Interactive Response System for Crisis Management

Design and organization of autonomous systems project

Tom van der Weide
tom.vanderweide@gmail.com

Siwei Wang
Swang@science.uva.nl

Ivo Everts
ivoeverts@gmail.com

Gijs Dubbelman
gijs.dubbelman@student.uva.nl

Abstract

This paper focuses on Distributed Perception Networks [3,4] initialization and incorporation of information coming from humans. In this paper we describe our results from the Design and Organization of Autonomous Systems project which is part of the Masters Multi Modal Intelligent Systems at University of Amsterdam.
For this project we designed, implemented and experimented with several extensions for the Distributed Perception Network (DPN), in order to use it in an interactive response system for crisis management.

1 Introduction

When there is a crisis, there will be lots of emergency troops trying to help. Police, fireguards and medical personnel will be all over the place. Accurate and up-to-date information is crucial for making good decisions. But often there are not enough sensors in an area to fulfill that need. However, there may be numerous people near the scene who could provide valuable information. For example, on the sixth floor of a building 2 out of 15 smoke detectors are being triggered. Are these detectors malfunctioning or is the sixth floor on fire? Perhaps a gas has been released that triggers the smoke detectors. Several scenarios could be possible, but none of them with a definite certainty. Our system detects that and contacts persons near the potential fire to gather more information. Those persons may be in another room not knowing what's happening. But there may be some who actually see the fire. They respond to the system and with that extra information, it is calculated that fire is the most probable cause. Within minutes the fireguards enter the building and extinguish the fire.

Our goal is to provide useful advice to experts in the control room or teams on location.

The following chapters are organized as follows. In chapter two our system design is discussed. Chapter Three covers the systems performance. Our conclusion can be found in chapter Four. We end with a discussion in chapter five.

1.1 Distributed Perception Network

A DPN is a multi-agent based approach to fusion of heterogeneous and distributed data. There are two types of nodes in the network: Sensor Agents (SA) and Fusion Agents (FA). The sensor value is represented by a Boolean value in the SA. A Fusion Agent fuses the values of several agents, either Sensor- or Fusion- agents. This results in a probability, for the Fusion Agent. In the DPN, Bayesian inferences can be made. So one can think of a DPN as a distributed version of a Bayesian network, where the Fusion Agents represent concepts, which are the *cause* of the concepts that are represented by Sensor Agents. So nodes in the network close to leaf nodes typically represent lower level concepts than concepts close to the root concept (e.g. the concept ‘concept’). All Agents may reside on geographically distinct locations. Communication is done via a Yellow Pages agent. [3,4]

2 System design

In order to get a feeling for what modules have to be designed to create a system like we discussed in the introduction, lets take a look at the flow of the system, as we would like it to be:

A certain sensor on a certain location registers a sudden change in value. This causes the Concept Manager Agent to create the DPN (for details see chapter 2.1). Then, a Human Agent is added to each Fusion Agent present in the DPN. The task of this Human Agent is to gather more information about the concept that is represented by the Fusion Agent. It does this by contacting the Callcenter. The Callcenter maintains a

priority queue of queries from Human Agents. It selects a query, and then determines which humans are suitable to contact in order to retrieve an answer to the query (naturally depending, among other properties, on the location of the human). It

contacts the humans by sending an SMS to the mobile phones of the selected humans. The Callcenter then gathers the answers and decides if the answer to the query is yes or no, and sends it back to the Human Agent that was responsible for sending the query to the Callcenter. The Human Agent then Incorporates this new information in the network. This flow of the system clearly identifies three modules that have to be designed: the Concept Manager, the Human Agents and the Callcenter. This chapter is devoted to these modules.

2.1 Concept Manager Agent

The DPN software currently only supports one kind of initialization method. This method relies one a human controller to select certain hypotheses and let the DPN spawn their corresponding world-models. The human controller can choose this initial hypothesis based on information such as a 911 call. This method can suffice for many real-world problems. The ideal situation however is that the DPN system itself can choose and spawn this initial hypothesis. As part of our research we have looked at possible initialization methods for a DPN.
The initialization methods can be divided into three groups: top-down, bottom-up and hybrid. In the next section the most important methods are briefly discussed.

