Applying Situation Awareness to Mobile Proactive Information Delivery

SuTe Lei, Kang Zhang, and Edwin Sha

Department of Computer Science, The University of Texas at Dallas,
Richardson, TX 75083-0688, USA
sute@acm.org, {kzhang, edsha}@utdallas.edu

Abstract. Proactive information delivery systems disseminate information to users based on their current tasks. Interests for proactive information delivery to mobile users are growing. In a mobile environment, users have access to additional ambient information that is not usually present in a fixed working location. It provides an opportunity to improve the quality of information delivery service by utilizing ambient information. However, mobile users tend to be distracted by their surroundings. Ideally, the information delivered to mobile users should match with their current tasks and needs. The challenge is how to determine users' needs. We tackle this issue by analyzing users' responses to delivered information and learning from their perceptions towards situations. This paper presents an interval-based approach for situation specification. We show a prototypical system that uses the reinforcement learning technique to obtain better understanding of users' perceptions towards situations.

Keywords: visual specification, situation awareness, proactive system, reinforcement learning, interval algebra

1 Introduction

Proactive information delivery is a software service that delivers information to users based on their current tasks. Typical examples are systems like Amazon.com’s associated product recommendation and Google’s AdWords advertising. Today’s advanced mobile technology enables users to gain access to information ubiquitously. Unlike the user using a stationary desktop computer, mobile users have more choices of information access such as mobile phones, UMPCs and PDAs. The mobility nature provides proactive computing an opportunity to improve the quality of information delivery by identifying users' locations and their current tasks via pervasive or ubiquitous computing services. In a pervasive computing environment, additional ambient information is made available for more accurate detections of the user’s current context. It thus facilitates a better quality of proactive information delivery service. However, there are some challenges in delivering information to mobile devices. First, mobile devices have limited resources, e.g., CPU power, display size and memory space. Input techniques are also different. Second, the ambient environment is more dynamic. Users might be on the subway train or in a shopping mall while using the system. The possibility of being distracted by the surroundings is high. As pointed out by Chittaro [4], in mobile situations, using the device often becomes a secondary rather than a primary task. Proactive information delivery might result in being intrusive to mobile users. Thus, we must take extra measures when migrating proactive services from desktop operating environments to mobile ones. In addition, the precise meaning of a situation is vital for determining the contents to be delivered to mobile users. A specification method is needed to achieve this.

In order to provide proactive services, we apply the concept of situation awareness to improve the appropriateness of information delivery. The notion of situation awareness as defined by Endsley [5] is the perception of the elements in the environment within a volume of space and time, the comprehension of their meaning and the projection of their status in the near future. We present our idea based on Endsley's definition that encompasses the notion not only on spatial aspect of the perceived information but also the temporal constraint which might have an impact on users' tasks. We begin the discussion by reviewing related work in section 2. Section 3 describes our specification method. In section 4, we discuss the issues in applying situation awareness and propose a prototypical system. We conclude our discussion in section 5. The contributions of our work are twofold:

- we propose a specification approach using interval algebra to model situations.
- we implemented a prototypical proactive information delivery system that employs a reinforcement learning technique to understand users' perceptions towards the delivered information.

2 Related Work

The key feature of proactive information delivery systems is the ability to provide information based on the detected user needs. In the literature, the capability is often referred as context awareness. A plethora of context-aware systems have been proposed in recent years. Due to space limitation, we limit our discussion to those systems that are closely related to proactive systems and systems employing the concept of situation awareness.

