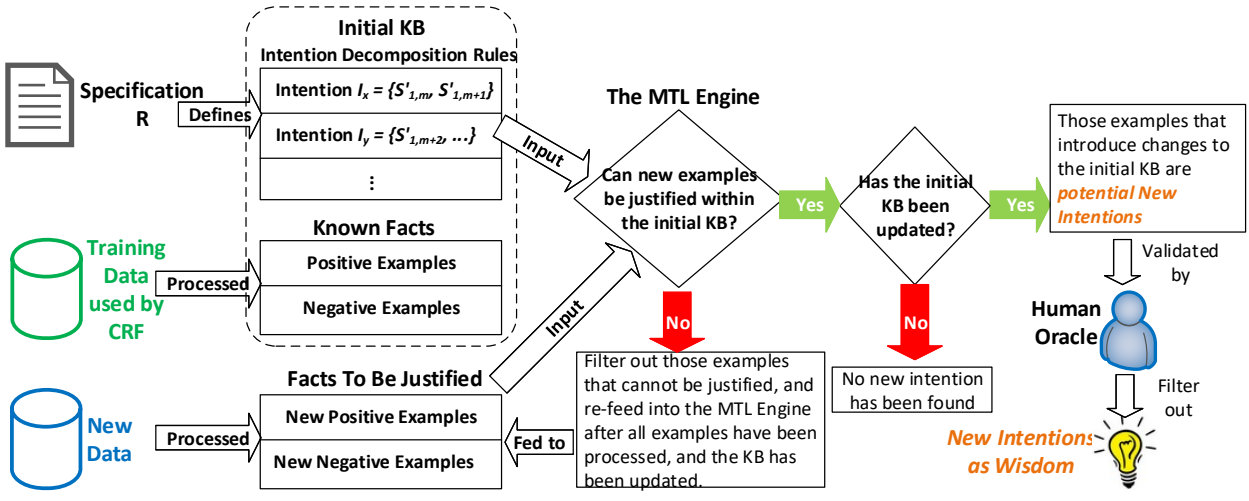


**Figure 9.** An intuitive idea of new intentions detection using the MTL method

New examples as facts to be justified are acquired from the set of situation sequences with inferred labels, grouped by desire type. Then, we feed the initial knowledge base and new examples into the MTL engine, and start the learning process. The MTL engine will attempt to justify each example within the available KB. If some examples cannot be justified, they will be put into a set, and re-fed into the MTL engine after all other examples are processed (either justified or not). For those examples that can be justified, check if the KB has been added with new rules, i.e., has been updated during the justification process. If not, there are no new intentions found. Otherwise, those examples that introduce changes to the KB are the potential new intentions, which will be later validated by human oracle, and filtered into real new intentions as wisdom.

Roughly speaking, the above approach appears to be very similar to the classical MTL process. However, a major difference is that the MTL method is usually used for concept learning, and situation sequences are a quite different "concept", because of the temporal attribute embedded within each sequence. Hence, to carry out new intention learning, the classical MTL method needs to be modified in order to handle the temporal factor of situation sequence.

## 3.3 The Meta Model

To make *Situ* and MTL work together, situation sequence has to be represented in a form that is decomposable. There is an existing first-order language, *SiSL* [32], to formally describe the situation domain. However, in *SiSL*, situation is defined as a basic entity, which makes it

difficult to further decompose it. Hence, we propose another representation scheme which fits the learning needs.

A set of basic elements are defined as the following:

*OBJECT o.*

*DESIRE* is represented using a first-order predicate, *d(x).*

*ACTION* is represented using a first-order predicate, *a(y).*

*CONTEXT* is represented using a first-order predicate, *c(z).*

*SITUATION*: $s(p) = (d(x), (a_i(y), (c_1(z_1), c_2(z_2),\ldots,c_n(z_n)))$, $p$ is a first-order predicate. a situation is satisfied when *d, a, C* are all satisfied.

*INTENTION*: $I(q) = \{s_1, s_2,\ldots,s_i\}$, in which, $d_1=d_2=\ldots=d_i$, $q$ is a first-order predicate. An intention is satisfied when the final situation in the corresponding situation sequence is reached.

*CONSTRAINT* is constructed with a number of predicates, to specify certain conditions that must be satisfied within the intention, or the condition of intention decomposition.

*AND* joint is used when decomposing an intention into a number of sub-intentions, and the intention is satisfied if and only if all of the sub-intentions are satisfied.

*OR* joint is used when decomposing an intention into a number of alternative intentions, and the intention is satisfied when any of the alternative(s) is/are satisfied.

*Time-AND* joint is used to decompose an intention into its corresponding situation sequence. Each situation is associated with a time-label, to indicate its temporal order within the sequence. An intention is satisfied if and only if all its adjacent lower-level situations are satisfied in the specified temporal order.

Figure 10 shows the meta-model for intention decomposition. In essence, it is a combination of KAOS meta-model [47] and *Situ* [8].

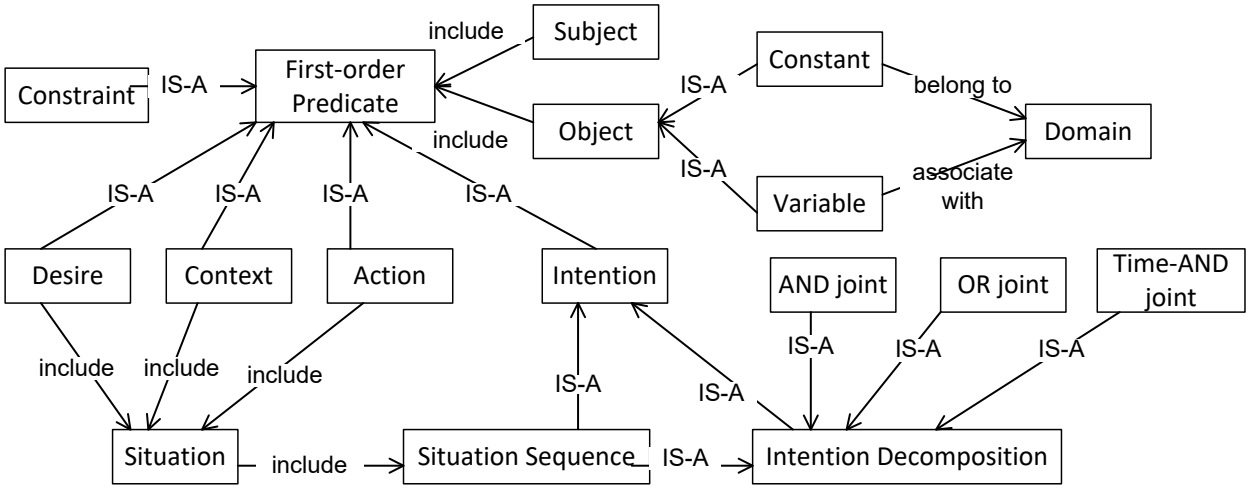


**Figure 10.** Meta-model for intention decomposition.

## 3.4 The Approach

Based on the discussion in Sections 3.1, 3.2 and 3.3, we now can formally propose a situation-centric approach to identifying new user intentions. In Section 3.2, the pre-requisite of the approach was discussed (as an assumption). The ultimate goal is to identify potential new intentions, which will be further exploited as new requirements for future development or evolution. In this way, our approach is different from classical concept learning, which is mainly based on positive examples. In our case, knowledge about the negative examples are as important as that of the positive ones, because new requirements can be found through analysis of how things can go wrong. Because of that, it is necessary to learn both positive and negative justification rules. Figure 11 is the overview of the proposed approach.

**Step 1** Initial KB generation

a) Inference rules including:

i. a set of rules $T_f$ to infer that an intention is fulfilled.

ii. a set of rules $T_u$ to infer that an intention is still unfulfilled.

b) Constraints

c) Examples

**Step 2** Learning examples generation

**Step 3** Learning the first positive (or negative) example $I_1$

a) Build a plausible justification tree $T$ of $I_1$

i. Always try to justify $I_1$ using the strongest inference method available. (Deduction > Analogy > Inductive Prediction > Abduction)

ii. If $I_1$ cannot be justified, add it into the set of unjustifiable examples $U$

b) Build the plausible generalization $T_f$ (or $T_u$) of $T$

c) Generalize the KB so as to entail $T_f$ (or $T_u$)

**Step 4** Learning from each new positive (or negative) example $I_i$

a) Generalize $T_f$ (or $T_u$) so as to cover a plausible justification tree of $I_i$

b) Generalization of the KB so as to entail the new $T_f$ (or $T_u$)

**Step 5** Learning from each negative (or positive) example $I_j$

a) Specialize $T_f$ (or $T_u$) so as not to cover any plausible justification tree of $I_j$

b) Specialized the KB so as to entail the new $T_f$ (or $T_u$) without entailing the previous $T_f$ (or $T_u$)

**Step 6** Re-feed $U$ into the updated KB, and repeat **Step 3 – 5** until the set remains stable.

**Step 7** Finalize updated KB, identify new intentions.

**Figure 11.** Overview of the proposed approach

With the help of the newly defined meta-model, the whole learning process is quite similar to that of a standard MTL concept learning [30], except for Steps 1, 2, and 7, which are described as follows.

***Step 1. Initial KB generation***

The initial knowledge base includes a set of inference rules, a set of positive and/or negative examples, and a set of constraints (if necessary). The inference rule set consists of a subset of rules $T_f$ to infer that an intention is fulfilled, and a subset of rules $T_u$ to infer that an intention is still unfulfilled. Sources to generate the initial KB are all available requirements artifacts of the existing system, e.g., specification, and training data set used by the CRF method.

***Step 2. Learning examples generation***

As discussed in Section 3.2, new learning examples (both positive and negative ones) can be generated from new data (situation sequences) that have been labeled with pre-defined desires by the CRF method. Find out all sequences with the same desires. Certain rules can be defined to separate positive examples from negative ones based on system domain characteristics. For example, a certain final situation should be reached in all positive examples. To achieve that, the situation sequence should contain a minimum number of situations essential for accomplishing an intention.

***Step 7. Finalize updated KB and identify new intentions***

Identify examples that introduce changes to the initial KB as potential new intentions. Human oracle will manually check and filter out real new intentions based on their domain knowledge. For those examples left in U, we will treat them as unjustifiable facts, and will keep them for later usage when new learning data is available.

It must be noted that the learning process of $T_f$ (or $T_u$) is completely independent of that of $T_u$ (or $T_f$), i.e., they are two separate learning processes. At the end of the learning, consistency checking should be formed in order to make sure if there is any conflict between $T_f$ and $T_u$.

# CHAPTER IV

# CASE STUDY: NEW INTENTIONS IDENTIFICATION USING THE MTL METHOD

## 4.1 Data Collection through The CoRE System

### 4.1.1 Experiment platform

The raw data is collected from an online library system, called Cooperative Research Environment (CoRE), which has been designed and developed for a research community to share their thoughts and views on academic papers. It was modified based on an open-source web application, MyReview [37], for managing the process of paper submission and paper review. The original system has served many academic conferences, and our modifications still keep its basic functionalities. CoRE users can upload research papers, submit comments for papers, and view paper information, etc. Figure 12 shows the use case diagram of the CoRE system. Figure 13 shows the interface of the CoRE system where users can edit a paper.

To monitor users' behaviors and capture related context values, an embedded program is deployed in CoRE as a sensor. Users' operations, papers and comments submissions, and the contents on the web pages entered by users will be recorded and stored in the database. During each experiment session, users report/select their current desires from a dropdown list containing a set of expected desires. Examples of desire options are "*Upload a paper*", "*Submit a comment*", "*View a paper information*", etc., and "*Not in the list*" (as a wild card). The users can also correct their desire selections in the post-session questionnaire in case that they forgot to report desires or reported an incorrect one during the experiment. It is very important to have users report their desires because these data will later be used to validate our desire inference

result, which is the input to the MTL engine. In fact, as part of our data processing, domain experts will need to manually check if users' reported desires are legit for our validation purpose. More detailed information about our experiment on CoRE can be found in [38] [39].