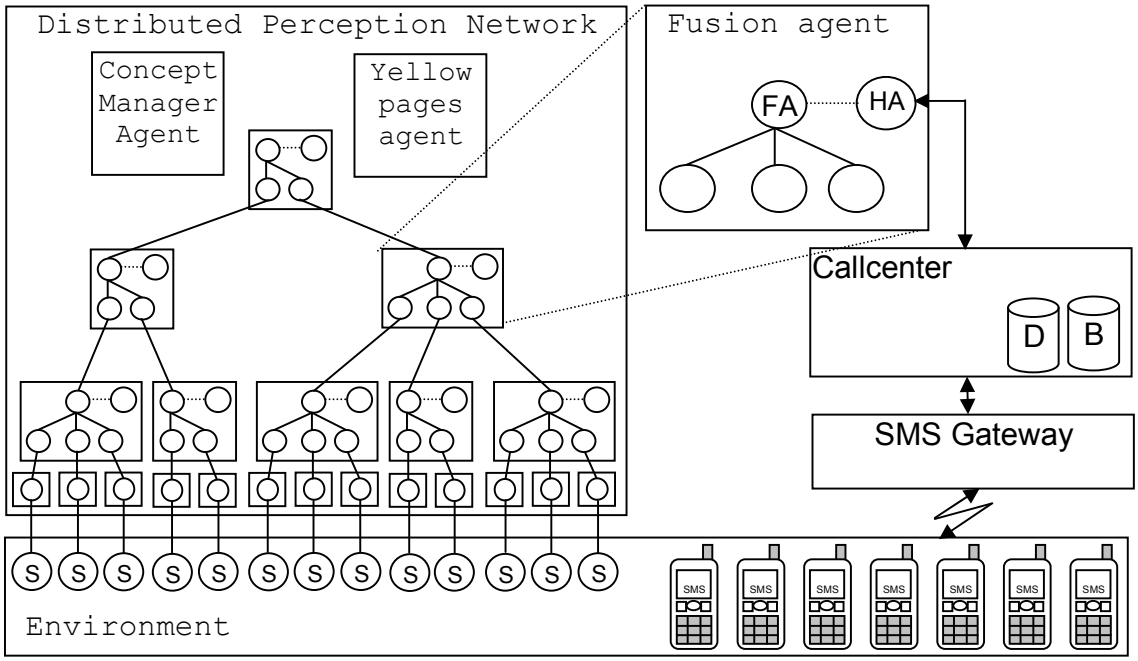

Figure 1, System design.

2.1.1 DPN Initialization methods

1) Human initiated top-down.
In this method a human controller receives a warning that a particular concept may be happening at a certain location. The controller then decides to let the DPN software spawn a world model for that concept at a certain location. As said before this method may suffice for many real-world applications.

2) Sensor initiated top-down
In this method an artificial intelligent manager takes the role of the human controller. This manager chooses a set of hypotheses to spawn based on alarm signals coming from sensors (this sensor also can be 911 calls interpreted by a speech recognition system). There is one big disadvantage of this approach. Namely that at some point the system must choose a set of hypotheses to spawn based on the initial sensor readings. For this choice we have to use some sort of pattern classifier. The problem with most pattern classifiers is that they heavily rely on examples. Because the DPN system is designed for disaster scenarios, good examples are very rare. Another approach is to use the native Bayesian networks of the DPN to calculate initial probabilities for the top concepts. The problem with this approach is that the initial evidence is very minute and can therefore not provide enough information for complex top concepts.

3) Sensor initiated bottom-up
In the following section this method will be discussed more thoroughly. The basic idea is that when a (sensor) agent receives enough evidence from its sensor(s)/children to say with certainty that its concept is not false, it activates all agents that have this (sensor) agent as its child. This method is applied until top concepts are reached.

2.1.2 Proposal

We propose a DPN design that can handle methods 1 and 2 combining the native top-down spawning with a new bottom-up method. It is based on the following assumption: Only a increase in certainty for the True value of a agent can contribute in the increase in certainty for the True value of its parent. Another assumption we make is that all agents are already present within the system or that they can be created on demand.