Yau and Liu [10] proposed an approach to incorporate situation awareness in service specification for situation-aware service-based systems using SAW-OWL-S, an extension of OWL-S with situation ontology. They presented a method to identify the relationship between situations and services in situation-aware service based systems. We share a similar view with their work that situation specification plays an important role in system developments. The Watson system [3] is an information management assistant system that observes user interactions with everyday applications, anticipate information needs, and automatically fulfill them using Internet information sources. A query in Watson is grounded in the context of the user's tasks. The system turns everyday applications into intelligent, context-bearing interfaces to conventional information retrieval systems. The FXPAL Bar system [2] is a proactive information recommendation system designed to provide contextual access to corporate and personal resources. It enhances information recommendations by adding three features - *translucent recommendation windows* in the program interface which increases the user's awareness of particularly highly-ranked recommendations, *query term highlighting* that communicates the relationship between a recommended document and the user's current context, and a r*ecommendation digest* function that allows users to return to the most relevant previously recommended resources. The TaskNavigator system [7] improves the reuse of previous process know-how by proactively suggesting similar tasks or relevant process models based on textual similarities. It uses the functionality of the commercial system, BrainFiler, to find similar document categories. The system allows the user to build a personal knowledge space. The concept is similar to the use of user profile.

One of the earlier systems using the user profile approach is the Letizia system [8]. It uses implicit feedback such as bookmarking, to learn a user profile. It performs lookahead search in the locus of the page the user is currently viewing and recommends links accessible on the current page. Many other related work using recommendation agents to improve information retrievals can be found in Greengrass's survey [6]. Our work differs from previous approaches in that we focus on the aspect of data representation and information delivery services to mobile users.

3 Situation Representation

A situation is a finite sequence of actions as defined in situation calculus [9]. In our view, a situation is a result of an orderly combination of actions that occur over a finite interval of time. We use a four-layer model to represent situations – data atom, data element, data transition and situation. The data atom layer contains low-level data obtained directly from data sources. On top of the data atom layer is the data element layer. Data elements are fusions of data atoms. The next level is the data transition layer which specifies the actions performed on data elements. The top level is the situation level which defines situations using a combination of actions. The data components at the four layers are described in table 1.

Each data atom has five main attributes - name, value, value type, timestamp, and data source. The sources of data atoms could be one of the followings:

- raw data from sensor systems such as temperature, humidity, and noise level
- data feeds from web services such as up-to-the-minute financial and weather data
- message notification such as mobile phone's SMS messages and emails

A data element is an interpreted piece of information inferred from data atoms. Each data element is defined by its name, an element type, a set of data atoms, an interpretation function, and a list of interpretations. For example, we may define a data element indicating three degrees of comfort level (cold, pleasant, humid) of a meeting room by setting the readings from its room temperature and humidity. We may also define an email filtering function that serves as a rule to process the content of a data atom pulled from an email box, and the data element being defined is assigned with an email category which allows an email program to perform further processing.

As mentioned above, a situation is an orderly combination of actions. An action is the fact or process of doing something. From the system perspective, we want to be able to reflect this fact in a way that the system can utilize it for further processing. In real life, an action spans over an interval of time despite the granularity of the interval. For example, the declining process in a particular stock from \$35.8 to \$32.2 from 10:00 AM to 11:00 AM. The process may be caused by a variety of reasons. In a situation-aware environment, the causes for any change in data value are quite often unknown. To capture the act of a process, we thus pay attention to the consequential effect of an action, that is, the state change of a data element. We use *data transition* to represent the effect of an action and define it as a process spans over an interval marked by a begin state and an end state (see Table 1). Since actions are treated as an interval, we need to consider the temporal relations among them. According to Allen's interval algebra [1], seven basic possible relations between two distinct intervals exist – *before*, *meets*, *overlaps*, *starts*, *during, finishes* and *equals*. We developed a set of notations for specifying situations based on these terms. Basically we specify the begin and end of a data transition by adding two suffixes at the end – *begin* and *end*. For example, *StockRise.begin* indicates the begin of *StockRise* data transition. Fig. 1 depicts the relations and our notations. We left out the *equals* relation as it is not used in our context.

Table 1. Definitions of data components.