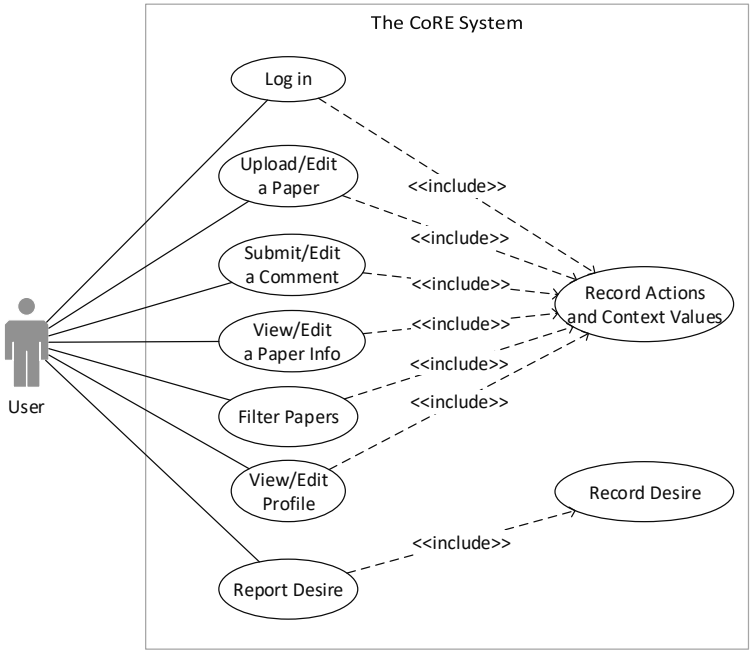


**Figure 12.** The use case diagram of the CoRE system.


**Figure 13.** *CoRE system interface – Edit paper*

Figure 14 a) and b) show the interfaces 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 desires during the experiment.

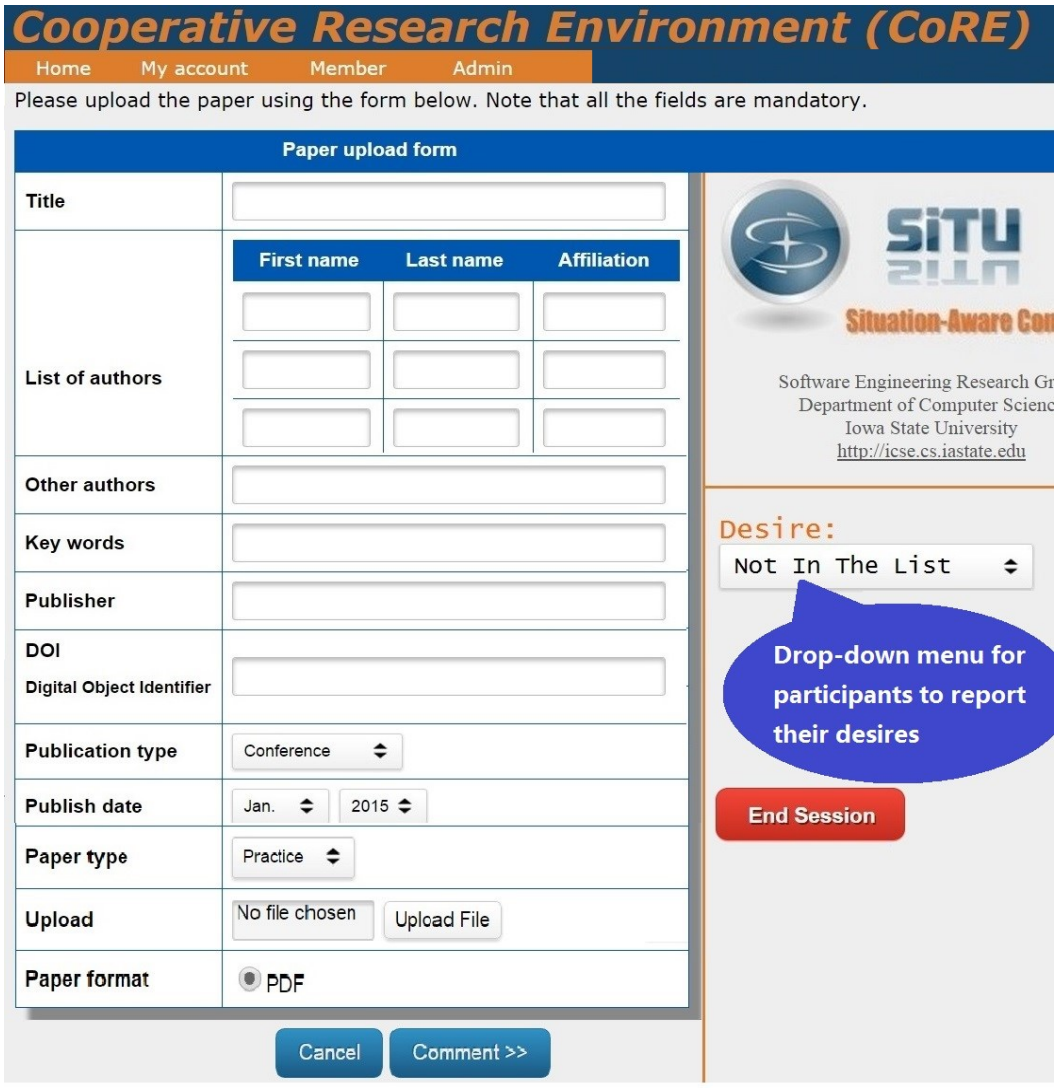


**Figure 14. a)** Paper Information page for uploading a paper in CoRE Version I.

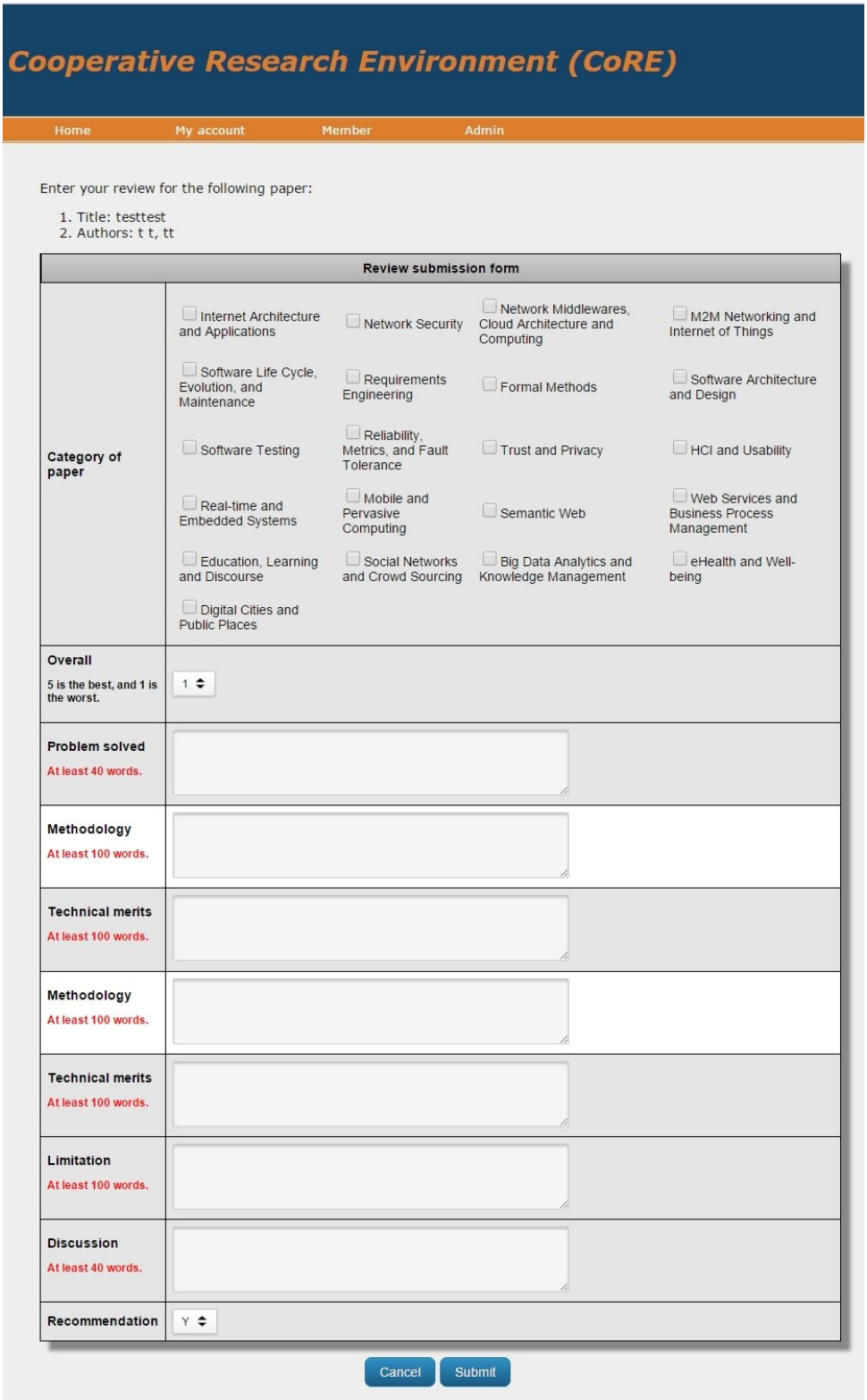


**Figure 14. b)** Comment page for uploading a paper in CoRE Version I.

### 4.1.2 Procedure of an IRB approved experiment

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

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

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

5. Run the second-round experiment for 30 days: invite some new participants, collect data records of participants' actions, desires and context values.
6. Shut down CoRE Version II. Apply our proposed methodology on the raw data captured in Round 2, using the CRF method to infer participants' desires.
7. Evaluate the effectiveness of our methodology on the evolution of CoRE from Version I to II.

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

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

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

**Table 2.** Example Raw Data Record

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

Each raw data record has the following attributes, with an example shown in Table 2:

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

### 4.1.3 Data collection and preprocessing

There are 10,063 raw data records and 585 experiment sessions captured in the first-round experiment, and 10,524 raw data records and 582 experiment sessions captured in the second-round experiment. A raw data record contains all the attributes shown in Table 2. A record of an experiment session is the data sequence that starts when the user logs into the experiment, and ends when the user logs out.

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

1. If most (>50%) self-reported desires are "Not in the list", which is the default value in the desire selection dropdown list. We will assume that in this case the participant forgot to report his/her desire at all or most of the time in the experiment.
2. If the answer is "no" for the question "Did you select the desire every time when you had a new desire?" in the post-session questionnaire.
3. After filtering based on (1) and (2) was done, we had the rest of the data records manually checked by domain experts and system designers, and evaluated in the perspective of whether the participants' self-reported desires were reasonable or not. Those obviously wrong ones should and have been removed because they cannot be used to validate our desire inference results. An example is that a participant might have forgotten to report his desire (change) when he accomplished his previous task and started a new one.

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

### 4.1.4 Result of desire inference by the CRF method

The preprocessed data set in each round was separated into two sets, one training set, which is used to train the CRF model, and one testing set. For each record in the testing set, the

trained CRF model was used to infer its corresponding desire. So at the end of the process, each data record (situation) has been labeled with a desire from the pre-defined set. Our experience shows that the inference accuracy was reliably above 90% in this very limited case study, which is more than acceptable. This experiment has also been externally validated by anonymous reviewers [39].