Our proposal adds a Concept Manager Agent (CMA) to the existing building blocks. The concept manager can be one program or it can be distributed throughout the system. The CMA facilitates a Bottom-up construction method for the DPN.

The CMA is able to receive messages of all agents that are assigned to this CMA (Assignment can be based on location).

The CMA stores the name , position and Partial world model for each agent assigned to this CMA. The CMA mechanism for bottom up construction is described below.

1) A sensor is monitoring its environment and sends its data to its Sensor-agent.
2) At a particular moment the sensor data may provide enough evidence for the sensor agent to set its value to true.
3) It then sends a message to the Concept manager. This message contains the concept that has become true. The sensor type, id and location.
4) The concept manager has a table of all agents that are present within the system. For every agent it also has stored its local world model. Based on this information the Concept manager can select all agents that have the particular sensor Agent as one of its possible children.
5) The Concept manager will then send a activation message to all these agents.
6) When the agents receive the activation messages they will spawn the world models that are needed to gather the information for this agent. This spawning uses the native DPN top-down approach [3,4].

7) When there is enough evidence to determine with certainty that the agent value is not false then again a message is sent to the CMA with the concept, agent type, id and location. and the process starts again from step 4. this goes on until concepts are reached that have no parents e.g. top-concepts.

This mechanism is very flexible. It builds upon the already existing top-down spawning and adds on top of this a Bottom-up spawning method. It is even possible to start spawning form an agent that is some where in the middle of a world-model. When this agent is activated (for example through a interpreted 911 call) it will gather its information in the usual top-down