Data component	Description
Data atom	A low-level data unit obtained from a data source by data pushing or pulling.
Data element	A data element is $e = (A, f)$, where A is a set of data atoms and f is an interpretation function that maps the combined values of A to a list of interpretations.
Data transition	A data transition is $t = (e_i, v_i, v_e)$, where v_i is the initial value of a data element or elements e_i with the element type i, and v_e is the ending value of e_i.
Situation	A situation is $S = (T,R)$, where T is a set of data transitions, and R is the relations among I. The possible relations are *before*, *meets*, *overlaps*, *starts*, *during*, and *finishes*.

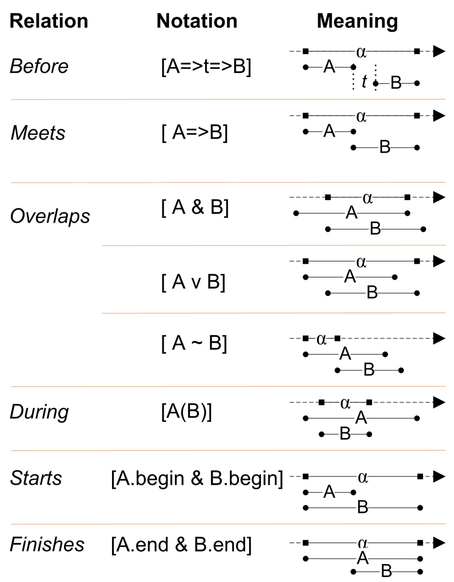


Fig. 1. Temporal relations of data transitions. The set of operations of two distinct data transitions, their notations, and temporal meanings. The value of α is evaluated to true if the relation condition is met.

By using an interval-based approach as opposed to a point-based one, we are able to more intuitively represent situations and perform queries on them. For example, suppose we have the following two actions:

```
A1: Tom enters through the building entrance.
A2: Tom is in a meeting.
```

A1 normally occurs and ends in just few seconds, whereas the duration of A2 is usually longer than 2 minutes. We may determine if Tom is still in the meeting by checking the existence of A2. We may also check if Tom is in the building by checking if A1 has occurred. The difference here is in the time of checking. We check the status of A2 while A2 is still happening, and we are only interested if A1 has occurred.

4 A Situation Aware Proactive Information Delivery System

The concept of situation awareness has long been used in mission-critical information systems such as military and disaster control systems. There are basically three levels of awareness - perception of elements in current situation, comprehension of current situation, and projection of future status. Working memory and attention are the two key factors that influence the accuracy and completeness of situation awareness. The way in which attention is employed in a complex environment with multiple competing cues is essential in determining which aspects of the situation will be processed to form situation awareness [5]. We must take these factors into consideration when applying situation awareness to proactive information delivery services. In the subsections below, we will firstly discuss the method for discovering situations and assisting situation comprehension and projection. We then propose a prototypical system based on the methods discussed.

4.1 Situation Awareness

The goal of applying the concept of situation awareness to proactive information delivery systems is to improve the appropriateness of the delivered contents. One of the key aspects in situation awareness is that it is a cognitive process of continuous adjustments in the user's perception and comprehension towards a situation. The task is not merely identifying the current situation of a user, but also continuously monitors the implication of the detected situation to the user's perception. From the system perspective, we need a way to discover how a user perceives a detected situation. It thus requires a mechanism that enables the system to learn from users' actions and adjust the delivery policy accordingly.

Among several types of machine learning techniques, reinforcement learning best suits our goal. It differs from supervised learning in that correct input and output pairs are never presented, nor sub-optimal actions explicitly corrected. It tries to find a balance between exploitation of current knowledge and exploration of uncharted areas. To apply reinforcement learning into our system, we have the following setup:

- specify a set of situations for triggering information deliveries
- define a discrete set of environment states
- specify a discrete set of actions
- define a set of scalar rewards for the set of environment states.

