Abstract

In recent years, the rapid growth of m-commerce is being driven by various emerging mobile applications and services. Among them, mobile information service is considered to be one of important applications. User’s situation is a key factor in the design of adaptive mobile information services. Personalized and user’s situation dependent information services have been focused as one of value-added services. But little attention has been given to the point on implementing situation dependent information services based on a formal logical approach. In this research, the concepts of our logical architecture, which adopt logical reasoning capabilities of the situation calculus, enable users to receive situation dependent information services. Moreover, because a user’s situation can never exist in isolation from his/her surrounding world, one important purpose of this research is to relate evolution of surrounding circumstances to the architecture, and to model a situation dependent information service system with surrounding awareness.

1. Introduction

Thanks to its unique attributes such as mobility, personalization, instant connectivity, and so on, m-commerce creates a huge market potential. In general, m-commerce (also called mobile commerce or mobile e-commerce) refers to “any transaction with monetary value that is conducted via a mobile network” [1]. There are many diverse m-commerce applications, various classes of these applications have been identified and defined [2, 3]. Among them, one of important applications is mobile information service.

Issue about what information service is a user desired or interested in is a critical point concerning the implementation of accepted mobile information services. The information service will not be accepted if per se is not perceived as useful [4]. In order to deliver accepted and useful information services to users, a key factor is to get to know more about the current situation of user in the face of constantly changing environment and build a one to one relationship with the user. Such a relationship is expected to optimize the information and services presented to users [5], and to become an additional source of revenue opportunity in the m-commerce value chain.

Many researchers are paying increasing attention to utilizing user’s situation as one of key determinants to deliver high-quality service options or information. There are various different terms used to describe such services, such as situation dependent services [6, 7, 8], context aware services [9, 10], or context sensitive services [11], etc. Understanding a user’s situation is important for us to develop such services. In recent years many related academic studies arise from the different research interests, such as, Figge conceives user’s situation as three dimensions composed of user identity, access position, and access time, and then uses these dimensions to adapt mobile services [7]. Amberg and Wehrmann categorize the measurable aspects of a user’s situation according to four dimensions: time, position, static and dynamic profile [8]. Followed by these various classifications, in general, a service can be regarded as situation dependent or context aware if it uses situation or context information when providing relevant information and/or services to a user. But in most previous studies, their descriptions and explanations mainly focus on the aspects of conceptual and organizational thinking. Little attention has been given to the point on implementing situation or context aware services based on a formal logical approach.

We presented a logical approach that is built on the situation calculus and Java application, and proposed an original architecture for implementing users’ situation dependent service system in one previous study [12]. However, a user’s situation can never exist in isolation from his/her surrounding world. Both
user’s world and his/her surrounding world interact, evolution of surrounding circumstances will affect user’s behavior. Therefore in this research, we attempt to relate evolution of surrounding circumstances to the proposed logical architecture, and to model and implement a situation dependent information services system with surrounding awareness (SDIS) that not just dynamically adapts to users’ situations, but also responds to evolution of their surrounding circumstances. We consider that a user’s surrounding circumstance does not directly belong to the user’s situation but may contribute to identify the user’s behavior, therefore, in this study we distinguish user’s situation from his/her surrounding circumstance. The concept of situation here represents user’s situation, which indicates factors including both user’s personal subjective aspects, such as user’s preference or profile, and objective aspects, such as user’s position, access time and so on. Meanwhile user’s surrounding circumstance represents external factors relevant to the user’s situation. For example, for a businessman who wants to book a hotel for the duration of his business travel, surrounding available hotels will become one of his external factors. The price, comfort and convenience of a hotel will have a direct impact on his later choice. Through connecting a user’s situation, including the user’s personal factors and objective factors, with the related external factors, more adaptive information services will be expected to fill the user’s needs.

The outline of this paper is as follows. First, we briefly describe research approach addressing this study. Then after our logical architecture, we turn to a discussion about a simple case for illustrating the implementation of the architecture. Finally we draw conclusions and attend to areas for future work.

2. Research approach

Generally, as compared to empirical or informal approaches based on experience and argument, formal approaches have the advantage of offering analytical methods relying on mathematical/logical models and precise rules of inference. Among them, in comparison to other ways, logical-based approaches are to provide a higher level formalism and tools for more powerful inferences, especially in a computer system. One of the benefits of the logical approaches is to offer a more intuitive perception of concepts. Such approaches are being increasingly applied to many areas involving information engineering and computer science as a comprehensive methodology. Accordingly, in order to make sure our SDIS can be specified and analyzed more precisely, taking a logical based approach is regarded as a significant means in this research.