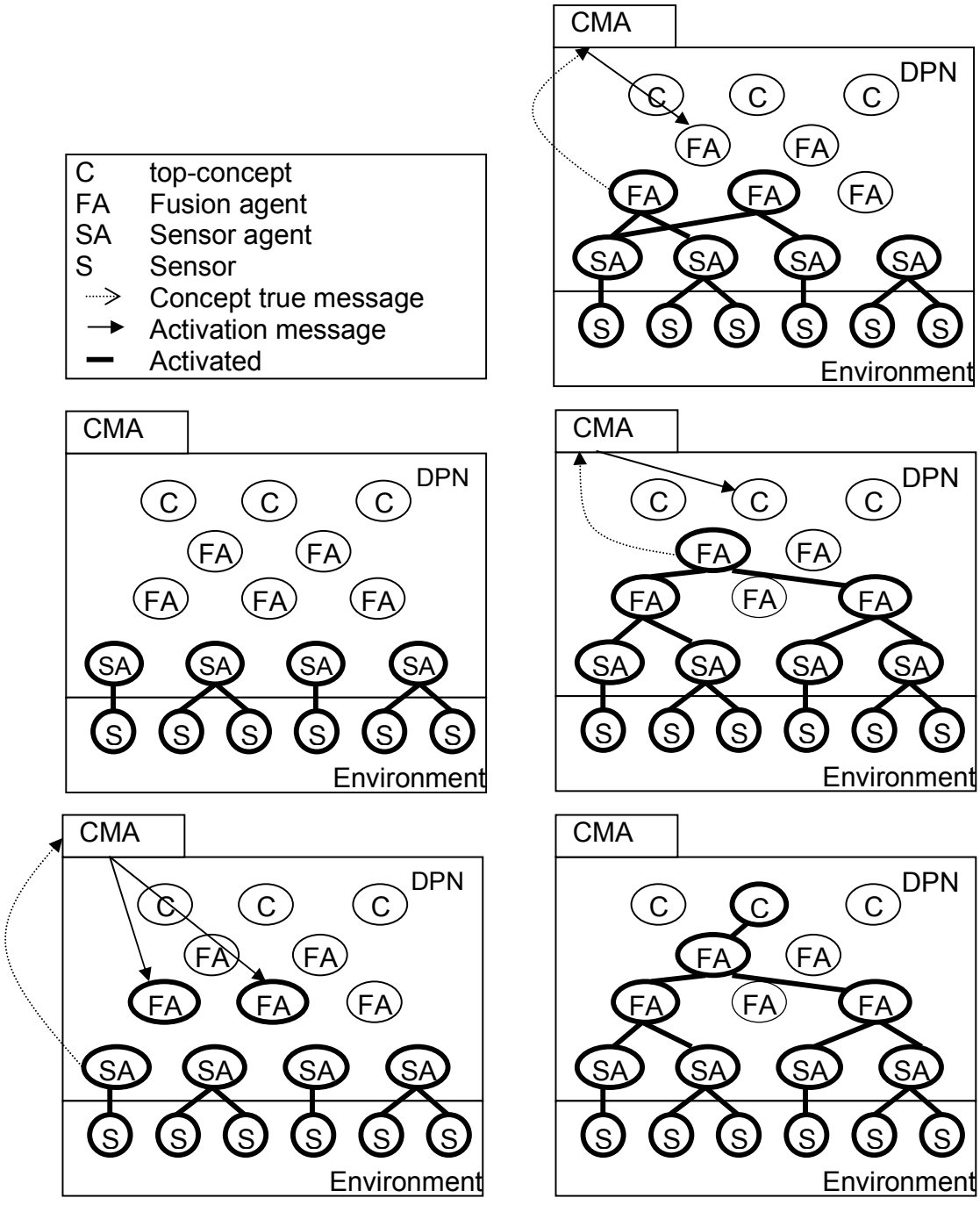

Figure 2, DPN initialization.

mechanism. When there is enough evidence to determine with certainty that the agent value is not false then again a message is sent to the CMA and starts the bottom-up mechanism.

2.2 Fusion Agents & Human Agents

The task of the Concept Manager was to spawn the correct DPN, given a change in sensor value(s). There is a preselection made in order to determine which concepts in the DPN need extra attention. This preselection is based on the certainty of a concept: if, for example, the concept 'fire' is true with a probability of 0.5, then we actually know nothing about it, and assign maximal uncertainty to it. In DPN-terminology this concept is in fact a Fusion Agent, since it fuses information from its children in the network and determines its own probability, given the child probabilities.

So now we know that each Fusion Agent that has passed the selection of the Concept Manager represents a concept of which we would like to have more information in order to be able to say that it is true or false. A plausible way of thinking about a concept is to view it as a hypothesis, e.g. think of the concept 'fire' as the hypothesis 'there is fire'. It then is the task of a Human Agent to gather information that answers the question 'is there a fire?'. For the design of the Human Agent we must be aware of the definition of agents:

"An agent is anything that can perceive its environment through sensors and act upon that environment through actuators. An agent that always tries to optimize an appropriate performance measure is called a rational agent." [1,2]

For each present Fusion Agent, a Human Agent is created. A Human Agent takes over the role of the fusion node in the Fusion Agent (the hypothesis). It then must interact with the Callcenter, in order to obtain information about the hypothesis. This makes the environment of a Human Agent two fold:

- A Human Agent should be able to perceive the Fusion Agent and update the probability of the hypothesis.
- A Human Agent should be able to perceive the Callcenter, and it should be able to send a query to the Callcenter.

The actionspace of a Human Agent consists of two actions: he can choose to send a query to the Callcenter, or he can choose not to do so. The goal of a Human Agent is to remove uncertainty about the hypothesis: if he knows that a hypothesis is true or false, its goal is reached and the performance measure should be maximal.

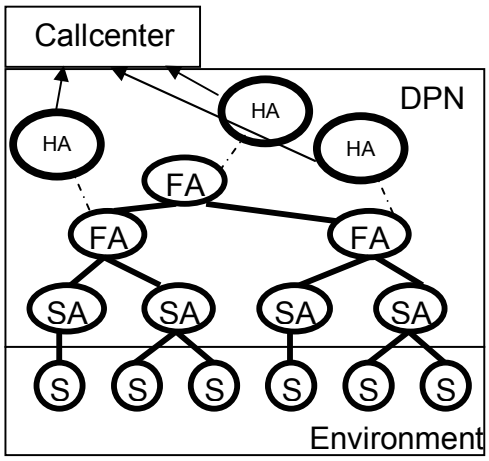

Figure 3, Human agents.