### 4.2 Case Study on New Intentions Identification

In this case study, we studied the specification of the CoRE system, and abstracted out a set of inference rules. The training data set used to train the CRF model was studied, and representative situation sequences were selected as known positive and negative examples. It is important to make sure that the inference rules are consistent with the specification, which means they should reflect what has been implemented in the current system. However, it does not mean that the inference rules must be complete and correct, because it is assumed that the system is not a perfect one. On the other hand, the examples must also be consistent with the initial inference rules. To ensure the above mentioned qualities, the initial knowledge base has been checked by domain experts. Note that for each intention, a corresponding KB will be constructed, meaning that the learning is about the unanticipated situation sequence to fulfill a certain intention. Table 3 shows the initial KB for intention "EditPaper($x$)". Next, a set of new examples are generated from the set of situation sequences grouped by the inferred desires. For intention "EditPaper($x$)", a set of examples has been filtered out as shown in Table 4. (Note that the user-reported desires are not used in this case, because in most of real-world application scenarios, there is no such

data available.) Based on the CoRE domain character, a set of filtering rule has been defined as follows:

(1) Minimum situation sequence length for positive examples shall be 2;

(2) Final situation for positive examples shall be $s_2$(EditPaper($p_1$), clickSubmitEditPaper($p_1$), Context(PaperFile($p_1$), "*Uploaded*"), Context(PaperSubmitStatus($p_1$), "*Yes*"))

**Table 3.** The Initial KB

**Deductive rules:**

- Time-AND{Situation(clickEditPaper&PaperID), Situtation(clickSubmitEditPaper&EditPaperGood)} => Intention(EditPaper(x), Fulfilled)
- {Situation(EditPaper(*x*), Action(any), Context(PaperFile, "*Empty*"), Context(PaperSubmitStatus, "*No*"))} => Intention(EditPaper(*x*), *Unfulfilled)*

**Examples**:

- **Positive example 1**: Intention(EditPaper($p_1$), *Fulfilled*) = {$s_1$(EditPaper($p_1$), clickEditPaper&PaperInfo($p_1$), Context(PaperFile($p_1$), "*Empty*"), Context(PaperSubmitStatus($p_1$), "*No*")), $s_2$(EditPaper($p_1$), clickSubmitEditPaper($p_1$), Context(PaperFile($p_1$), "*Uploaded*"), Context(PaperSubmitStatus($p_1$), "*Yes*"))}
- **Positive example 2**: Intention(EditPaper($p_2$), *Fulfilled*) = {$s_1$(EditPaper($p_2$), clickEditPaper&PaperInfo($p_2$), Context(PaperFile($p_2$), "*Empty*"), Context(PaperSubmitStatus($p_2$), "*No*")), $s_2$(EditPaper($p_2$), clickSubmitEditPaper($p_2$), Context(PaperFile($p_2$), "*Uploaded*"), Context(PaperSubmitStatus($p_2$), "*Yes*"))}
- **Negative example 1**: Intention(EditPaper($p_3$), *Unfulfilled*) = {$s_1$(EditPaper($p_3$), clickEditPaper&PaperInfo($p_3$), Context(PaperFile($p_3$), "*Empty*"), Context(PaperSubmitStatus($p_3$), "*No*")), $s_2$(EditPaper($p_3$), clickCancelEditPaper&PaperID($p_3$), Context(PaperFile($p_3$), "*Empty*"), Context(PaperSubmitStatus($p_3$), "*No*"))}
- **Negative example 2**: Intention(EditPaper($p_4$), *Unfulfilled*) = {$s_1$(EditPaper($p_4$), clickEditPaper&PaperInfo($p_4$), Context(PaperFile($p_4$), "*Empty*"), Context(PaperSubmitStatus($p_4$), "*No*")), $s_2$(EditPaper($p_4$), clickCancelEditPaper&PaperID($p_4$), Context(PaperFile($p_4$), "*Empty*"), Context(PaperSubmitStatus($p_4$), "*No*"))}

**Table 4.** Positive and negative examples of "EditPaper($x$)"

- **Positive example 1**: Intention(EditPaper($p_5$), *Fulfilled*) = {$s_1$(EditPaper($p_5$), clickEditPaper&PaperInfo($p_5$), Context(PaperFile($p_5$), "*Empty*"), Context(PaperSubmitStatus($p_5$), "*No*")), $s_2$(EditPaper($p_5$), clickSubmitEditPaper($p_5$), Context(PaperSubmitStatus($p_5$), "*Yes*"))}
- **Positive example 2**: Intention(EditPaper($p_6$), *Fulfilled*) = {$s_1$(EditPaper($p_6$), clickEditPaper&PaperInfo($p_6$), Context(PaperSubmitStatus($p_6$), "*No*")), $s_2$(EditPaper($p_6$), clickSubmitEditPaper&NoFile($p_6$), Context(PaperFile($p_6$), "*Empty*"), Context(PaperSubmitStatus($p_6$), "*No*")), $s_3$(EditPaper($p_6$), clickSubmitEditPaper($p_6$), Context(PaperFile($p_6$), "*Uploaded*"), Context(PaperSubmitStatus($p_6$), "*Yes*"))}
- **Positive example 3**: Intention(EditPaper($p_7$), *Fulfilled*) = {$s_1$(EditPaper($p_7$), clickEditPaper&PaperInfo($p_7$), Context(PaperSubmitStatus($p_7$), "*No*")), $s_2$(EditPaper($p_7$), clickSubmitEditPaper&NoFile($p_7$), Context(PaperFile($p_7$), "*Empty*"), Context(PaperSubmitStatus($p_7$), "*No*")), $s_3$(EditPaper($p_7$), clickSubmitEditPaper($p_7$), Context(PaperFile($p_7$), "*Uploaded*"), Context(PaperSubmitStatus($p_7$), "*Yes*"))}
- **Negative example 1**: Intention(EditPaper($p_8$), *Unfulfilled*) = {$s_1$(EditPaper($p_8$), clickEditPaper&PaperInfo($p_8$), Context(PaperFile($p_8$), "*Empty*"), Context(PaperSubmitStatus($p_8$), "*No*")), $s_2$(EditPaper($p_8$), clickSubmitEditPaper&NoFile($p_8$), Context(PaperFile($p_8$), "*Empty*"), Context(PaperFile($p_8$), "*Empty*"), Context(PaperSubmitStatus($p_8$), "*No*"))}
- **Negative example 2**: Intention(EditPaper($p_9$), *Unfulfilled*) = {$s_1$(EditPaper($p_9$), clickEditPaper&PaperInfo($p_9$), Context(PaperFile($p_9$), "*Empty*"), Context(PaperSubmitStatus($p_9$), "*No*")), $s_2$(EditPaper($p_9$), clickCancelEditPaper&PaperID($p_9$), Context(PaperFile($p_9$), "*Empty*"), Context(PaperSubmitStatus($p_9$), "*No*"))}

So far, Step 1 and 2 (in Figure 11) have been completed. And now we can start learning $T_f$.

**Positive example 1** can be justified by directly applying the deductive rule in KB (see Figure 15.) No update to the initial KB has been made, so **Positive example 1** is not a new intention.

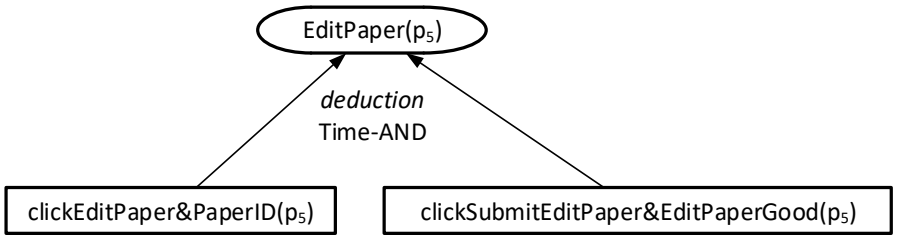


**Figure 15.** Justification of positive example 1

Next, we try to build a plausible justification tree for **Positive example 2** within the current KB. However, methods of deduction, analogy, and inductive prediction all failed, leaving the last option – method of abduction. According to [30], abduction involves two steps: generation of explanatory hypotheses and selection of the "best" hypothesis. For intention decomposition, we consider the following form of abduction:

- Hypothesizing an ISA relationship (i.e., $d_2$ isa $d_1$) if

$P(a, d_1)$ is to be explained and $P(a, d_2)$ is true, then hypothesize that $P(a, d_2) \rightarrow P(a, d_1)$

In the case of justifying **Positive example 2**, represented as $P(a, d_1)$, we can use $P(a, d_2)$ to represent the known **Positive example 1** in the initial KB. Because $P(a, d_2)$ is true, we can hypothesize the following ISA relationship (Table 5) so that $P(a, d_1)$ can be true.

Note that we have to strictly follow the temporal character of a situation sequence during the decomposition process, and that is why Time-AND joint is used to decompose the sub-sequence.

After **Positive example 2** is explained, we need to generalize KB so as to entail the new $T_f$. Figure 16 shows the generalization result, i.e., the updated KB. However, there is an empty node in the intention decomposition tree. By removing it, we can get a simplified version as shown in Figure 17. Then KB can be generalized to the form shown in Figure 18.

**Table 5.** Hypothesizing a ISA relationship

| |
|---|
| {$s_2$(EditPaper($p_1$), clickSubmitEditPaper($p_1$), Context(PaperFile($p_1$), "*Uploaded*"), Context(PaperSubmitStatus($p_1$), "*Yes*"))} |
| ISA |
| Time-AND{$s_2$(EditPaper($p_6$), clickSubmitEditPaper&NoFile($p_6$), Context(PaperFile($p_6$), "*Empty*"), Context(PaperSubmitStatus($p_6$), "*No*")), $s_3$(EditPaper($p_6$), clickSubmitEditPaper($p_6$), Context(PaperFile($p_6$), "*Uploaded*"), Context(PaperSubmitStatus($p_6$), "*Yes*"))} |

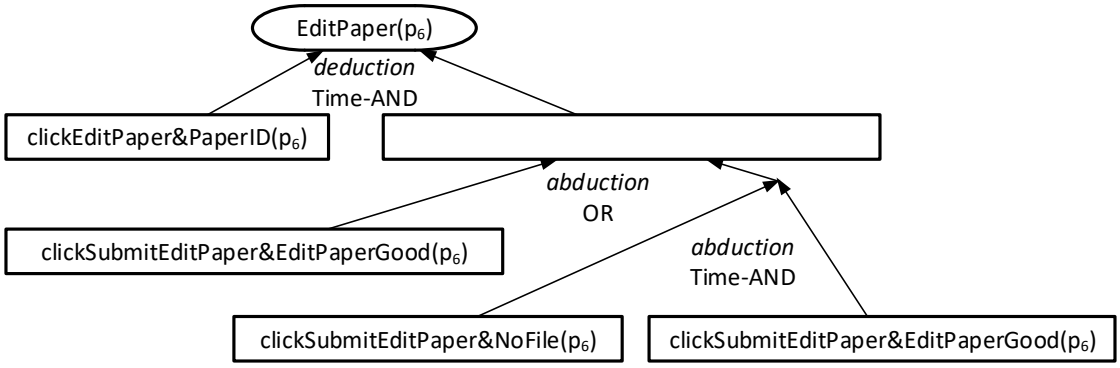


**Figure 16.** Generalization of KB after justifying positive example 2

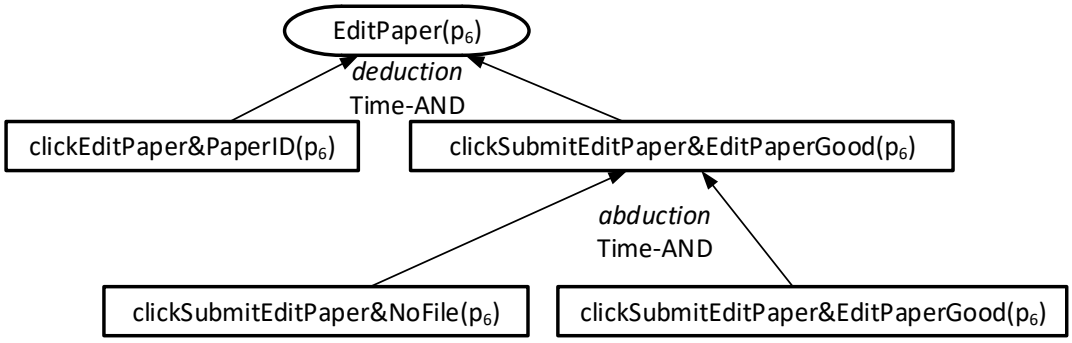


**Figure 17.** A simplified KB by removing empty node.

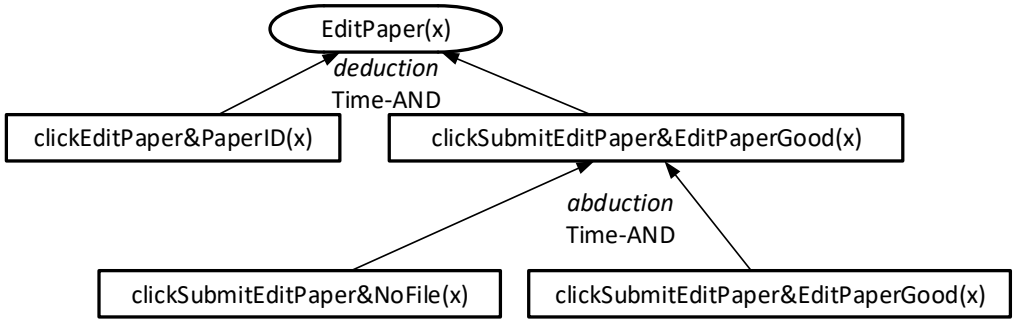


**Figure 18.** The Generalized KB

For **Positive example 3**, since it is similar to **Positive example 2** which has already been added into KB, we can make an inference through inductive prediction, shown in Table 6. Consequently, the KB is generalized as Figure 19.

