Anytime Cognition: An information agent for emergency response

Felipe Meneguzzi, Jean Oh, Nilanjan Chakraborty and Katia Sycara
Carnegie Mellon University
Pittsburgh, PA
{meneguzz,jeanoh,nilanjan,katia}@cs.cmu.edu

Siddharth Mehrotra
Agent Dynamics Inc.
Pittsburgh, PA
siddharthmehrotra11@gmail.com

Michael Lewis
University of Pittsburgh
Pittsburgh, PA
ml@sis.pitt.edu

***Abstract*—Planning under pressure in time-constrained environments while relying on uncertain information is a challenging task. This is particularly true for planning the response during an ongoing disaster in a urban area, be that a natural one, or a deliberate attack on the civilian population. As the various activities pertaining to the emergency response need to be coordinated in response to multiple reports from the disaster site, a user finds itself cognitively overloaded. To address this issue, we designed the Anytime Cognition (ANTICO) concept to assist human users working in time-constrained environments by maintaining a manageable level of cognitive workload over time. Based on the ANTICO concept, we develop an agent framework for proactively managing a user's changing information requirements by integrating information management techniques with probabilistic plan recognition. In this paper, we describe a prototype emergency response application in the context of a subset of the attacks devised by the American Department of Homeland Security.**

I. Introduction

Planning for complex activities often involves consulting multiple information sources in order to reduce uncertainty associated with their decision making. As humans interleave planning, execution and re-planning, managing information to meet the changing requirements becomes a cognitively demanding task. Consequently, users who must make time-critical decisions are cognitively overloaded due not only to the planning activities but also to the information requirements of the planning and re-planning. In this context, we develop the *Anytime Cognition* (ANTICO) concept to assist cognitively overloaded users through an information assistant agent.

Our approach to prevent user cognitive overload is to consider user tasks before one actually needs to carry them out. By recognizing a user's plan for future activities, the assistant agent can act proactively to help the user balance her workload over time. ANTICO envisions a networked system of humans and software agents where the agents enhance human user performance by adaptively aiding in efficient and accurate decision-making. The agents help the humans by reactively (in response to direct user requests) and proactively accessing information, and presenting information in a suitable form that takes into consideration the time available for decision making and the user cognitive workload.

In order to transition the ANTICO concept to a real-world scenario, we develop a proof of concept application of ANTICO created to assist a disaster response manager. This manager must deal with a chemical attack against a large civilian facility in a major US city, while dealing with uncertainty throughout the response. Uncertainty stems from the nature of the chemical used in the attack, and later from the various second-order effects. Here, an ANTICO agent is tasked to assist an emergency response manager who must make decisions under time-pressure, analyzing a stream of information arriving from various localized sources while keeping track of the big picture in order to effectively coordinate multiple agencies that must be directed to perform activities around the affected areas. Most crucially, since response managers must make decisions within tight time schedules, information they need must be presented to them in ways that can be acted upon within decision deadlines. For example, in the case of a chemical attack involving the sarin gas, the first responders should reach the site within $10 - 15$ minutes of attack (as per Department of Homeland Security (DHS) guidelines). Thus assuming some travel time required for the first responders to reach the event site, the operator will have about 5 minutes to diagnose the attack and contact the relevant authorities, *e.g.*, Hazardous Materials (HazMat) unit and emergency medical services. Delays or wrong diagnosis can be fatal in some cases, like sarin gas attack, not only for the population under attack but also for first responders.

Our contributions are the following. We introduce the Anytime Cognition (ANTICO) concept for a hybrid team of humans and software agents. ANTICO extends prior work on proactive agent architecture [1], integrating a new module for cognitive workload estimation. To support our domain-independent agent architecture, we define a language, ANTICO Domain Description Language (ADDL), for specifying problem domains. We develop an emergency response scenario based on the Standard Disaster Scenarios Planning documents [2], and present an application using this scenario.

The rest of this paper is organized as follows. After describing the emergency response scenario in Section II, we introduce ADDL, an object-oriented domain description language to formally describe the target problem in Section III. We describe our agent architecture that supports the ANTICO concept in Section IV, and the software design and implementation using a step-through example in Section V.

We discuss related work in Section VI and discuss practical issues in Section VII.