Furthermore, the next important thing to be considered is to select what kind of logical formalism we would like to use. Because of the changing situations of a user and continuously evolving surrounding circumstance, SDIS can be considered as a dynamic evolving system. There are various ways to represent a dynamically changing world, such as modal logic, event-oriented approach, and so on. Among them, a logical language called the situation calculus, which is introduced by McCarthy [13] and developed by Reiter [14, 15, 16], is one of the most simple and powerful ways to do so [17]. The situation calculus is one of the oldest and the most used formalism for knowledge representation and reasoning in dynamically evolving domains [18]. Its expressive richness allows the integration of various aspects of dynamic systems in a unique formalism [19]. One of the main advantages of the formal method is that it can be used to analyze dynamic systems in a formal way so that we are able to verify formally that each task is achieved based on some logical rules. In addition, as a theoretical and computational foundation, the situation calculus has the pragmatic advantage that it allows us to fully model dynamic systems and implement them. In the general sense, a dynamically evolving system can be regarded as any system or set of processes that evolves over time. Such systems can range from relatively simple to extremely complex one. Therefore the formalism of situation calculus is regarded as an appropriate research approach for the SDIS domain.

We attempted to adopt the situation calculus, Golog [20] and Prolog programming language to develop a logical architecture for implementing users’ situation dependent service system. However, a system that interacts with the external world is unable to be fully dealt with only using the Golog language. We need an extending function from Golog to address the interactions between a system and its external world. This aspect is the so-called reactive behavior. Some studies revealed that the natural way to model such reactive behavior is with interrupts and interleaving concurrency [15], which can be described by the Reactive Golog (RGolog) evaluation mechanism. The RGolog language is an advanced logic programming language, which inherits the basic elements of the situation calculus, extends it with procedures and rules, and allows the modeling of more complex behaviors [21]. An RGolog interpreter is easy to implement under a Prolog interpreter and makes available all the capabilities of the situation calculus [22]. In the RGolog, usually there are two types of actions: primitive actions and exogenous actions. In order to realize an SDIS that faces influences from the
surrounding world, the RGolog rules are necessary to allow the interactions describing how the SDIS evolves when an external change is performed. Therefore, the utilization of the situation calculus and RGolog to our SDIS architecture can enable it to have a distinguishing feature when compared with other traditional systems developed to solve problems or give advice in some knowledge domain, which is its ability to adapt to and interact with changing or evolving surrounding circumstances.

3. Logical architecture for SDIS

Through exchange of information and data between user side and server side, the logical architecture for implementing SDIS allows a user to require the server side to provide adaptive information according to his/her initial situation and evolution of his/her related surrounding circumstances. The provision of such information services is supported by the use of a suite of corresponding logic programs in the form of a three-layer structure (as shown in figure 1).

Service dependent rules are composed of rules based upon an application domain that aims at one specific service. In this study, only using a set of logical programs, which describes system manager (SM) for being in charge of the affairs of how the system should behave and what kind of information the system should provide without concern for the evolution of surrounding circumstances, is not sufficient to cope with the problems describing how the system reacts to the occurrence of external changes. Therefore another set of programs is necessary for us to address the problems resulting from the evolution of surrounding circumstances. The two sets of logical programs need to be operated together. In contraposition with the former set of programs, the latter one describes the insertions, in the form of if-then rules, which are triggered by the occurrence of surrounding changes. Such rules are necessary to represent effects of changes in surrounding world to user’s behavior. They consist of a condition part and an action part: if condition then action. These rules mean that whenever the conditions become true, the correlative actions will be performed [15]. For example, if it is raining, and the traveler does not take an umbrella, then outdoor sightseeing will be cancelled, which can be expressed formally as follows:  

\[ \text{raining} \land \neg \text{umbrella} \Rightarrow \text{cancelOutdoorSightseeing}. \]

Where, \( \land \), \( \neg \), and \( \Rightarrow \) are logical connectives indicating conjunction, negation and implication respectively.

Here the condition part represents the change in surrounding world, raining, and a fact that the traveler does not take an umbrella. The action part indicates the effect of the change to the traveler’s behavior, which is to cancel the outdoor sightseeing. On the other hand, service independent rules are in general use whatever application domain a user is in. In this implementation they mainly describe involved logical operations, complex actions, and a series of syntactic transformations in Prolog clauses.

Figure 2 shows a 3-phase process through which the SDIS can be achieved. The process is initiated by the user at the time he/she is trying to ask for some information via user interface. The determination of the service facts comes from either inputted facts by the user or other inherent facts in the information database of server side. Based on the set of service facts, the aim of logical analysis phase is to reason and analyze the adaptive results at the level of logical theory and principles.

---

**Figure 1. Three-layer structure**

As the first and most basic layer, in order to address the problem of possible changes of users’ situations, evolution of the surrounding world, and the interactions between them, service facts involve not only information about users’ initial situations, including their objective situational information and subjective preferences, but also some related facts of their surrounding circumstances. A part of service facts are inherent in database of server side originally, whereas some other facts come from inputted data. User’s initial situations and related facts of surrounding circumstances are always dynamically changed.
In this research, a Java program component is mainly to implement user interfaces, provide access to the logical analysis process, interpret and transfer the results back to user interfaces in result interpretation phase. The function of logical analysis and inference will perform two sets of programs: in the absence of the occurrences of surrounding changes, the system will offer a correlative result based on the user’s initial situation; on the other hand, the occurrences of any change or evolution of surrounding circumstances will have a directly effect on the result. That is, the SDIS may give different responses to the same user simply due to the interactions between the user and his/her surrounding circumstances.