**Table 6.** Making an inference through inductive prediction

**Examples from KB:**

- **Positive example 2:** Intention(EditPaper($p_6$), *Fulfilled*) = {$s_1$(EditPaper($p_6$), clickEditPaper&PaperInfo($p_6$), Context(PaperSubmitStatus($p_6$), "*No*")), $s_2$(EditPaper($p_6$), clickSubmitEditPaper&NoFile($p_6$), Context(PaperFile($p_6$), "*Empty*"), Context(PaperSubmitStatus($p_6$), "*No*")), $s_3$(EditPaper($p_6$), clickSubmitEditPaper($p_6$), Context(PaperFile($p_6$), "*Uploaded*"), Context(PaperSubmitStatus($p_6$), "*Yes*"))}

**Inductive generalization:**

{$s_1$(EditPaper($x$), clickEditPaper&PaperInfo($x$), Context(PaperSubmitStatus($x$), "*No*")), $s_2$(EditPaper($x$), clickSubmitEditPaper&NoFile($x$), Context(PaperFile($x$), "*Empty*"), Context(PaperSubmitStatus($x$), "*No*")), $s_3$(EditPaper($x$), clickSubmitEditPaper($x$), Context(PaperFile($x$), "*Uploaded*"), Context(PaperSubmitStatus($x$), "*Yes*"))} → Intention(EditPaper($x$), *Fulfilled*)

**Predicted inference:**

{$s_1$(EditPaper($p_7$), clickEditPaper&PaperInfo($p_7$), Context(PaperSubmitStatus($p_7$), "*No*")), $s_2$(EditPaper($p_7$), clickSubmitEditPaper&NoFile($p_7$), Context(PaperFile($p_7$), "*Empty*"), Context(PaperSubmitStatus($p_7$), "*No*")), $s_3$(EditPaper($p_7$), clickSubmitEditPaper($p_7$), Context(PaperFile($p_7$), "*Uploaded*"), Context(PaperSubmitStatus($p_7$), "*Yes*"))} → Intention(EditPaper($p_7$), *Fulfilled*)

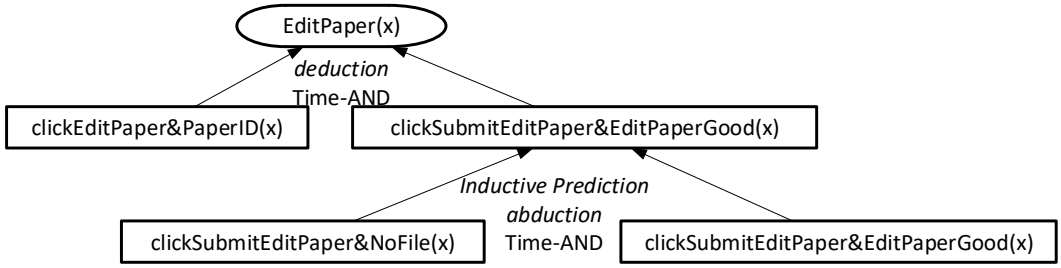


**Figure 19.** The updated KB after learning form positive example 3

The two negative examples can be justified through current KB. No unjustifiable examples can be found. So the learning process for $T_f$ is finished.

To learn $T_u$, the two negative examples can directly be justified by applying the deductive rule:

{Situation(EditPaper(*x*), clickCancelEditPaper&PaperID, Context(PaperFile, "*Empty*"), Context(PaperSubmitStatus, "*No*"))} => Intention(EditPaper(*x*), *Unfulfilled)*

Next, all positive examples can be justified by $T_u$. and there is no unjustifiable example left. So this ends the learning of $T_u$. The whole learning process is thus finished. The updated KB is shown as Figure 19, and the intention that introduced such changes is **Positive Example 2** in Table 4, i.e., the new intention.

By examining the "New Data" set acquired from the result of CRF desire inference on the first round experiment data, we get a total of 7 new positive examples and 5 new negative examples. By feeding all of them into our MTL learning engine, the resulting KB is the same as that in Figure 19, with 2 new positive examples identified as new intention, which is essentially the same as the new intention found above.

Next, we will briefly show the learning result of a more complicated KB, for intention "Upload a Paper", i.e., UploadPaper(*x*). Its initial KB is shown in Figure 20. For this intention, there are 40 new positive examples and 12 new negative examples to be justified. The learning process is the same, so we will not reiterate on the details. Figure 21 shows the updates KB structure. However, among 40 new positive examples, 33 of them are identified as new intentions. Example of new intentions are given in Figure. 22. The reason why there are so many new intentions is that the original KB contains very limited amount of knowledge. We can see that the application of OR joint significantly simplify the new KB structure, which may become a lot complicated with numerous combination of different types of error users can get. We can

also see that the application of abduction & induction methods plays a crucial role in this learning process.

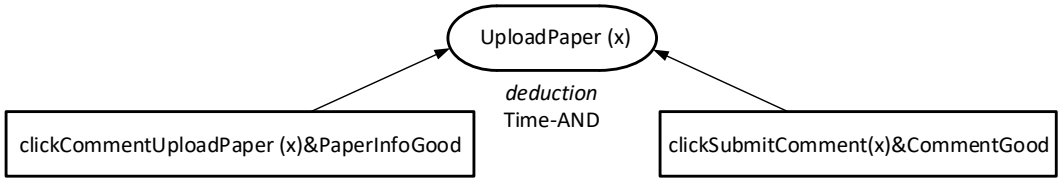


**Figure 20.** Figure Initial KB structure of intention "Upload a Paper"

| | | | |
|---|---|---|---|
| New Intention 1 | | | |
| clickMenuUploadPaper | 30s | UploadPaper | UploadPaper/0.987714 |
| clickCommentUploadPaper&NoKeywordDOI | m | UploadPaper | UploadPaper/0.979297 |
| clickCommentUploadPaper&PaperInfoGood | m | UploadPaper | UploadPaper/0.943477 |
| clickSubmitComment&NoCategoryShortMethodologyTechnicalMeritsLimitationDiscussion | m | UploadPaper | UploadPaper/0.759959 |
| clickSubmitComment&ShortMethodologyTechnicalMeritsLimitationDiscussion | 60s | UploadPaper | UploadPaper/0.340362 |
| clickSubmitComment&NoCategoryShortMethodologyTechnicalMeritsLimitationDiscussion | 20s | UploadPaper | UploadPaper/0.281002 |
| clickSubmitComment&ShortMethodology | 60s | UploadPaper | UploadPaper/0.434254 |
| clickSubmitComment&NoCategory | 20s | UploadPaper | UploadPaper/0.705801 |
| clickSubmitComment&CommentGood | 10s | UploadPaper | UploadPaper/0.743047 |
| | | | |
| New Intention 2 | | | |
| clickMenuUploadPaper | 60s | UploadPaper | UploadPaper/0.950302 |
| clickCommentUploadPaper&NoKeywordDOIFile | m | UploadPaper | UploadPaper/0.964158 |
| clickCommentUploadPaper&PaperInfoGood | m | UploadPaper | UploadPaper/0.971687 |
| clickSubmitComment&ShortMethodologyTechnicalMeritsLimitation | m | UploadPaper | UploadPaper/0.931654 |
| clickSubmitComment&NoCategory | 30s | UploadPaper | UploadPaper/0.923622 |
| clickSubmitComment&CommentGood | 10s | UploadPaper | UploadPaper/0.942890 |
| clickSubmitComment&CommentGood | 10s | UploadPaper | UploadPaper/0.869341 |

**Figure 22.** Example of New Intentions Identified for "UploadPaper"

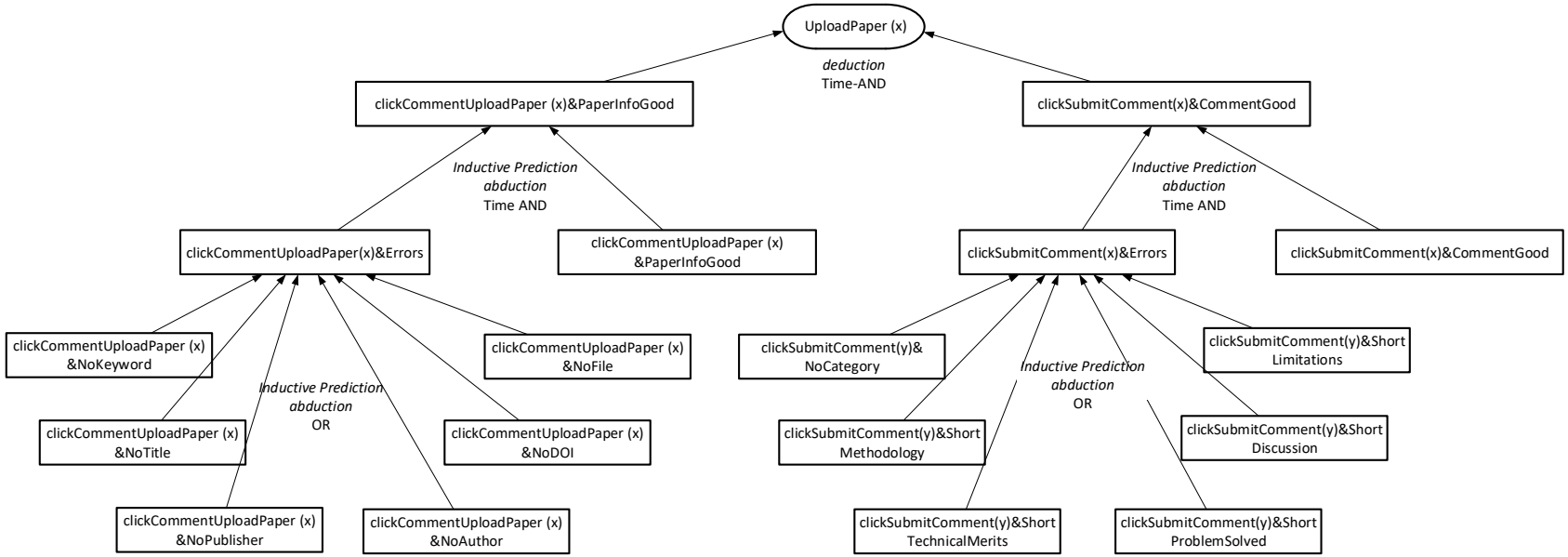


**Figure 21.** Updated KB for intention "UploadPaper(*x*)"

Table 7 summarizes the transformation process of DIKW in a quantified view. We can see that the number of data points has been significantly reduced through the whole learning process. Such trend can be seen as the effect of applying the $(MR)^2$ paradigm proposed in [40].

**Table 7.** A quantified view of the reduce process

| Raw Observations (Data) | Refined Observations (Information) | Intentions (Knowledge) | New Intentions (Wisdom) |
|---|---|---|---|
| 10,063 records<br>585 sessions | 6880 records<br>369 sessions | EditPaper | |
| | | 12 Intentions | 1 New Intention |
| | | UploadPaper | |
| | | 52 Intentions | 33 New Intentions |

# CHAPTER V

## TRANSFORMING NEW INTENTIONS TO NEW REQUIREMENTS USING STRATEGIC MODELING *i**

The ultimate goal of this work is to elicit new system requirements with limited help from human experts. So far, we are able to identify new intentions in a mostly computable fashion using the MTL method, although human input is still needed to construct the initial knowledge base and to validate the end result. However, new intentions are not equivalent to new requirements. In fact, most of the newly identified intentions are low-level tasks (in goal modeling terms), i.e., they describe how users utilize existing system functions and features to fulfill their needs in a different way from what has been anticipated. There is still a huge gap between those low-level tasks and system requirements.

To solve the above mentioned problem, we may recall the concept of DIKW Hierarchy, but with a different perspective. Figure 23 visualizes DIKW in a view of knowledge network. When relationships among randomly isolated data points are discovered, a potential information cluster is to be formed. In the CRF-DIKW view (Figure 8), this phase corresponds to the transformation from raw data to processed observation sequences. Note that at this point, the so-called “relationships” are still vague, requiring further exploration. By applying the CRF method, hidden user desires are inferred, and as a result, intentions are identified. Consequently, the “relationships” are now fully revealed, and a local information cluster is constructed. Since knowledge can be viewed as a network, linkages from the information cluster to existing knowledge network shall be established. In the CRF-DIKW view, new intentions can serve as these linkages that connect to existing domain knowledge in form of system requirements. However, to implement such linkages, an interface is needed.