II. Scenario Description

To demonstrate the applicability of the ANTICO approach to the real world, we develop a scenario based on the National Planning Scenarios created by the Department of Homeland Security (DHS) [2].[1] According to [2], planning the response for these scenarios encompasses 10 mission areas, which are more or less equivalent to phases in the response. ANTICO focuses on six of these areas, namely:

1) Emergency Assessment/Diagnosis;
2) Emergency Management/Response;
3) Incident/Hazard Mitigation;
4) Public Protection;
5) Evacuation/Shelter; and
6) Victim Care.

One major source of uncertainty in the early stages of many emergency response regards the identification of the nature of the disaster, i.e. diagnosing the emergency. This is particularly crucial when the disaster in question involves an invisible source of casualties, such as in the case of industrial accidents or attacks involving chemical, radiological and biological agents. In such cases, the wrong response might be more damaging than no response at all, since, for example, one may turn first responders to casualties if these are not prepared with appropriate protection. As a consequence, we have designed a response that considers the possibility of two different chemical agents from among the 15 possible disaster scenarios included in the DHS report, namely Sarin and the (fictitious) Yellow (Lewisite) chemicals. These two chemical agents were chosen because they have a small number of common symptoms and are colorless, creating a degree of initial uncertainty during the initial stages of the emergency response. Managing a response for a chemical attack involves complex workflow and information requirements for the operator. Although we have taken these two chemical agents as representative of the initial uncertainty, in a full deployment of ANTICO, dozens of similar chemical agents would needs to be considered in a complete response workflow.

The attack scenario utilized in the demo consists of the simultaneous release of multiple canisters of Sarin gas within a crowded public building in Washington DC, namely Union Station, by elements of a militant cell. Sarin is a chemical warfare agent classified as a nerve agent, and as such, is among the most toxic and rapidly acting chemical warfare agents currently known. These toxic agents act in a way similar to pesticides, but have a much stronger effect on human physiology. Sarin is expected to kill 95% of the people exposed to it in a confined environment, as well as affect a significant portion of the first responders, if not protected adequately. Since Sarin is a liquid that can be easily evaporated, once released, it will affect not only the people within the building, but it will also be released into the environment through the rooftop ventilation system within the building, creating a hazardous poison cloud that will affect people downwind from the original attack site.

In our simulation, the operator plays the role of an emergency management coordinator who receives messages from the field, tries to develop situation awareness for the developing incidents and dispatches/provides information to other human teams, such as police and medical services, to resolve the incidents. Notice that the operator has no prior knowledge of the actual chemical agent used in the attack. Since Sarin and Lewisite have somewhat similar symptoms, extra care must be taken that the correct chemical is identified and the appropriate responses are made, especially since the two chemicals have different time horizons and damaging effects. In order to provide assistance for operators in charge of coordinating response for these scenarios, we have developed detailed response workflows encoding the relevant actions, constraints and needed information for the various possibilities of these chemical attacks. To accomplish these tasks the operator must combine *pushed* information (*e.g.*, messages from units in the field or the ANTICO agent) with actively *pulled* information from various sources, such as contact telephone numbers of needed units (*e.g.*, HazMat) or reports of preparedness levels.

III. ANTICO Domain Description Language

ANTICO is designed as a generic agent architecture that can be applied to various problem domains. When developing a specific application, a software designer needs to engineer domain specific information such as user workflows and information sources. To facilitate this task, we designed the ANTICO Domain Description Language (ADDL) as an XML-based language for the problem domain. ADDL is not only a language to design a new application but also a well-structured medium used within the ANTICO components.

An ADDL object consists of a set of user state *variables* and a set of user *activities*. An example fragment of ADDL is shown in Table I. A user state **variable** element specifies a significant factor for determining the user's current state and work progression. A state variable has a unique name and a domain of valid values, *e.g.*, variable `zip-code` shown in line 4–6 has a numeric value in range $[15201, 15295]$. A complete value assignment of variables represents an instance of user state, from which changing the values of one or more variables results in transitioning to a new user state. A set of state variables and associated value domains, therefore, define the user's state space.

For each user activity, ADDL must specify the following set of critical elements associated with a user's activity (detailed below):

- observable features while a user performs the activity;
- information needs for the user activity, and required presentation details; and
- non-deterministic effects in variables.