Upon a situation detection, the system sends out information to its users. When users receive the information, they have few options to respond. The system registers each response and calculates the reward. Based on the calculated reward value, the system adjusts the delivery policy for the next delivery. The goal is to find an optimal delivery policy, mapping states to actions, that maximizes a long-term reward value. Fig. 2 illustrates the process.

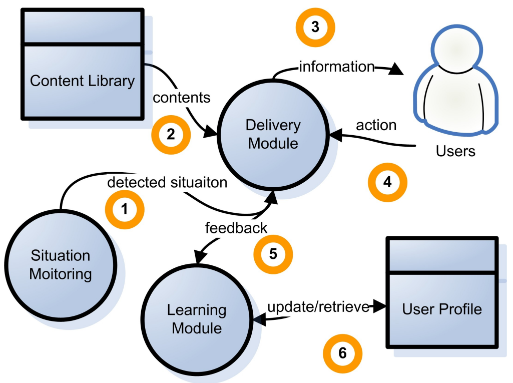


Fig. 2. The reinforcement learning process. A detected situation is processed by the delivery module which chooses an appropriate delivery policy and data contents from the content library. This information is then disseminated to users. Responses from users are fed back to the learning module. The learning module registers users' actions and calculates the reward values and stores the data into the user profile. Updated information contents are sent to users. Depending on the information content, the process can be repeated until the interaction session ends.

As the interactions between users and the system are open, the learning process can take an indefinite period of time to converge. For example, suppose that we have a set of recommended contents and a series of steps for users to follow. Depending on users' actions, predefined recommendations will be sent at each step. When a user receives a pushed content, he/she may or may not reply. In addition, we do not know if the user will follow through the preprogrammed steps, and we do not know when the user will respond to the delivered contents. The user might stop using the system at any point due to a variety of reasons. As a result, the learning task becomes an ongoing process. In some cases, such as interactive advertisements, immediate or short-term learning results are required. One way of forcing the system to obtain a final reward value is to impose a time constraint or to limit the number of interactions. It can be specified in each delivery policy. In next section, we will describe a prototypical system based on the concept discussed.

4.2 A Situation Aware Proactive Information Delivery System

We implemented a prototypical system that facilitates proactive information delivery services using the reinforcement learning technique described. The system has four main components – a visual tool, situation monitor, dispatching and learning module. Fig. 3a shows the overall structure of the system.

We developed a visual tool for specifying situations, information contents and delivery policies. Situation specifications are stored in the situation library in XML formats for the situation monitor to process. Fig. 4 shows a screenshot of the visual tool.

The situation monitor consists of a pool of data watchers and a controller. The controller parses the situation specifications and evenly distributes data monitoring tasks to each data watcher. The data watcher periodically checks the values of data elements and sends back the latest status to the controller. The controller keeps track of the status of a situation. Based on the specification, it triggers the information delivery by alerting the dispatching module.