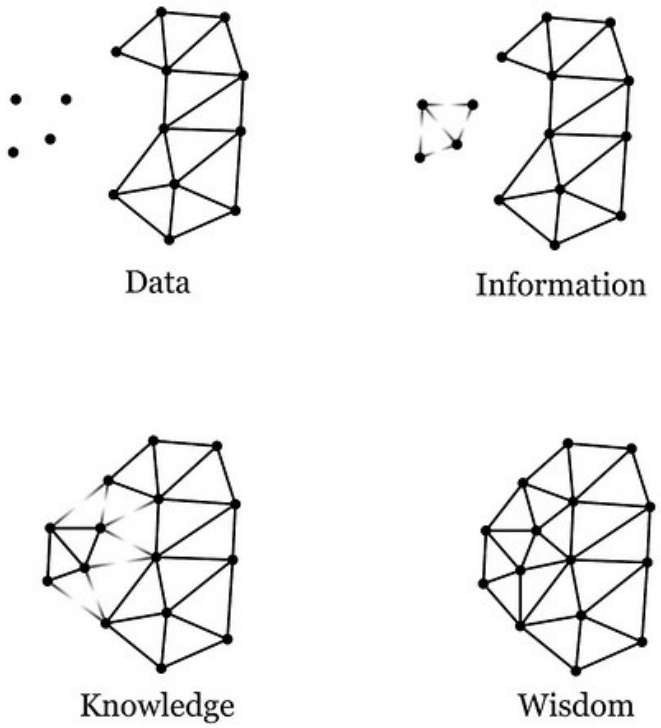


**Figure 23.** Four Tiers – a network view of DIKW

## 5.1 Means, Ends, and Beliefs

Before diving into the gap between low-level tasks and system requirements, we can first revisit the example in section 3.1.2, and informally discuss how we can make the leap. In the online shopping example shown in Figure 7, after the new knowledge is acquired, we have already known customers who buy tablets are likely to buy covers together. Such stand-alone knowledge doesn't have much application value, unless it is connected to other relevant knowledge within the existing knowledge base. In this case, relevant knowledge include a business goal "*Maximizing sales profits*", along with two business believes "*Selling more products can increase sales profits*" and "*Customers like to buy products that are easily to be located*". The underlying logic (shown in Figure 24.) can be described as follows. To maximize

profits, more products need to be sold. And it is known that customers have a habit of buying tablets and covers together. So it would be wise to add a hyperlink to cover onto the tablet webpage, so that customers can easily find covers when they are shopping for tablets, and consequently, sales profits will be increased. Putting it in another way, adding a hyperlink to cover is our mean to meet the end goal of maximizing sales profits, which is so-called "Means-Ends Reasoning" in strategic reasoning [16][42] (Figure 25).

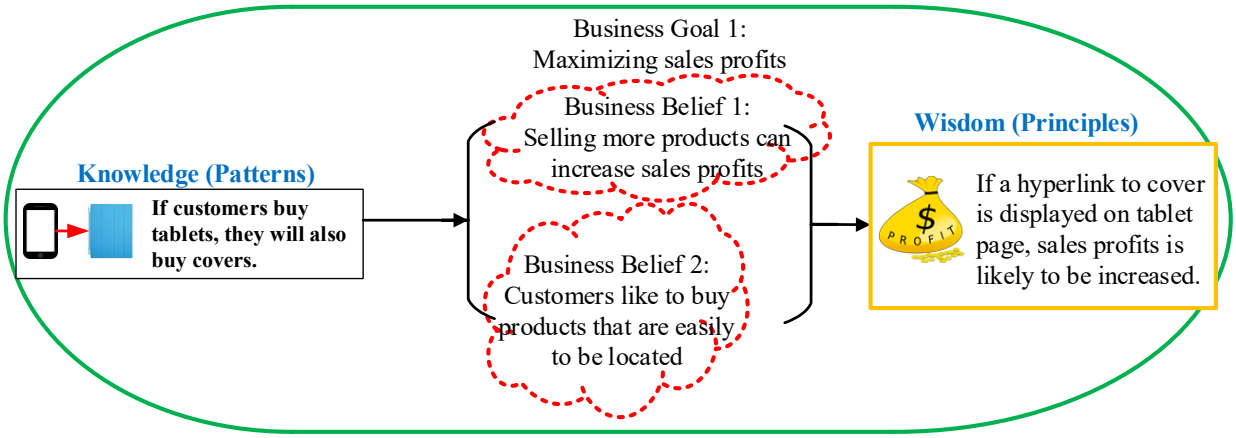


**Figure 24.** Realization of wisdom through business goal and business belief analysis – online shopping example

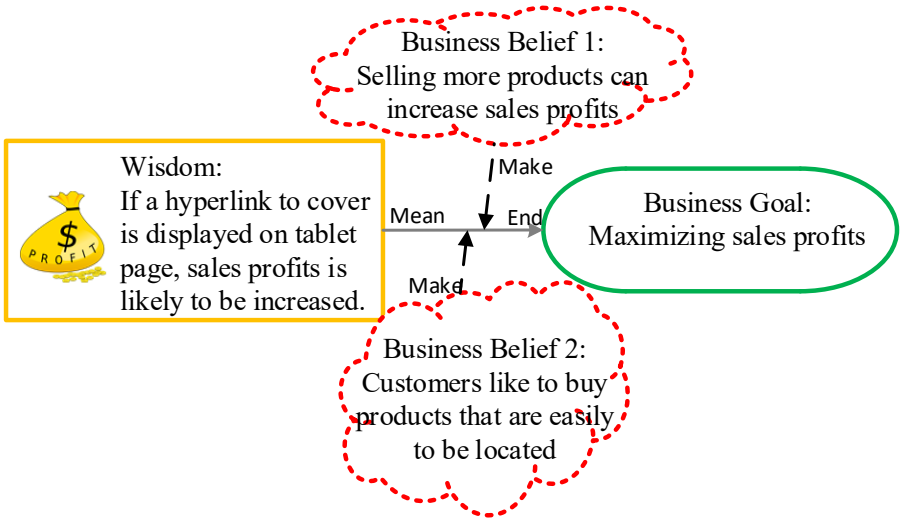


**Figure 25.** Means-Ends reasoning (1)

Similarly, the situation analytic example (shown in Figure 7) can be perceived in a similar way using system goal and design beliefs as shown in Figure 26, as well as a "Means-Ends Reasoning" (Figure 27). The underlying design rationale can be described as follows: To pursue high usability, the number of repeated operations needs to be reduced. Given the fact that users are likely to download all papers that show up in their search result, it is wise to add a compound operation "Download All" to reduce the number of repeated operation. Hence, system usability can be increased.

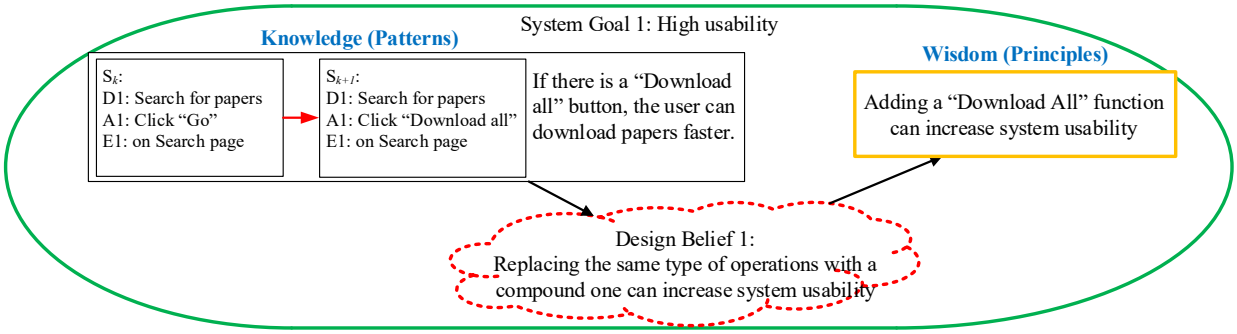


**Figure 26.** Realization of wisdom through system goal and design belief analysis – Search/Download example

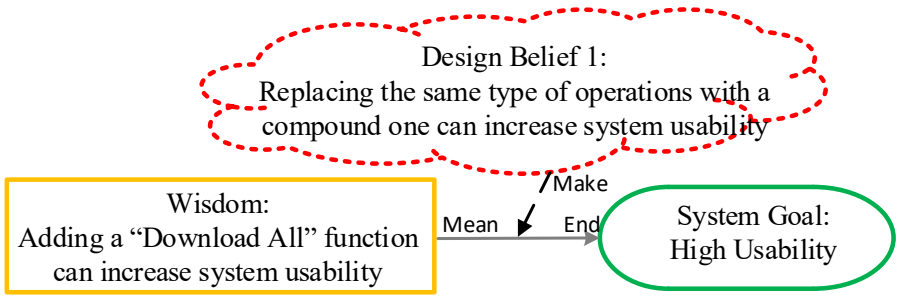


**Figure 27.** Means-Ends reasoning (2)

After breaking down the above two examples, we can see that a high-level, strategic analysis can help us fuse the new knowledge fragment with the existing knowledge network, through its interaction with design beliefs and goals.

## 5.2 A Strategic Analysis Framework – *i**

The *i** framework was proposed by Eric Yu [42] for modeling and reasoning about organizational environments and their information systems. It aims to model and analyze stakeholder interests and how they might be addressed, or compromised, by various system-and-environment alternatives. The framework builds on a knowledge representation approach to information system development [43]. It has also been applied to business process modeling and redesign [44] and to software process modeling [45]. In 2008, the *i** framework was approved as an international standard by International Telecommunication Union (ITU).

The central concept in *i** is that of the intentional actor [46]. Organizational actors are viewed as having intentional properties such as goals, beliefs, abilities, and commitments. Actors depend on each other for goals to be achieved, tasks to be performed, and resources to be furnished. By depending on others, an actor may be able to achieve goals that are difficult or impossible to achieve on its own. On the other hand, an actor becomes vulnerable if the depended-on actors do not deliver. Actors are strategic in the sense that they are concerned about opportunities and vulnerabilities, and seek rearrangements of their environments that would better serve their interests. The *i** framework consists of two main modeling components. The Strategic Dependency (SD) model is used to describe the dependency relationships among various actors in an organizational context. The Strategic Rationale (SR) model is used to describe stakeholder interests and concerns, and how they might be addressed by various configurations of systems and environments.

The reasons why we believe the *i** framework is able to help with our knowledge fusion problem are as follows.

1. It can help answer the question “Why”, which is exactly what needs to be addressed in the layer of “Wisdom” in the DIKW Hierarchy in Figure 6.
2. Strategic Rationale (SR) model includes both low-level tasks and high-level hard goals or soft goals.
3. Implicit design beliefs and rationale can be modeled and incorporated into SR analysis.
4. Existing tools, such as OME (Open Modeling Environment), which supports graphical model representation and automated reasoning, can help facilitate and automate our problem solving process.