A user activity in a workflow is represented by the **activity** XML element. In order for the agent to properly infer the planning state of the user, it must have a model of

[1] https://www.llis.dhs.gov/docdetails/details.do?contentID=13712

```
1  <?xml version="1.0" encoding="UTF-8"?>
2  <anticoDomain>
3    <stateVariables>
4      <variable name="zip-code">
5        <domain type=numeric min=15201 max=15295/>
6      </variable>
7      <variable name="hazmat-dispatch">
8        <domain type=boolean/>
9      </variable>
10     ...
11   </stateVariables>
12   <activities>
13     <activity name="callHazMat">
14       <observations>
15         <observation name="dialedXYZ" prob=".5" />
16         <observation name="lookedContacts" prob=".5" /
               >
17       </observations>
18       <infoObject>
19         <query value="select phone from Contacts where
                name='HAZMAT' and zip=$(zip-code)$" />
20         <constraints>
21           <deadline value="17:00 02-06-2011 GMT" />
22         </constraints>
23         <retrieval status="queried" source=Contacts"
               timestamp="" data="" />
24         <presentation><zoom-coords="" /></presentation
               >
25       </infoObject>
26       <effects>
27         <variable name="hazmat-dispatch" value="true"
               prob="0.9" />
28       </effects>
29     </activity>
30     ...
31   </activities>
32 </anticoDomain>
33 </xml>
```

Table I
AN EXAMPLE FRAGMENT OF ADDL SPECIFYING THE EMERGENCY RESPONSE SCENARIO.

the observable features for each activity. An **observations** element represents a possible observation from the specified user activity, associated with field **prob**–the probability of the named observation occurring during the user activity. Although domain engineers might provide initial estimates of the probabilities of these observations, in the long term, it is expected that these values will be learned using data collected from previous episodes.

Each activity may have one or more associated pieces of information that the user will need when carrying out the activity. For example, lines 18–25 in Table I specify a requirement for contact information in the **infoObject** XML element. A **query** specifies the SQL statement for retrieving phone numbers, *e.g.*, of HazMat units in the area specified in the zip code. The contact information must be made available by the **deadline** included in the set of **constraints**. Additionally, an information object also contains its **retrieval** status, and an optional **presentation** sub-element. Using the presentation element, domain engineers can specify instructions for how the information should be presented, *e.g.*, an area of map can be *zoomed* in or information can be displayed as an *alert*. When a new type of instruction is needed, ADDL can be extended to support the new instruction. Subsequently, appropriate handling methods must also be implemented to support the new instruction in the Information Presenter module described in Section IV-F.

IV. ANYTIME COGNITION ARCHITECTURE

Concrete implementations of the ANTICO concept for proactive assistance are created following the generic ANTICO architecture. This architecture comprises multiple AI components including probabilistic plan recognition and intelligent information management. Figure 1 shows a modularized view of the ANTICO components and how those components are interconnected. The rectangles represent the main components; the third-party components are drawn in dotted lines; an ADDL specifying a problem domain is provided as an input to the system; and the information object is the communication medium representing a user's information needs.

Here, we specifically focus on the following two desiderata for the assistant agent. First, the agent must be able to recognize a user's current and future activities. Second, the agent's interaction with the user must be unobtrusive and adaptive to user cognitive workload. The User Observer module is responsible for monitoring various parameters indicating a user's current activities and her environment. When a change is observed, the Intent Predictor module analyzes the new observation to identify the user's intention and make predictions for the user's future activities according to a workflow model specified in ADDL. Subsequently, the Information Gatherer communicates with a set of information sources to meet the information requirements relevant to the predicted future user activities. Concurrently, the agent maintains an estimated user cognitive workload based on a set of observed temporal parameters in order to determine the appropriate level of detail in presenting information to the user.

The functions of each main component are described in the following subsections. Many of these components are leveraged from our previous work [1] which we refer to for a more detailed description of the AI techniques involved in each component. In this paper, we focus on software design.