4. Case study and discussion

To understand how a specific SDIS is implemented in our architecture, let us consider a simple case of mobile shopping information service system. Based on a shopper’s situation and the evolution of her surrounding circumstance, the system is supposed to recommend appropriate arrangement and make adjustments to the results. Let us assume that the shopper is in an unknown supermarket. She would like to find a more efficient route to get the foods in her own shopping list and timely messages about the time service, which means service performed on an unscheduled basis. Commonly, some particular articles are available at a special discount price during the time service. In the following, we will look over the process from requesting information service to the suitable information provision.

4.1. Input interface

User interface of an SDIS for this case is as shown in figure 3. This figure displays that the shopper selects three articles from the shopping list, which are fennel, beef, and celery. Moreover, she especially attaches her mind to two of them, fennel and beef. She would like to learn some detailed information about them.

4.2. Development process of situations
The development process of the shopper’s situations for the above case can be illustrated as the following chain diagram, as shown in figure 4. It reveals that time evolution in the surrounding circumstances enables the system to automatically remind the shopper when the time service is about to start or finish. Interleaved with this procedure are the various arrangements responding to the shopper’s diverse preferences and requests.

![Diagram](chart.png)

**Figure 4. Development process of the situations**

To put it concretely, the situations change with both the shopper’s diverse preferences or requests, and the evolution of the surrounding circumstances. For example, when she selects fennel from the shopping list, her situation will change from the initial situation $s_0$ to $s_1$, and then if she requests more detailed information about this article, the situation will change again from $s_1$ to $s_2$, and so on. In the meanwhile, when time evolution comes close to beginning a time service, the system will send a reminder to the shopper, which makes situation change once again, and vice versa. Such a dynamically changing process in response to both changes of users’ situations and evolution of their surrounding circumstances is difficult to be dealt with using traditional problem-solving systems because of their limitations in the lack of flexibility to keep up with changing environments and to interact directly with the surrounding circumstances.

4.3. Output interface

Based on her initial situation, her preference, and the evolution of her surrounding circumstances, the correlative results for this shopper can be obtained. The key service dependent rules in performing the case analysis can be classified as the following three types (see table 1).

<table>
<thead>
<tr>
<th>Precondition</th>
<th>Axioms</th>
</tr>
</thead>
<tbody>
<tr>
<td>near(N,S), topNo(T), N &lt; T.</td>
<td>poss(goAhead,S) :- near(N,S), topNo(T), N &lt; T.</td>
</tr>
<tr>
<td>firstNo(F), F &lt; N.</td>
<td>poss(goBack,S) :- near(N,S), firstNo(F), F &lt; N.</td>
</tr>
<tr>
<td>poss(presentDetails(N),S) :- details(N,S).</td>
<td>poss(needDetails(N),S) :- not details(N,S).</td>
</tr>
<tr>
<td>poss(endDetails(N),S) :- details(N,S).</td>
<td>poss(putinList(N),S) :- not inList(N,S).</td>
</tr>
<tr>
<td>poss(remindTimeservice,S).</td>
<td>poss(remindEndTimeservice,S).</td>
</tr>
</tbody>
</table>

With the help of Java language, the results at the level of logical form can be transformed into a more easily understandable form, which is displayed on the user interface as shown in figure 5.

![Interface](interface.png)

**Figure 5. Output interface**
The shopping route planned out by the SDIS is in accord with location relationship between the selected articles and the initial position of the shopper. In addition, the detailed information of those articles that the shopper shows interest in are presented. Also the result window provides timely reminders to help the shopper catch some particular articles during the time service.

5. Conclusion

We present a logical architecture for implementing the SDIS with surrounding awareness capabilities. The proposed architecture is achieved through a three-phase process, which can logically reason and offer information that addresses both the changes in user’s situation and the evolution of his/her surrounding circumstances. It is the first use of the formal language, the situation calculus and the corresponding logic programming language RGolog for the domain of mobile applications and services. This utilization enables us to be able to represent and analyze a dynamically evolving system in a formal way, and also enables the proposed SDIS architecture to adapt to and interact with changing or evolving environments. Our logical architecture and its application to one case demonstrate feasibility of the proposed approach. However, it is still not adequate only depending on the current architecture to accommodate more complex scenarios. One of our future works will concentrate on outlining a systematic methodology, which can produce detailed formal specifications of a situation dependent information service process so as to deal with more complex cases and deliver more adaptive content and information to users. Another future work is to integrate more technologies, such as Felica or automated sensors, with our logical architecture so as to develop more intelligent system.

6. References