So one might wonder why a Human Agent should ever choose not to contact the Callcenter. This is because the queue of waiting queries from Human Agents can become a bottleneck of the system. Note that a real life application of this system might consist of thousands of Fusion Agents, which can all be activated at the same time. Since in our system the Callcenter gathers information via a SMS interface, it will not be a very good idea to send thousands SMS´s at the same time. This potential bottleneck affects the Human Agent´s preference of being in the world state in which his query is waiting in the queue of the Callcenter.

An agent´s preference for a certain world state is represented by a utility function. For a Human Agent, this function takes two parameters:

- The certainty of the hypothesis
- The queue that was mentioned above

The certainty of a hypothesis is computed by first defining a gaussian distribution over the possible probabilities of a hypothesis. We do this by setting the mean of the distribution on 0.5 and the standard deviation on 0.25. What we then get is actually a measure of *uncertainty*: a very low (e.g. 0.1) or very high (e.g. 0.9) probability leads us to belief with low uncertainty that a hypothesis is false or true, respectively. As we stated earlier in this section, a probability of 0.5 leads to maximal uncertainty, which is why this value is taken to be the mean of the distribution. In order to get a sensible certainty measure of a probability, we then substract the normalized uncertainty from 1:

$Uncert(p) = N(0.5, 0.25)(p)$
$Cert(p) = 1.0 - (Uncert(p) / Uncert(0.5))$

To be convinced that this is a sensible certainty measure, we verify that:

- $Cert(0.5) = 0$
- $Cert(0) = Cert(1) = 0.87$
- $Cert(0.2) = Cert(0.8) = 0.51$
- $Cert(0.3) = Cert(0.7) = 0.27$

This is in fact the function that the Concept Manager used to select uncertain concepts, with a 'certainty threshold' of 0.5: above this value there is enough certainty, whereas a lower certainty activates a Fusion Agent. Note that the normalization procedure makes sure that a probability of 0.5 results in a certainty of 0, but not that a probability of 1 (or 0) results in a certainty of 1. This is because the domain of the distribution is larger than the interval [0,1]. As long as a sensible threshold value is chosen, this does not matter.