A. User Observer

The User Observer module obtains and interprets user activities and messages that arrive from team members in the field. This module includes multiple *observer objects*, responsible for collecting and interpreting observations from different sources, *e.g.*user interface, input devices, external communication. Each observer object must implement the following three main methods: *monitor*, *interpret*, and *notify*. For instance, the mouse observer *monitors* a user's mouse clicks on a map, or scrolling up and down actions and *interprets* them into a set of predefined observation terms. The User Observer *notifies* the intent recognition module of new observations so that the agent's belief about the user's current and future intent can be updated accordingly.

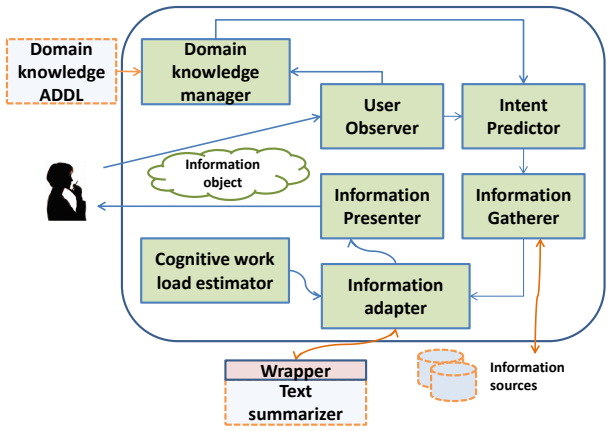

Figure 1. Architecture Overview

B. Intent Predictor

The Intent Predictor employs a user workflow model in ADDL (defined in Section III). According to the workflow model, the Intent Predictor analyzes a series of observations collected from the User Observer to recognize the user's current state and the most likely sequences of future user activities. A user's current state (which the agent cannot observe directly) is represented as a probability distribution over a set of possible states, known as a *belief state vector*; and the most likely sequences of user actions are represented in a tree structure, a *plan-tree*, where each node of the tree specifies a user activity.

We update the belief state vector as follows. For each state $s \in S$, element $b(s)$ of belief state vector represents the probability of user being in state s given a sequence of observations $z_1, ..., z_t$, denoted by $b(s) = p(s_t = s|z_1, ..., z_t)$. We can rewrite this probability as a fraction of seeing the particular state s after the series of observations over all possible cases as follows:

$$b(s) = \frac{p(z_1, ..., z_t \wedge s_t = s)}{\sum_{s' \in S} p(z_1, ..., z_t \wedge s_t = s')},$$

where $p(z_1, ..., z_t \wedge s_t = s)$ can be efficiently computed using a dynamic programming technique as shown in [3]. For instance, after observing the user opening an event message and mouse clicking on a certain area of map, the agent updates its belief state to reflect that the user is assessing the event.

The updated belief state triggers two agent functions. First, the belief state is passed to the Information Presenter to determine if the agent has relevant information to present to the user. Second, the agent recomputes the plan-tree of predicted user activities likely to follow from the updated current user state, using a sampling method. Based on the assumption that the user will act optimally according to the workflow model, a node is added for each action such that the information requirements associated with actions with higher expected utilities is assigned higher *priorities*. In addition, an activity is associated with the task *deadline* constraint specified in the workflow model, by which the data must be fetched since the user will need the information before executing the action.

As a result, a plan-tree node contains a predicted user-action (*e.g.*, Dispatch ambulance), *queries* specifying associated information requirements (which is annotated in the workflow model), *priority*, and *deadline*. For instance, based on the belief state that there is a toxic gas attack in a populous region, the agent recognizes that the user will need contact information of HazMat group as well as police for crowd control and orderly evacuation.

The updated plan-tree is then supplied to the Information Gatherer to revise information gathering plan.

C. Cognitive Workload Estimator

The Cognitive Workload Estimator (CWE) uses inputs from the cognitive workload model[2] and *deadline* specified in the workflow model. We use a queuing network based model for computing the workload of the user [4]–[6]. A standard queue-based model of mental workload treats the user's attention as the server and tasks as jobs (see [7] for details). As jobs arrive they are placed on the queue. Typically some simple discipline such as first in first out (FIFO) or priority-FIFO is used to manage the queue. The user's capacity is considered to have been exceeded when jobs on the queue exceed the user's ability to process them while avoiding tardiness. Based on the predicted plan-tree, the arriving tasks are determined, and the CWE maintains a running estimate of the probability that an operator will fail to complete a task by deadline. In the prototype implementation, we use a binary indicator such that the user cognitive workload is `overloaded` if the estimated probability of activity failure exceeds a certain threshold. Otherwise, the workload estimator returns `normal`.