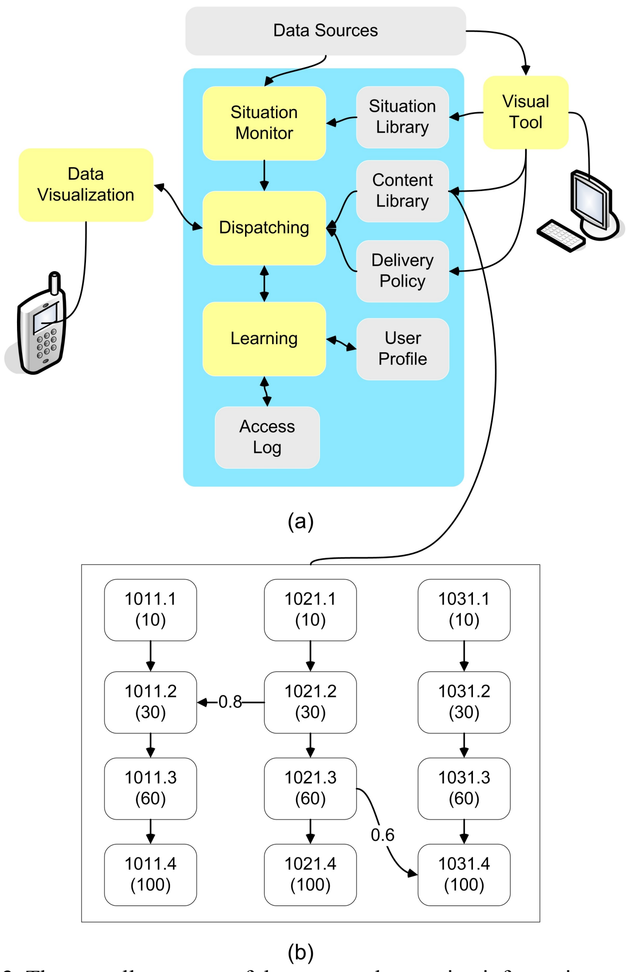


Fig. 3. The overall structure of the proposed proactive information system.

The messages stored in the content library are marked with *message title*, *message type*, *code number*, *level*, *reward value*, *message content*, and *visualization type*. The level of a message indicates a user's level of interest to the message. The content library structures messages in a message hierarchy. Each type of message has several levels of details. At each level, a message can be linked to other types of messages. Whenever the system delivers a message, it attaches two basic options – *cancel delivery* and *get more info*. The *cancel delivery* option notifies the system not to send the same type of message in future deliveries. The *get more info* option requests the system to send more detailed information. If there are links to other types of messages, these links will be delivered as menu options for the user to choose. That is, in addition to the two basic options, a delivered message will be attached with options to get more information on the linked contents.

Each response from users gets a reward value. The reward value is an indicator of interest. It is calculated by the message level. Links to other types of messages get a percentage of the reward value. Fig. 3b shows an example of message hierarchies. Suppose that message 1021.2 is sent. The attached options will be '1) *cancel delivery'*, '2) *get more info'*, and '3) *more on 1011.2'.* If the user selects 3), the reward value for this type of message will be 30 + 0.8*30, that is, 54. If the user selects 2), the reward value will be the current reward value of 30 plus the reward value at the next level, that is, 90. If the option '1) *cancel delivery'* is selected, the reward value is set to -1, which means that the user is not interested in the delivered message.

Users' responses are traced and stored in the access log. Each record has six attributes – *user id, timestamp, message id, response, originating message id* and *location coordinates*. The timestamp and location attributes play an important role in analyzing a user's interaction behaviors with the system. All delivered messages to users have reward values and are stored in the user profile. The reward value is calculated by the following formula:

$$\sum_{t=1}^{n} (I * Rt)$$

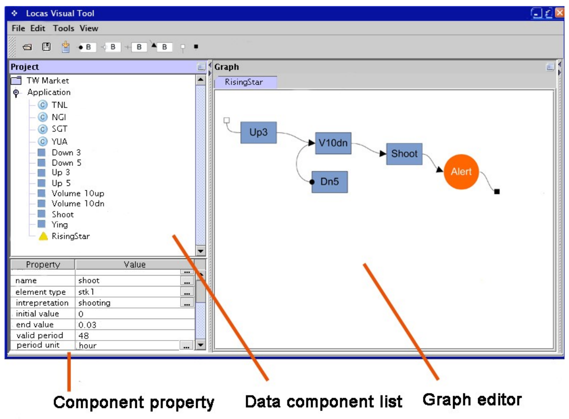
where n is the number of level in content details, I is the percentage value of a link, Rt is the reward value for the level t.