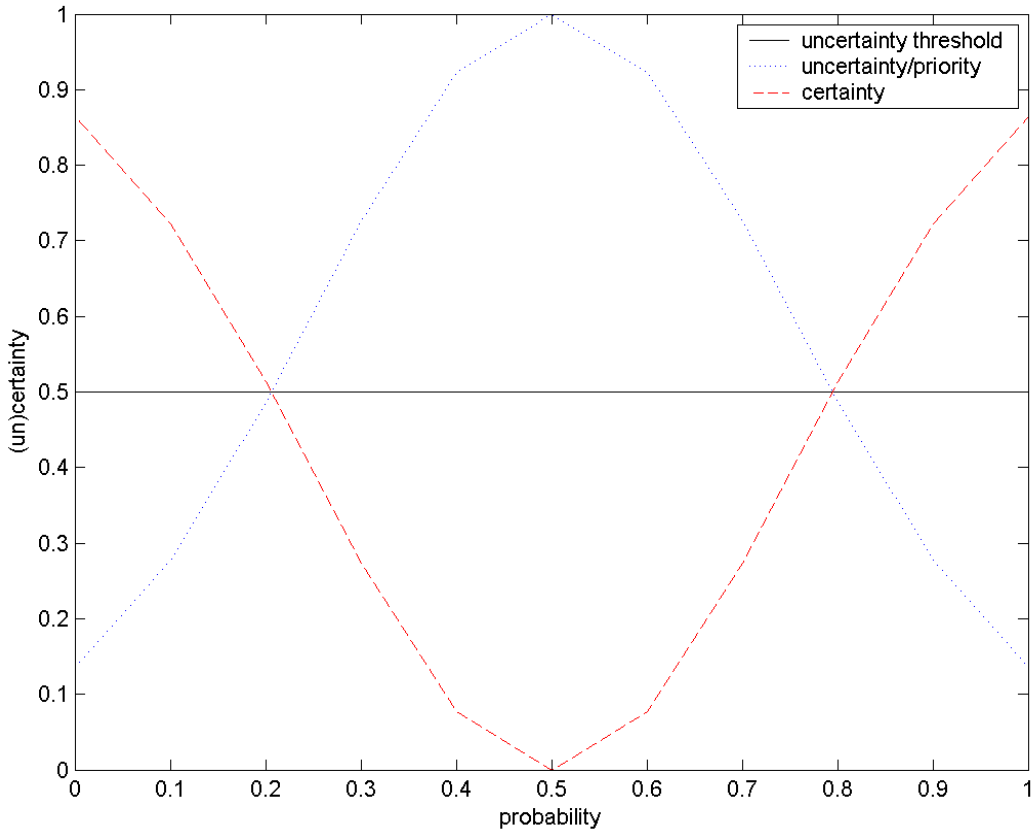

Figure 4, Uncertainty curve, Certainty curve and certainty threshold.

Since a preselection of concepts has already been made by the Concept Manager based on the above formula, it is not feasible for a Human Agent to use it in a similar way: we already know that the certainty is below the threshold. Therefore, a Human Agent gives a slightly different interpretation to the certainty measure. Instead of testing the certainty to a threshold, it uses the actual value of the *un*certainty as a *priority* measure. (High uncertainty → high priority)

Now we have all ingredients to construct the utility function of a Human Agent.

We compute the utility for an agent *i* by simply summing the priority of agent *i* and the inverse of the length of the queue:

Utility(i) = priority(i) + (1 / length(queue))

A threshold value is used to decide if the utility is high enough to contact the Callcenter. We have set this to 1: a hypothesis with maximal uncertainty / priority will always be sent to the data source, whereas a hypothesis with a lower priority may be sent, depending on the queue size.