D. Information Gatherer

Given a plan-tree of predicted information-gathering tasks, the Information Gatherer determines (or schedules) when and which information sources to use in order to satisfy the information needs of the user as well as coping with resource constraints (e.g., network bandwidth) imposed by the problem domain; specifically, the agent should not interfere with the user's planning activities by over consuming computing resources. Initially, the information-gathering tasks are ordered by the priorities and the deadlines, ensuring not only the acquisition of the most useful information, but also a timely acquisition of data to meet the deadline constraints. To accommodate changing information requirements, the Information Gatherer must optimize its current schedule incrementally to satisfy newer (thus more relevant) information-gathering constraints. The retrieved data is stored locally until used by the Information Presenter.

[2]The model development is outside the scope of this paper.

E. Information Adapter

The Information Adapter determines the level of detail when presenting the data received from the Information Gatherer to the user, considering both the user cognitive workload and the estimated time available to the user for the activity. Obtaining precise quantitative relationships between cognitive workload of a user and the information content in a document that can be processed by her is an open problem. Thus we design the Information Adapter to present the information at various levels of granularity so that the user has a choice of the kind of content she wants to read. In the current implementation, we classify information sources according to the granularity of data stored in the sources.

F. Information Presenter

Given a belief state representing the user's current state, the Information Presenter selects relevant information in the local storage and presents it to the user. In order to avoid information overload, the Information Presenter must only present data in temporal proximity to the actual need, with a sufficient time for the information to be intelligible to the user to meet her decision deadlines for the action at hand. Finally, user feedback (e.g., whether the presented information has been used) is collected and is provided for the agent as reinforcement in order to allow future improvements on the quality of supplied data.

V. Application description

This section illustrates the prototype application of ANTICO in the emergency response scenario described in Section II using a working example.

A. Graphical User Interface (GUI)

Figure 2 shows a snapshot of the operator interface annotated to explain the purpose of each panel. To display an interactive map, we integrate Google Maps[3] in our GUI.

The messages arriving from the field or other groups are shown in the *Event Messages* panel. In a real situation, these messages will arrive to the agent from actual sources (*e.g.* electronic messages from 911 operators). Here, the messages are introduced into the environment in order to simulate the real-time scenario. The time left to deal with the overall emergency situation is displayed in the top left part of the interface. The panels in the bottom of the interface display the following types of information:

- *Contact* information for organizations employed in emergency situations, *e.g.*, fire stations, local police, and facility security forces;
- *Diagnostics* for particular events;
- *Affected areas* and *vulnerable populations* such as schools and nurseries; and
- *Weather*: Weather information is useful since (a) it may affect the ability of response units to get to the emergency site, and (b) for certain types of events, for example during chemical attacks, winds can distribute the toxic chemical to large downstream areas, affected areas and vulnerable populations.

In addition, the *Reminder* panel (the leftmost in the bottom) provides reminders to the operator regarding special equipment or required capabilities for operation units. For example, the unit that evacuates vulnerable population requires special training for evacuating handicapped people. Note that the user may access information sources directly at any time using the organization panel on the right.

Finally, although not necessarily part of the interface seen by a human operator, our implementation also includes a "debugging" interface (shown in Figure 3) that allows one to visualize the internal belief state of the information agent. This visualization shows the workflow currently being used by the agent, including all activities and transitions between activities, as well as the agent's current belief state expressed as probabilities associated with each user activity. These probabilities represent the degree of belief possessed by the agent that a user is currently carrying out the activity in the workflow. Thus, the excerpt of a workflow shown in Figure 3 shows that the agent believes with $.88$ probability that the user is currently carrying out the *Dispatch Ambulance* activity in the Sarin partition of the workflow. At the same time the agent believes with $.12$ probability that the user is carrying out the same activity in the Yellow partition of the workflow (reflecting a small degree of uncertainty over the diagnostics performed earlier).

B. Working example

To illustrate the working of ANTICO agent, we use a simple example to describe in detail what the user sees in the GUI, what the agent performs in the background, and how the user makes decisions.