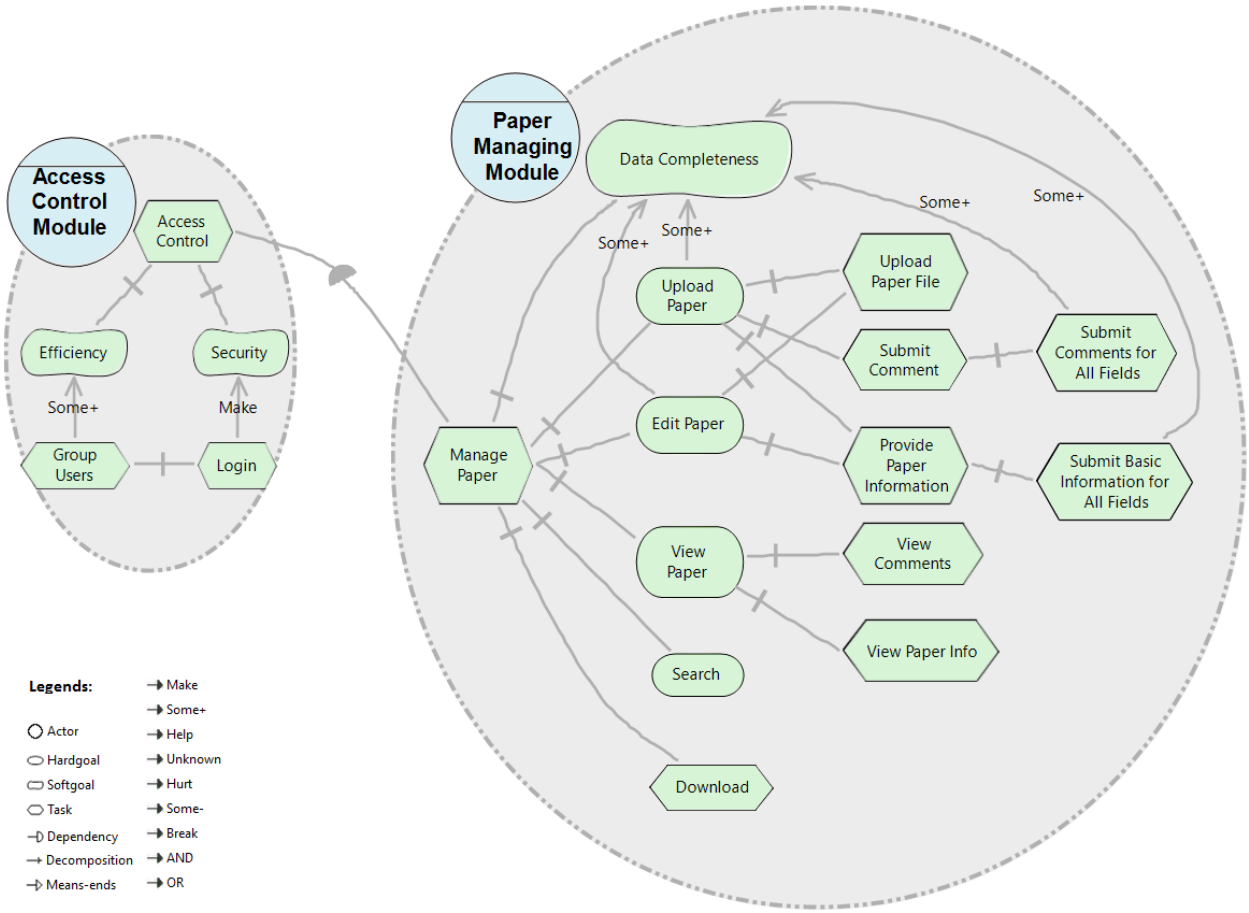


**Figure 28.** The SR model of access control and paper managing modules in CoRE

Figure 28 shows a fragment of SR model of the CoRE system (for 1st round experiment), including all soft goals, hard goals, tasks, and resources that are relevant to the two examples discussed in section 4.2. High-level system requirements are modeled through goal decomposition. Thus, we consider using this SR model as our existing domain knowledge network.

## 5.3 Knowledge Fusion using *i**

To establish connection to the existing SR model, we need to transform our new knowledge into the form of an SR model. We take the updated KB of intention "EditPaper(*x*)" as an example. To make it easier to understand, we represent the updated KB with Figure 29 using an explicit "OR" joint connecting two low-level alternatives. One alternative (Alt 1) is to upload paper information together with a paper file, which is the original design of CoRE; the other one (Alt 2) is to upload paper information alone without any paper file. These alternatives can be modeled as tasks in SR model (see Figure 30).

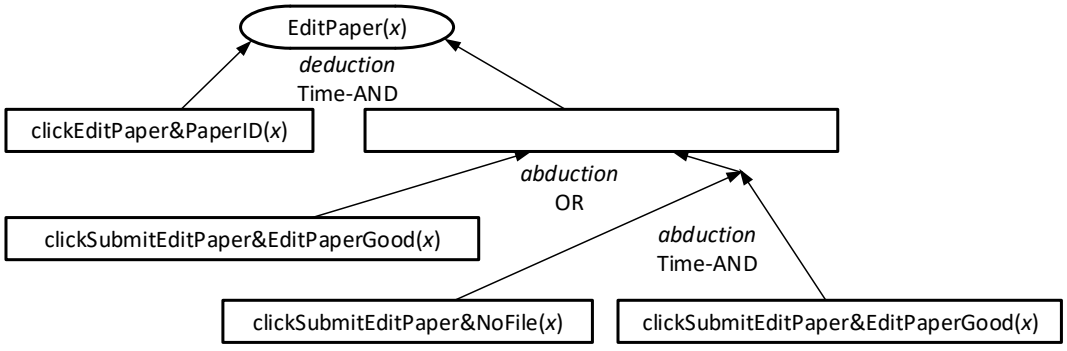


**Figure 29.** Updated KB for intention "EditPaper(*x*)"

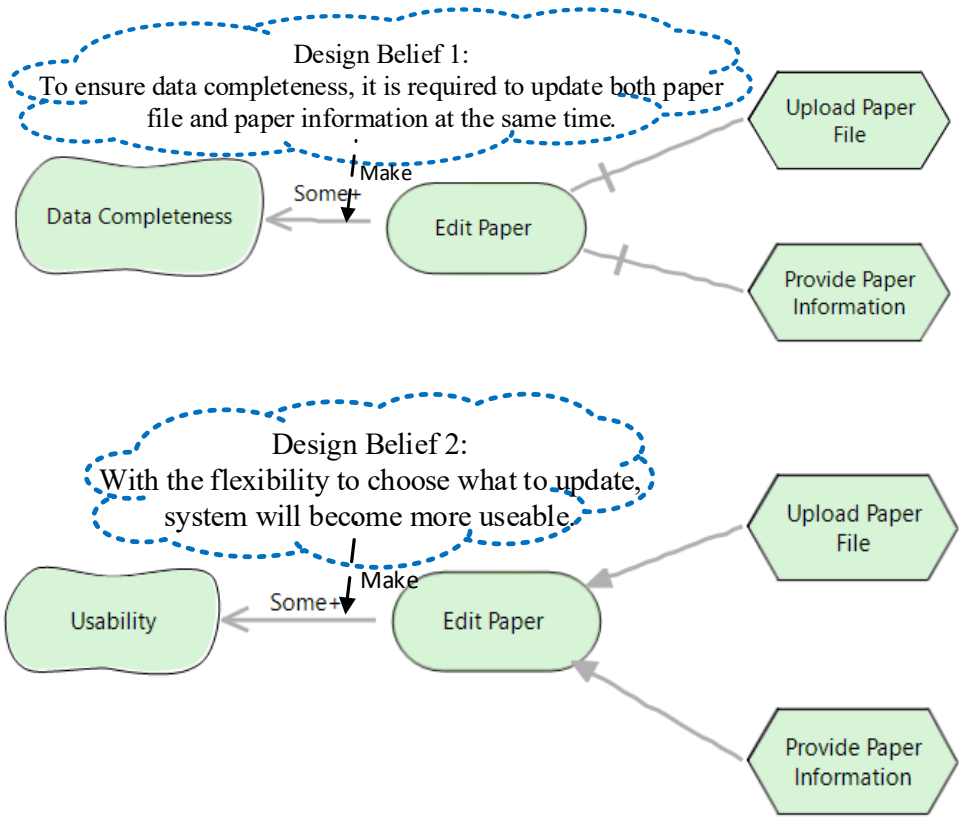


**Figure 30.** SR models for two alternatives

From the above SR models for Alt 1 and 2, we can see that Alt 1 mainly focuses on satisfying "Data Completeness", while Alt 2 emphasized on "Usability". By merging them together in Figure 31, we can see that two alternatives have different impact on these soft goals. At this point, the pros and cons for each alternative are presented to a system designer, who can make a trade-off decision based on the updated knowledge. In the knowledge fusion perspective, new knowledge, in form of design alternative, has been fused with existing knowledge network.

In a similar way, the updated knowledge base of intention "UploadPaper($x$)" can be represented with an SR model, and further fused with existing knowledge as shown in Figure 32. Consequently, by adding these updated small SR models to the system-level model in Figure 28, we can get an updated knowledge network shown as Figure 33.