Each message delivery is dictated by a delivery policy. The default delivery policy is delivering messages based on the reward value. The reward value shows which level of message is to be sent. For example, a reward value between 10 and 40 tells the system to deliver the message contents at level 2; a reward value greater than 40 tells the system to deliver the message contents at level 3. We may overwrite the default delivery policy by changing the reward value and level mapping. Location data may also be used to restrict the information delivery. For example, we can specify the system to deliver level 3 contents if the reward value is grater than 65, and we can also insert other message links in the options. The delivery policy is specified in a XML format. The following text shows an example.

```
<policy id="1011">
<userid>1238,1233,1452</userid>
<mapping>
  <level>1</level>
  <value>10-35</value>
</mapping>
<mapping>
  <level>2</level>
  <value>35-60</value>
  <options>1021.2,1031.2</options>
</mapping>
<location>(N49,W122),(N49,W101)</location>
</policy>
```

The dispatching mechanism is by means of short message services (SMS) and information downloading using a software program written in Sun's J2ME framework. We implemented a visualization program to be used on mobile devices in Adobe

Flash. When an SMS message is received, the user selects the link in the message and the visualization program is invoked to download the information from the server. After downloading, it displays an alert to notify the user the arrival of new information. Each delivered message indicates a visualization method. We currently implemented two main visualization methods – textual and graphical. The textual method just displays the message contents in a simple text form. The graphical method shows the message contents in tables and charts. We will illustrate how it works with an example in the next section.



Property	Value
name	shoot
element type	stk1
intrepretation	shooting
initial value	0
end value	0.03
valid period	48
period unit	hour

Fig. 4. A screenshot of the visual tool.

4.3 An Example

We implemented a stock market alerting system that has the following features:

- a list of alert types with several levels of content details
- users can subscribe to a list of stock alerts
- users have access to the history of sent alerts and responses

On the server side, we developed a data aggregator that collects stock data at an interval of 15 minutes and provides a web service for the situation monitor. We set up a set of situations based on the Japanese candlestick patterns such as shooting star, hanging man, etc. For each stock, we established an entry in the content library which has four levels of details. The data at the highest level contain full information about the stock including its company background, recent and historical financial figures, major shareholders and other related information. Each stock belongs to an industry sector. Stocks of the related sectors have links among them. Each link has a percentage in the degrees of interest. The percentage values are established by calculating the correlation values based on the historical stock data. We analyzed the data for the past two years, and identified the correlated stocks using an association rule technique. We do not discuss the technique here as it is not the focus of this paper.

The communication between users and the system is through SMS messages and data downloads. The user can obtain a list of stock alerts by sending a SMS message to a mobile phone number owned by the system. The server system has a GSM modem attached which picks up all incoming SMS messages. When an alert is triggered by the situation monitor, it checks the subscriber list, and sends out SMS messages to all subscribers based on the corresponding delivery policies.

We set the delivery policy to ensure the user gets same level of contents at least three times before moving up to the next level. That is, they have to select the *get more info* option at each level three times. In terms of the reward value, it has to be added three times to reach the next level. If the same type of an alert is sent 10 times without any reply, the system adjusts the reward value to -1 for that user and gives up future alerts to the same user.

The learning goal for the system is to identify the group of users who are seriously concerned about a set of stocks. These stocks are managed by an investment group who provides investment advice services. Future investment campaigns will first use the list of identified users as the main targeting customers.

During the development of the system, we studied several information visualization methods. As our targeting users are mobile customers, we need a way to fully utilize the limited display space and keep them focused on the presented information. In addition, we wanted to reduce the intrusiveness of an alert as much as possible. The most desirable way is to know what the user is currently doing. There are several methods to achieve this. One way is to obtain the user's daily schedule and current location, and avoid sending alerts while the user is busy, such as in the meeting or possibly driving. The problem for us is that we did not have such data. We resorted to the user access log. We developed a software program that walks through the log and sieves out the time each user most frequently replied to the system. The program establishes a set of time intervals that have higher possibility of getting replies. We added a scheduled delivery feature to the dispatching module which uses the statistics obtained.