During peak hour at Union Station in Washington DC, a chemical attack takes place. The expected size of crowd in the proximity of attack is approximately $20,000$ with heavy traffic in the roads surrounding the station. For this particular scenario, we have created a workflow containing two similar partitions: one for responding to a Sarin Gas Attack; and one for responding to a Yellow Gas Attack. Although at a higher level these workflows are quite similar, the specific information and reminders that will be provided for the user will differ, since the symptoms and treatment for these two chemical agents are different. After reading the event messages regarding toxic gas, the user clicks on an area in the map. This observation triggers the Intent Recognizer module and the agent's belief state indicates that the user is determining the type of gas whether it is Sarin or Yellow. At this point, with limited information, the intent recognizer will have probabilities distributed more or less evenly among the states in the various partitions of the workflow. This reflects the uncertainty at the diagnostics phase of the response, since Sarin and Yellow have a certain amount of similar symptoms. Thus, as represented in the workflow of Figure 3, the first activities the agent infers that a user will be carrying out

[3]http://code.google.com/apis/maps/

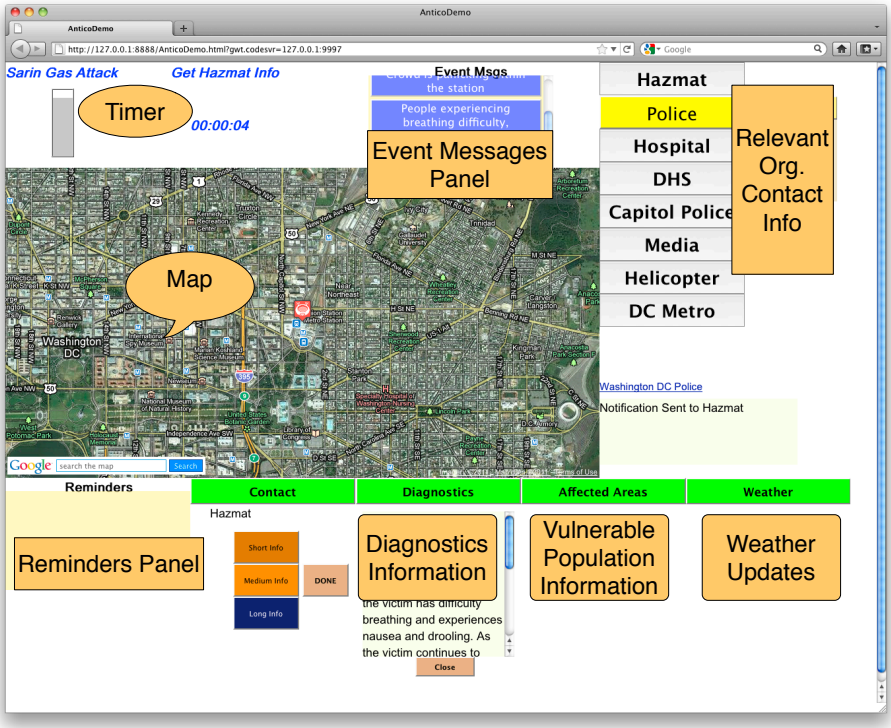

Figure 2. User interface for emergency response managers

relate to the determination of the nature of the attack and the symptoms reported from the scene. Accordingly, the type of information that the agent provides to the user at this point include a description of the symptoms of these chemical agents. Upon receiving a new set of event messages the agent's belief about Sarin gas is corroborated, and the agent immediately prefetches the diagnostics information of Sarin gas. Moreover, as the user peruses this information (which the agent detects as the user interacts with the information interface), the agent concludes that the user will soon want to dispatch a Hazardous Materials unit to the scene. Thus, the HazMat information nearby the affected area is retrieved.

At the same time, the agent presents HazMat information to the user as follows. Based on the workflow model, the optimal time for using HazMat information is calculated as two minutes. ANTICO presents the HazMat information in three levels of granularity (short, medium and long) classified based on the content size, and suggests the a level of granularity compatible with the predicted cognitive workload.

Figure 3 shows a visualization of the workflow available to the agent. Each activity is represented by a rectangle in the graph. Activities are connected by arrows in different directions; the numbers along the arrows represent the probabilities of the next activities. The number below each activity represents the estimated probability of the user performing the activity, *e.g.*, from the the agent's current belief state, the operator is mostly likely to be dealing with a Sarin gas attack. It also predicts that the operator's next action should be to "Dispatch Police" for crowd control, within the Sarin gas branch of the workflow.