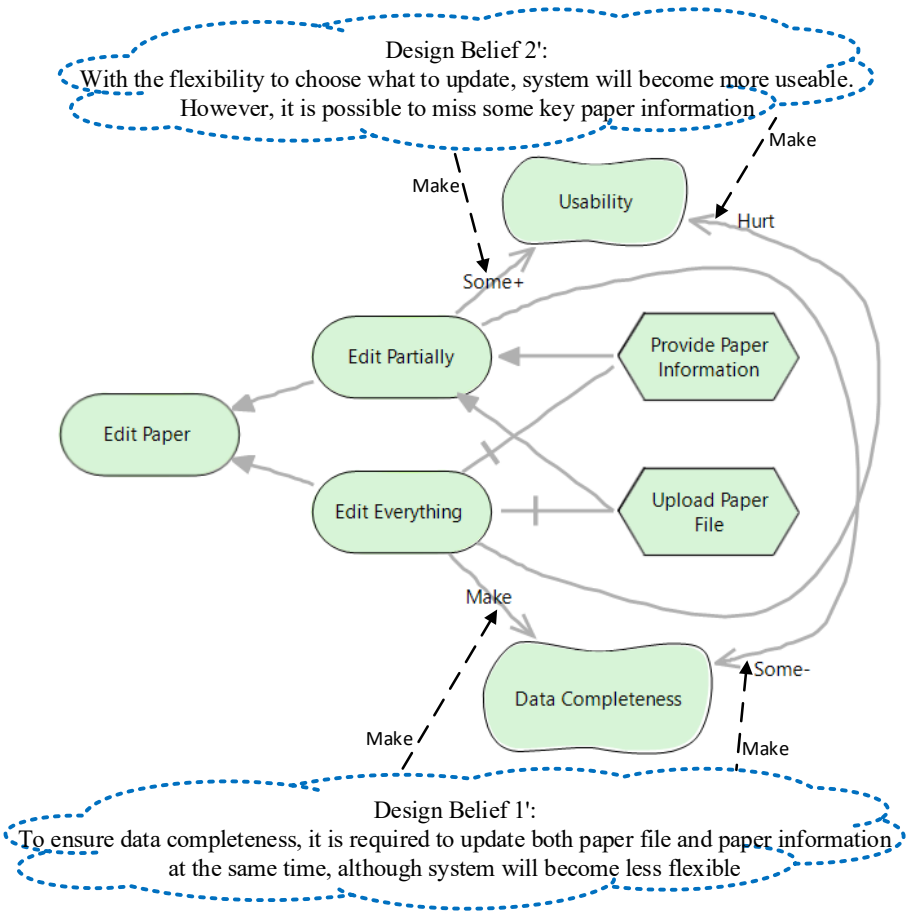


**Figure 31.** Merging two alternatives

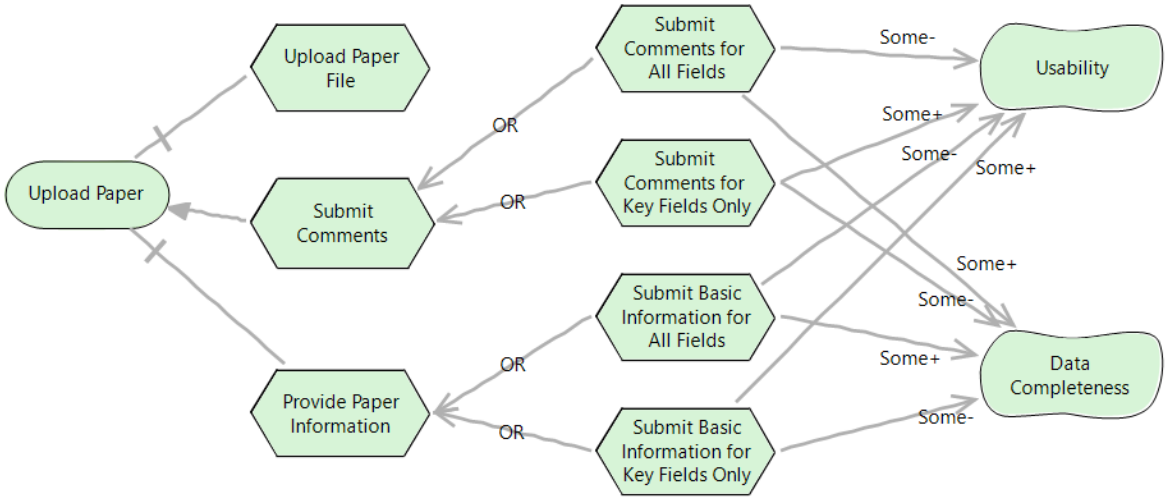


**Figure 32.** Alternatives for intention "UploadPaper(*x*)"

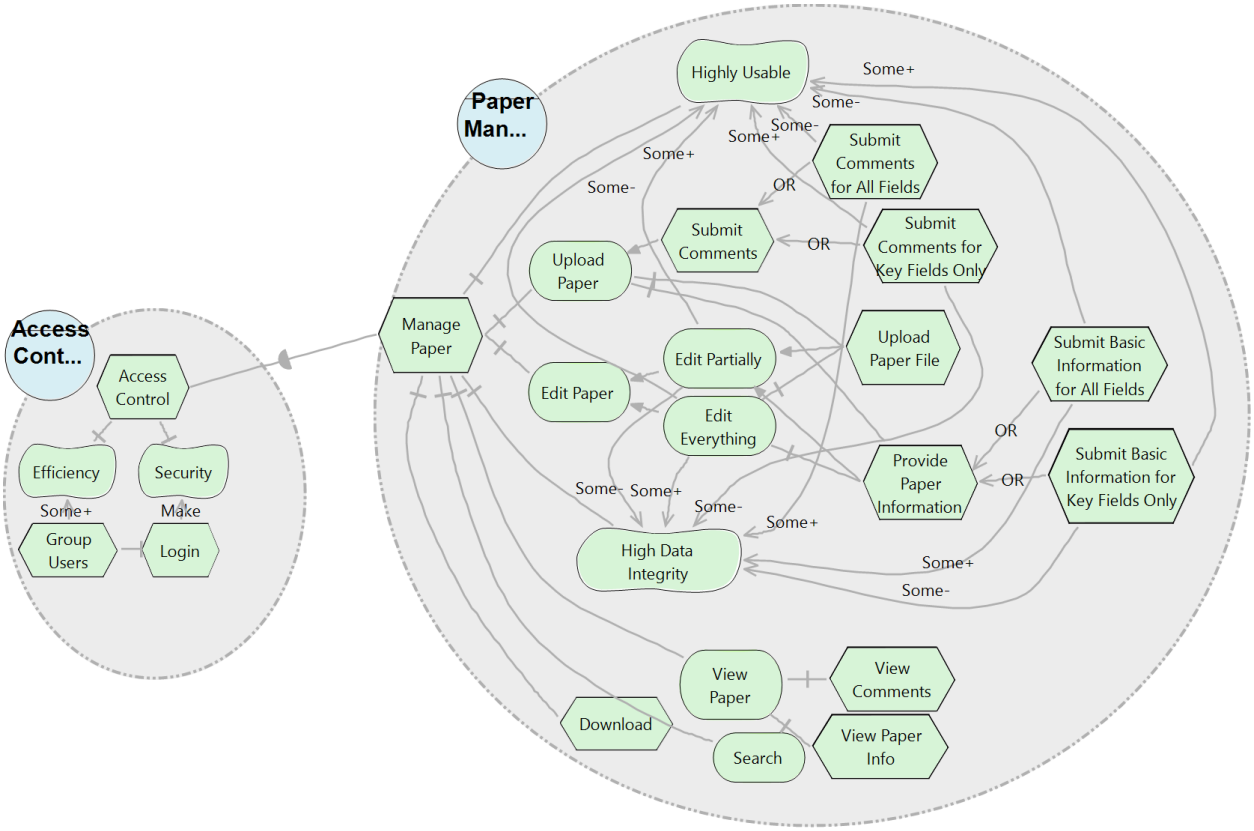


**Figure 33.** The updated system-level SR model (knowledge network)

One of the advantage of OME is that when prioritizing the satisfaction of certain hard goal or soft goal, it can automatically choose appropriate design alternatives using backward or forward evaluation function. In our case, for CoRE version II, we want to emphasize our design on “Usability”. By prioritizing soft goal “Highly Usable”, OME can generate a view shown in Figure 34, with recommended design alternatives checked, and contradictory design elements crossed out. Also, as part of the evaluation result, OME is able to show the influence/impact of prioritizing certain high-level goals. In this case, when “Usability” is prioritized, a side effect is that “Data Completeness” will be undermined. Such contradictory scenario is not uncommon when trade-off between different high-level system goals during early design analysis. A

decision has to be made on which high-level goal shall be prioritized. In our case study, we implemented CoRE version II according to the evaluation result shown in Figure 34.

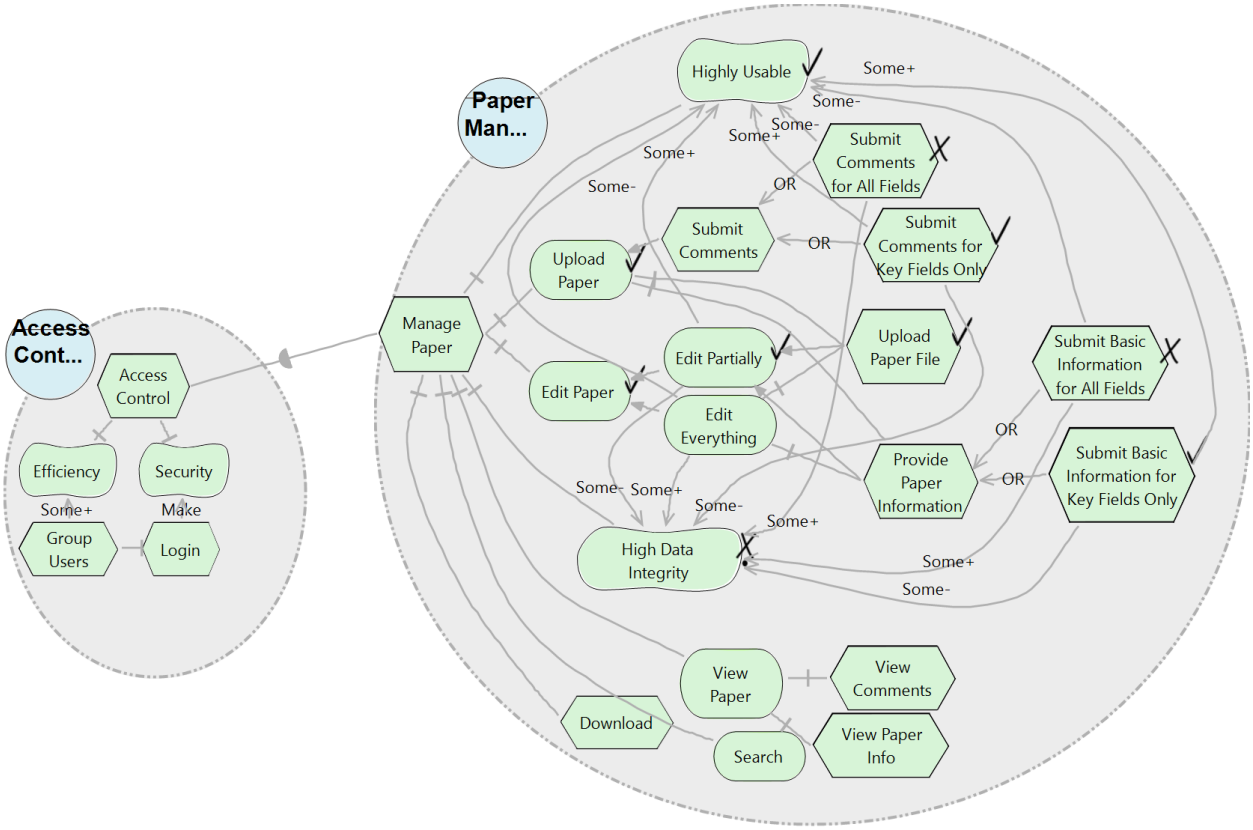


**Figure 34.** Evaluation result of "Highly Usable"

To sum up, new intentions can be transformed into new design wisdom through the following steps:

1. Elicit design alternatives from updated KB.
2. Build a local SR model for each design alternative, and show its contribution to related high-level goals.
3. Add all local SR models into the SR model of the existing system.
4. Prioritize certain high-level goal(s), then carry out forward/backward evaluation.

5. Based on the result of step 4, summarize a design plan for the evolved system.

## 5.4 Evaluation of New Design Alternatives

After implementing CoRE Version II based on Figure 34, to evaluate the effect of new design alternatives, a second-round experiment was done on the evolved system – Core Version II, by following exactly the same process as what we did in the first round. The only difference was that we recruited 40 new participants in the second round to make the comparison between two versions fair and even. In this section, we will not elaborate the technical execution of our methodology steps due to space limitations. Instead, we will mainly focus on evaluating the improvement of CoRE version II over its predecessor.

The difference between CoRE Version I and II are that users are allowed to edit a paper without uploading a file, and to upload a paper with only some key information submitted and without submitting comments too. Table 8 shows a comparison of user performance between two systems. As we can see, the success rate of two tasks both increased in the CoRE Version II, while the time cost and the error occurrence rate, especially the consecutive error occurrence rate, significantly decreased.

Although there is no absolutely right or wrong design, based on statistics shown in Table 8, we can get a sense on which alternative is in favor of users. We can say that, by fusing the SR models of updated knowledge base with that of the existing system (CoRE I), we recognize the needs to shift our design principle from Data-Completeness to Usability. And statistics shows that it is a wise move.

**Table 8.** Comparison of user performance between two rounds

| *Task* | *Version* | *# of Occurrence* | *Avg. Time Spent (s)* | | *Successful* | | *Errors*[a] | | *Consecutive Errors*[b] | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Time | Improvement | # (rate) | Improvement | # (rate) | Improvement | # (rate) | Improvement |
| Upload Paper | I | 55 | 387.87 | ↓55% (Reduced) | 38 (69%) | ↑41% (Increased) | 28 (51%) | ↓94% (Reduced) | 14 (25%) | ↓100% (Reduced) |
| | II | 32 | 173.57 | | 31 (97%) | | 1 (3%) | | 0 | |
| Edit Paper | I | 16 | 47.81 | ↓1% | 8 (50%) | ↑100% | 5 (31%) | ↓100% | 0 | – |
| | II | 9 | 47.33 | | 9 (100%) | | 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*

# CHAPTER VI

# DISCUSSION AND CONCLUSION

## 6.1 Threats to Validity

First of all, this work is based on the desire inference result using the CRF method in [39]. Thus, it naturally inherit the potential threats to validity discussed in [39]. In this section, we will only discuss the potential threats to the validity of our proposed method in Chapter III and V.

### 6.1.1 Threats to construct validity

The theoretical foundation of our work is the DIKW Hierarchy. Due to lack of a uniform definition of information and knowledge, it is still hard to set boundaries between information and knowledge, or to clearly define what wisdom is. Thus, our work may suffer from naming confusion of different layers in a DIKW Hierarchy. However, when dealing with specific domain, it is much easier to give a domain-specific definitions for each DIKW layer. Or put it another way, our goal is not to define those layers, but to adopt a layered approach to divide the big problem into smaller pieces, and to apply a $(MR)^2$ paradigm [40]. Still, one can argue the importance to set clear-cut boundaries before further proceeding.

Another potential concern is regarding the application of the MTL method, which was initially proposed for concept learning. Although we have seen research work using it on requirements elicitation [31], further evaluation on large data set is still needed to validate our meta-model (section 3.3). Also, it is also desired to test it on other domains, before finalizing the definition of the meta-model.

### 6.1.2 Threats to internal validity

The preparation of initial KB might involve our subjective understanding of the domain and data. Thus, others may create a different initial KB, and may experience a different learning process and results. On the other hand, our method only requires partially complete initial KB and at least one positive example. So in theory, it should be able to tolerate certain variations on initial KB and different inputs. Yet, its threshold has not been tested.

The application of SR model also requires analyst's expertise on strategic modeling and proficient understanding of design knowledge. At current stage, it is still very difficult to give any quantified evaluation regarding the process of transformation from new knowledge to wisdom.

### 6.1.3 Threats to external validity

So far, we have only worked on the domain of CoRE, although we believe that CoRE shares many features with other systems. Some example systems are mobile applications for wearable devices, home automation system, and the server end of dynamic web-based systems. One challenge is that the accuracy of desire inference using the CRF method needs to be further validated on other domains, since the quality of the initial KB largely depends on that.

## 6.2 Conclusion

In this work, we explored the feasibility of bridging the gap between data and requirements using *Situ* framework [8] as the foundation. The proposed approach is also an implementation of the concept of DIKW Hierarchy, in which techniques such as the CRF

method, the MTL method, and SR analysis has been applied. Figure 35 summarizes our logical workflow of this work. In the case study, we worked through one of the most immediate goals that is to identify new intentions. We further researched on using strategic modeling framework *i** to fuse new knowledge with existing knowledge network. Although we experienced a positive outcome, the current result is still primitive since the type of new intentions/requirements that can be discovered are mostly low-level design alternatives, i.e. new ways of fulfilling a certain anticipated intention. New requirements in the real world are much more diverse. Plus, when a large number of alternatives show up in the KB, some may not make much practical sense. Thus, learning shall be restricted to certain domain constraints, which is an important part of our future work.