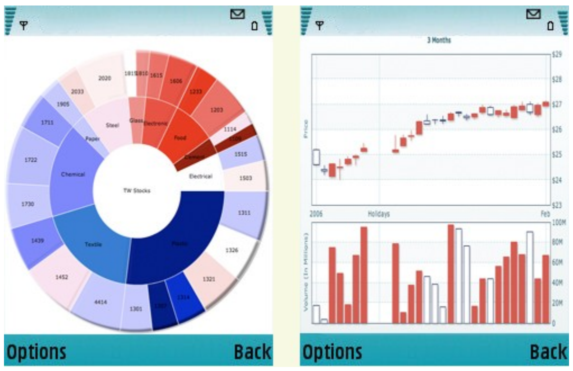


Fig. 5. Screenshots of mobile data visualization.

Fig. 5 shows a screenshot of the information visualization program on a Nokia mobile phone. A multilevel pie chart is used to give an overview of the interested stocks. The inner circle is formed by the industry sectors, and the outer circle consists of the interested stocks. For each piece in the pie chart, the hue of a color represents the level of increase or decrease. Red color represents an increase, whereas blue color indicates a decrease. The darker the color, the more intense it is. The size of a pie piece shows the trading volume. A detailed candlestick chart is shown by entering the stock number. The user can request more detailed information in the *options* menu. When the request is sent, the server will register the request and update the user profile as discussed above. Our initial experiments with a group of 50 users were quite positive. We were able to identify a small group of dedicated stock watchers who constantly checked a set of stocks they did not own. More experiments are still needed for more diverse groups of users.

6 Conclusion and Future Work

In this paper, we presented an interval-based approach for specifying situations. We employed a layered model to represent situation-related data. Situations are specified and represented in terms of interval algebra. In order to better understand users' perceptions towards various types of situations, we developed a prototypical proactive system for delivering information to mobile users. We used the reinforcement learning technique to adjust the delivery policy for each user. An example on stock alerts was presented. We showed how our system visualizes mobile data and learns from the user's responses. As it can be seen in our presentation, our approach requires more field studies to improve the learning results. Future work includes investigations into content library structures and content associations. Mobile interactions with the system also require more study.

References

1. Allen J. and Ferguson G. Actions and Events in Interval Temporal Logic. Journal of Logic and Computation 4 (5), 1994, 531-579.
2. Billsus, D., Hilbert, D. M., and Maynes-Aminzade, D. Improving proactive information systems. Proceedings of the 10th international Conference on intelligent User interfaces. IUI '05, 2005, 159-166.
3. Budzik, J., Hammond K., and Birnbaum, L. Information Access in Context. Knowledge-Based Systems 14 (1-2), Elsevier Science, 2001.
4. Chittaro, L. Visualizing Information on Mobile Devices. Computer 39, 3, Mar. 2006, 40-45.
5. Endsley, M. R. Theoretical underpinnings of situation awareness: A critical review. In M. R. Endsley & D. J. Garland (Eds.), Situation Awareness Analysis And Measurement. Mahwah, NJ: LEA, 2000.
6. Greengrass, E. Information Retrieval: A Survey, November, 2000. http://www.csee.umbc.edu/cadip/readings/IR.report.120600.book.pdf.
7. Holz, H., Rostanin, O., Dengel, A., Suzuki, T., Maeda, K., and Kanasaki, K. Task-based process know-how reuse and proactive information delivery in TaskNavigator. Proceedings of the 15th ACM international Conference on information and Knowledge Management CIKM '06, 2006, 522-531.
8. Lieberman, H. Letizia: An Agent That Assists Web Browsing. Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, Montreal, Quebec, Canada, 1995, 924-929
9. Reiter, R. The Situation Calculus Ontology. Electronic News Journal on Reasoning about Actions and Change, Vol. 2, 1998.
10. Yau, S. S. and Liu, J. Incorporating Situation Awareness in Service Specifications. Proceedings of the Ninth IEEE international Symposium on Object and Component-Oriented Real-Time Distributed Computing (Isorc'06) - Volume 00. Washington, DC, 2006, 287-294.