Based on the actions taken by the user with the current information on the interface, the agent updates its belief state and keeps presenting the user with the required information in a new state within the time constraints imposed by the new events. For instance, as the emergency simulation progresses, ANTICO brings up information for live weather and traffic updates, vulnerable population and hospitals near the emergency site[4].

VI. Related Work

In this section, we review three state of the art emergency response systems used today and point out the differences.

A. Google Crisis Response

Google *crisis response manager*[5] is an online tool that collects fresh high-resolution imagery plus other event-specific data, and then publishes this information on a dedicated landing page. Different users are also allowed to add information related to the disaster at a given geographic location, making this tool a wiki-like disaster response tool. Although having updated information is an important element of effective disaster response, this kind of application can easily overwhelm a responder by a deluge of information coming from user updates. On the one hand, the plethora of

[4]ANTICO demo video: http://goo.gl/o186E
[5] http://www.google.com/crisisresponse/

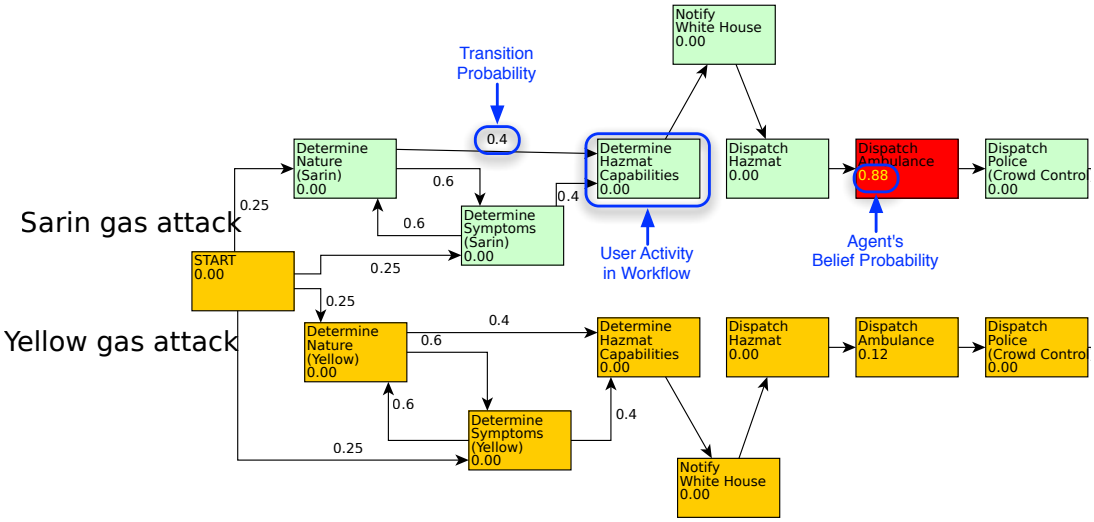

Figure 3. A fragment showing the user workflow model

information can result in the operator spending too much time in obtaining and absorbing the information; thus leaving less time to accomplish coherent response tasks. On the other hand, spending too little time in acquiring the information may lead to inferior task performance. Whereas Google crisis response aggregates large amounts of data, possibly jeopardizing user's responsiveness, ANTICO avoids this problem by monitoring the user's cognitive workload and providing an amount of information proportional to the user's ability to deal with it.

B. WIPER

WIPER provides emergency response managers with an integrated system that detects possible emergencies from cellular communication data, attempts to predict the development of emergency situations, and provides tools for evaluating possible courses of action in dealing with emergency situations [8]. WIPER conveys three distinct pieces of information to emergency responders via a web-based console:

- near-real time information on the location of cell phone users in an area, plotted on a GIS-based map of the area;
- potential anomalies, such as traffic jams, roving crowds and call patterns indicative of a crisis; and
- customized mitigation strategies, such as potential evacuation routes or barricade placement, suggested by computer simulations.