Our ultimate goal is to automate the process of new requirements elicitation as much as possible. However, at current stage, human oracle is still needed at the very end of the MTL learning process, in order to filter out meaningful newly identified intentions. Based on our experience, the level of help needed mostly depends on the quality of the output from MTL learning engine, which further depends on several factors including the quality of the initial KB, noise level in the input (new positive & negative examples), availability of domain knowledge (specified in form of constraint), etc. Also, the knowledge fusion process is still in a semi-automated stage. Manual modeling is still needed to transform new intentions into SR models. This is due to the lack of communicating interface between two frameworks, *Situ* vs *i**. To automate this transformation process, one option is to develop a special ontology to map basic elements in two frameworks.

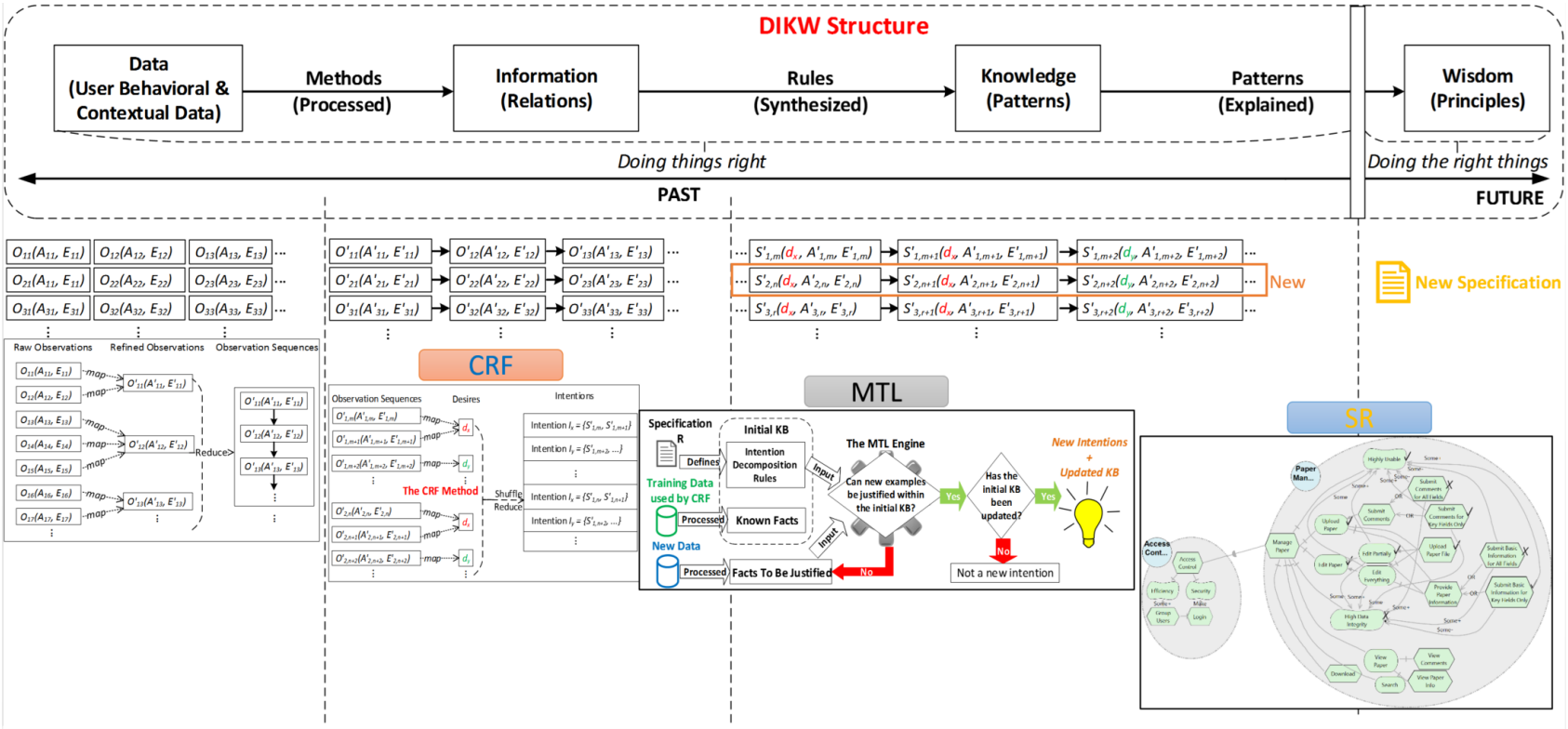


**Figure 35.** The logical workflow of applying CRF, MTL, and SR within the DIKW Hierarchy

This work can be further extended in the following directions:

1. Testing our approach on a large data set in a controlled experiment setting. Also, carrying out performance evaluation between MTL and other knowledge mining methods.
2. Extending our approach to improve the elicitation of non-functional requirements, for example, usability. We believe that with the help of historical data, fitness criteria could be better specified to meet individual users' needs in each specific situation.
3. Extending current *Situ* framework to support non-functional requirements analysis, so that new knowledge can be fused with existing knowledge network seamlessly within *Situ*.

# REFERENCES

[1] D. Zowghi and C. Coulin, “Requirements Elicitation: A Survey of Techniques, Approaches, and Tools”, Engineering and Managing Software Requirements, pp.19-46, 2005.

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

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

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

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

[6] 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.

[7] 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.

[8] C. K. Chang, H. Jiang, H. Ming, and K. Oyama, “Situ: A Situation-Theoretic Approach to Context-Aware Service Evolution”, IEEE Transactions on Services Computing, 2(3), pp. 261-275, 2009.

[9] R.L. Ackoff, “From data to wisdom”, Journal of Applied Systems Analysis, 16, pp. 3-9, 1989.

[10] J. Rowley, “The wisdom hierarchy: representations of the DIKW Hierarchy”, Journal of Information Science, 33(2), pp. 163-180, 2007.

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

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

[13] T. Keller, “Contextual Requirements Elicitation - An Overview”, Seminar in Requirements Engineering, Department of Informatics, University of Zurich, 2011.

[14] G.E. Kniesel and R.E. Filman, “Unanticipated Software Evolution”, Journal of Software Maintenance and Evolution: Research and Practice, 17(5), pp. 307-377, 2005.

[15] A. Dardenne, A. van Lamsweerde, and S. Fickas, “Goaldirected requirements acquisition”, Science of Computer Programming, 20(1-2), pp. 3-50, 1993.

[16] E. Yu and J. Mylopoulos, “Understanding “why” in software process modelling, analysis, and design”, Proc. 16th International Conference on Software Engineering (ICSE’94), pp. 159-168, 1994,.

[17] J. Mylopoulos, L. Chung, S. Liao, H. Wang, and E. Yu, “Exploring alternatives during requirements analysis”, IEEE Software, 18(1), pp. 92-96, 2001.

[18] B. Hui, S. Liaskos, and J. Mylopoulos, “Requirements analysis for customizable software: A goals-skills-preferences framework”, Proc. 11th IEEE International Requirements Engineering Conference (RE’03), pp. 117-126, 2003,.

[19] S. Liaskos, A. Lapouchnian, Y. Wang, Y. Yu, and S. Easterbrook, “Configuring common personal software: a requirements-driven approach”, Proc. 13th IEEE International Conference on Requirements Engineering (RE’05), pp. 9-18, 2005.

[20] A. Spink and C. Cole, “A human information behaviour approach to a philosophy of information”, Library Trends, 52(3), pp. 617-628, 2004.

[21] K. Herold, “The philosophy of information: introduction”, Library Trends. 52(3), pp. 373-376, 2004.

[22] L. Floridi, “Open problems in the philosophy of information”, Metaphilosophy, 35(4), pp. 554-582, 2004.

[23] J. Rowley, “What is information?”, Information Services & Use, 18(4), pp. 243-254, 1998.

[24] M. Tomasello, M. Carpenter, J. Call, T. Behne, and H. Moll, “Understanding and Sharing Intentions: The Origins of Cultural Cognition”, Behavioral and Brain Sciences, 28(5), pp. 675-691, 2005.

[25] P.R. Cohen and H.J. Levesque, “Intention Is Choice with Commitment”, Artificial Intelligence, 42(2-3), pp. 213-261, 1990.

[26] A.S. Rao and M.P. Georgeff, “Modeling Rational Agents within a BDI-Architecture”, Proc. 2nd International Conference on Principles of Knowledge Representation and Reasoning (KR), pp. 473-484, 1991.

[27] D. Hernando, V. Crespi, and G. Cybenko, “Efficient Computation of the Hidden Markov Model Entropy for a Given Observation Sequence”, IEEE Trans. Information Theory, 51(7), pp. 2681-2685, July 2005.

[28] G. Tecuci and R.S. Michalski, “A method for multistrategy task-adaptive learning based on plausible justifications”, Proc. 8th International Workshop on Machine Learning (1991a), pp. 549-553, 1991.

[29] R.S. Michalski, (1993), “Inferential learning theory as a conceptual basis for multistrategy learning”, Machine Learning, 11(2), pp. 111-151, 1993.

[30] G. Tecuci, “Plausible justification trees: A framework for deep and dynamic integration of learning strategies”, Machine Learning, 11(2), 237-261, 1993.

[31] R. Ankori, “Automatic requirements elicitation in agile processes”, Proc. the IEEE International Conference on Software - Science, Technology & Engineering (SwSTE’05), pp 101-109, 2005

[32] H. Xie, C. K. Chang, H. Ming, and K. Lu, “The Concepts and Ontology of SiSL: A Situation-Centric Specification Language”, Proc. 36th Annual IEEE International Computer Software and Applications Conference Workshops (COMPSACW'12), pp. 301-307, 2012.

[33] J. Yang and L. Liu, “Modelling Requirements Patterns with a Goal and PF Integrated Analysis Approach”, Proc. 32nd Annual IEEE International Computer Software and Applications Conference (COMPSAC 2008), pp 239-246, 2008.

[34] G. Browne and M. Rogich, “An Empirical Investigation of User Requirements Elicitation: Comparing the Effectiveness of Prompting Techniques”, Journal of Management Information System, 17(4), pp. 223-249, 2001

[35] R. Schumaker, “From Data to Wisdom: The Progression of Computational Learning in Text Mining”, Communications of the IIMA, 11(1), pp. 38-48, 2011

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

[37] P. Rigaux, The MyReview System, retrieved from http://myreview.sourceforge.net.

[38] H. Xie and C. K. Chang, “Detection of New Intentions from Users Using the CRF Method for Software Service Evolution in Context-Aware Environments”, Proc. 39th Annual IEEE

International Computer Software and Applications Conference (COMPSAC 2015), pp. 71-76, 2015.

[39] H. Xie, J. Yang, C. K. Chang, and L. Liu, "A Statistical Analysis Approach to Predict User's Changing Requirements for Software Service Evolution", The Journal of Systems & Software, 132C, pp. 147-164, 2017.

[40] H. Ming, C. K. Chang, and J. Yang, "Dimensional Situation Analytics: from Data to Wisdom", Proc. 39th Annual IEEE International Computer Software and Applications Conference (COMPSAC 2015), pp. 50-59, 2015.

[41] International Conference on Software Engineering & Knowledge Engineering, https://ksiresearchorg.ipage.com/seke/seke17.html

[42] E. Yu, "Towards Modeling and Reasoning Support for Early-Phase Requirements Engineering", Proc. 4th IEEE International Requirements Engineering Conference (RE'97), pp. 226-235, 1997.

[43] J. Mylopoulos, A. Borgida, M.Jarke, and M.Koubarakis, "Telos: Representing Knowledge about Information Systems", ACM Transactions on Information Systems (TOIS), 8 (4), 1991.

[44] E. Yu and J. Mylopoulos, "From E-R to 'A-R' – Modelling Strategic Actor Relationships for Business Process Reengineering", Proc. 13th International Conference on the Entity-Relationship Approach (ER'94), pp. 548-565, 1994.

[45] E. Yu and J. Mylopoulos, "Understanding 'Why' in Software Process Modelling, Analysis, and Design", Proc. 16th International Conference on Software Engineering (ICSE'94), pp. 159-168, 1994.

[46] E. Yu, "Modelling Organizations for Information Systems Requirements Engineering", Proc. 1st IEEE International Requirements Engineering Conference (RE'93), pp. 34-41, 1993.

[47] A. Dardenne, A. Van Lamsweerde, and S. Fickas, "Goal-directed Requirements Acquisition", Proc. 6th International Workshop on Software Specification and Design (IWSSD '93), pp. 3-50, 1993.

[48] J. Yang, C. K. Chang, and H. Ming, "A Situation-Centric Approach to Identifying New User Intentions using the MTL Method", Proc. 41st Annual IEEE International Computer Software and Applications Conference (COMPSAC 2017), pp. 347-356, 2017.