In order to provide such unobtrusive aid in context and in a timely manner, ANTICO signals to the user choices of information granularity that are likely to satisfy the effectiveness/timeliness tradeoff. WIPER, on the other hand, is an assistance system that helps a user plan to deal with a disaster. However, similarly to Google crisis response, WIPER does not deal with user information overload, and can provide too much information to the user. Thus, WIPER's capabilities for information gathering could be integrated in ANTICO's information gatherer module.

C. SOFER

Instead of providing proactive assistance, the Simulation-based Optimization of Fire and Emergency Response (SOFER) system [9] provides guidance on optimal allocation and placement of disaster response units to satisfy a set of service quality and availability criteria. SOFER uses a history of 911 requests to simulate future requests and determine how to distribute existing resources and the need to deploy additional resources in order to meet specific optimization criteria. It does not, however, provide on-demand assistance as disasters are occurring, but could be used in the context of an assistant to provide deployment suggestions for a user having to deal with multiple simultaneous disaster situations. Although SOFER is not an assistant *per se*, it provides optimal resource placement strategies that could be used to help mitigate a user's cognitive workload by suggesting deployment configurations as a disaster evolves.

VII. Discussion

In this paper, we presented the design and development of a software agent for managing the cognitive workload of a human decision maker, who has to make a sequence of decisions (or perform a sequence of activities) for time critical tasks in an emergency response scenario. We introduced the anytime cognition (ANTICO) concept, where the software agent uses an estimate of the users cognitive load and the available time for an activity, to manage the level of detail of information that it provides to the user, such that the user is able to absorb the given information in order to perform the activity (with acceptable quality) within time. Thus, ANTICO provides an adaptive way of reducing the information overload

of the user. In contrast to most information agents that are reactive, ANTICO proactively manages a user's information requirements, by predicting the future activities of the user.

To realize the ANTICO concept, we integrated several AI technologies including probabilistic plan recognition (intent prediction module), Bayesian reasoning (information adapter and information presenter modules), and constraint optimization (information gatherer module). Our agent architecture presented in this paper has two additional advantages. First, our approach is domain independent so that various types of applications can be developed from it. We proposed ADDL an XML-based language for describing the domain knowledge. As a proof of concept application, we demonstrated the ANTICO concept in the context of emergency response scenario. Second, our modular agent architecture provides flexibility in choosing the main algorithms used in each module without affecting the rest of the system. For instance, we plan to enhance the information adapter module by exploiting metadata to dynamically adjust the granularity of the data presented to the user, *e.g.*, extracting title, the first sentence, or abstract only. Such enhancements are transparent to other components since the specific algorithms used inside each module are independent.

REFERENCES

[1] J. Oh, F. Meneguzzi, and K. Sycara, "Probabilistic plan recognition for intelligent information agents," in *Proc. ICAART*, 2011.

[2] J. K. Benini, "National planning scenarios, version 21.3," March 2006, Homeland Security Council. [Online]. Available: https://www.llis.dhs.gov/docdetails/details.do?contentID=13712

[3] L. Rabiner, "A tutorial on HMM and selected applications in speech recognition," *Proc. of IEEE*, vol. 77, no. 2, pp. 257–286, February 1989.

[4] J. R. Carbonell, "A queuing model of many-instrument visual sampling," *IEEE Transactions on Human Factors in Electronics*, vol. 7, no. 4, pp. 157–164, 1966.

[5] K. C. Hendy, J. Liao, and P. Milgram, "Combining time and intensity effects in assessing operator information-processing load," *Human Factors*, vol. 39, no. 4, pp. 30–47, 1997.

[6] Y. Liu, R. Feyen, and O. Tsimhoni, "Queueing network-model human processor (qn-mhp): A computational architecture for multitask performance in human-machine systems," *ACM Transactions on Computer-Human Interaction*, vol. 13, no. 1, pp. 37–70, March 2006.

[7] C. D. Wickens, "Multiple resources and performance prediction," *Theoretical Issues in Ergonomic Science*, vol. 3, no. 2, pp. 159–177, 2002.

[8] A. Pawling, T. Schoenharl, P. Yan, and G. Madey, "WIPER: An Emergency Response System," in *Proc. ISCRAM*, 2008.

[9] R. Bjarnason, P. Tadepalli, A. Fern, and C. Niedner, "Simulation-based optimization of resource placement and emergency response," in *Proc. IAAI*, 2009, pp. 47–53.