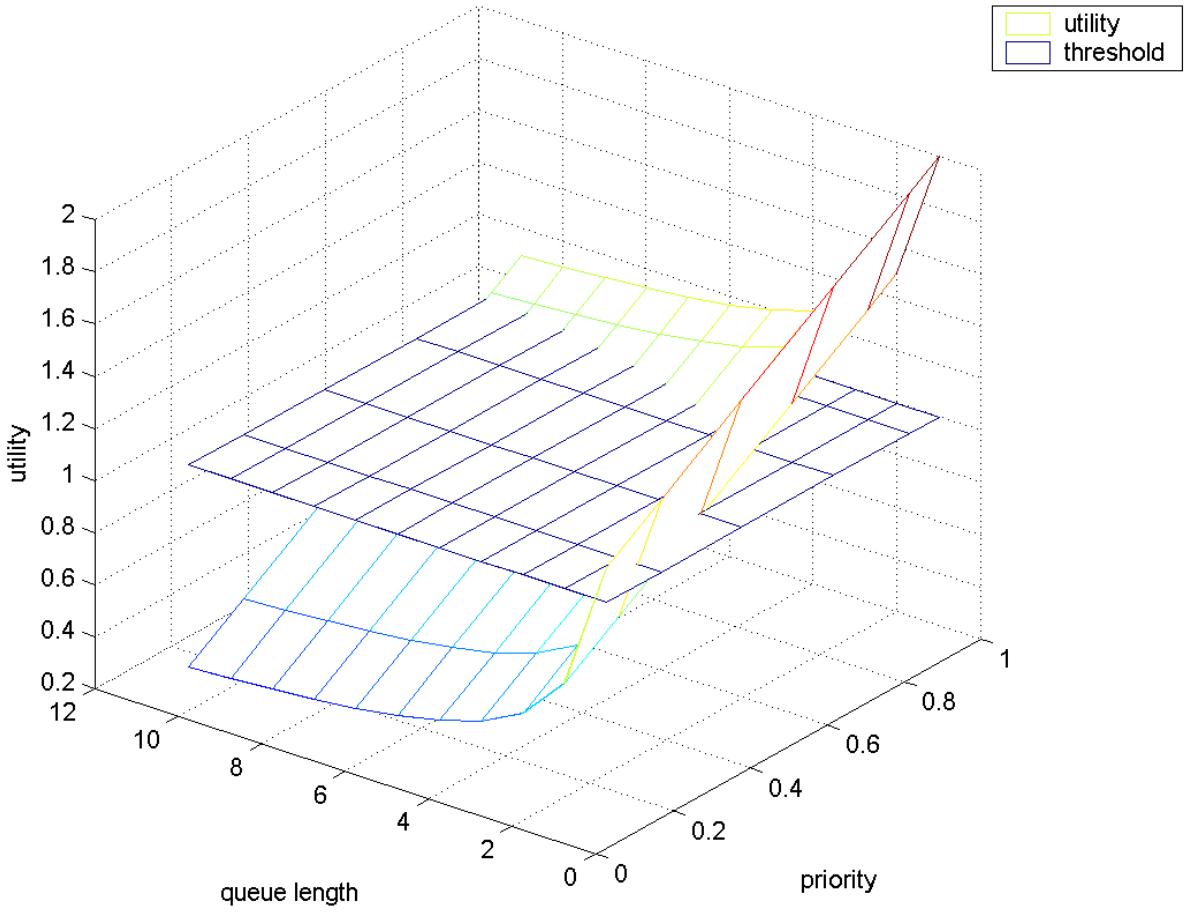

Figure 5, Utility function for a Human Agent.

2.3 Callcenter: Contacting humans

To make the system not only work on paper, but also in practice, there are quite some challenges that should be solved. A more final version of the system should be tested extensively with real humans in order to fine-tune the parameters. We came up with a list of challenges that we think can make or break the system.

(C1) flooding

An easy way would be contacting all humans after the slightest change. However, this could mean that you'll be contacted hundreds of times a day. We assume people will cooperate, but when they're flooded with questions, they won't respond to any of them.

	Alarm	No Alarm
Disaster	+	Very bad!
No Disaster	Annoying	+

(C2) people background

In some scenarios extensive knowledge is available about the causes. This knowledge can be very specific and detailed. The average person would not be capable of answering all these questions. Also, the response of a doctor to a medical question gives you more accurate information than the response of an 8 year old.

(C3) interpreting results

How many answers do you need in order to be certain enough about an outcome? Is it enough if only one person responds? Or would you like to have at least ten people confirming that outcome? And what would you do if half of the people confirm the question, and the others deny it? When there may be a fire in a building and a kid confirms it, but ten adults deny it, would you like to take the risk to assume there is no fire?

(C4) signal dominance

There is also the aspect of time. A smoke detector gets triggered and our system asks several persons if there is fire. The first ten minutes one hundred people respond that they don't see a fire. But in the eleventh minute there are ten positive responses coming in. If you would take the average, the outcome will be negative. But maybe it took some time for the smoke to travel through the building and only just now people are seeing it.

(C5) urgency

When a question comes in, we would like to ask those persons who would give the most accurate answer. But how many should we contact? In a hospital there are lots of persons with medical knowledge, but should we contact them all? Maybe we would already be satisfied if only five of them responded. So we contact five of them. But after thirty minutes only two responded. The need for that information was urgent, so perhaps if we had contacted ten, we would have gotten five results within a reasonable time.

(C6) how to contact

What is the best way to contact persons? Will yes-no questions suffice?

2.3.1 Our approach

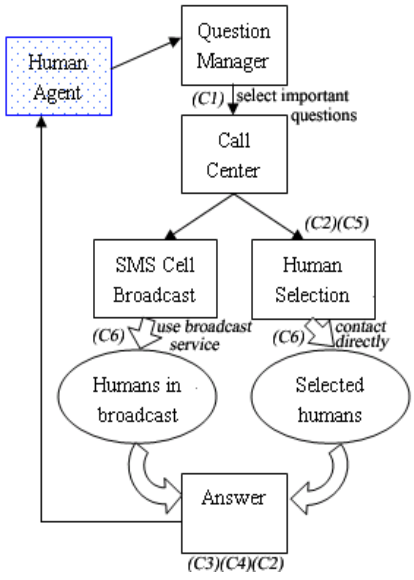

Figure 6, Callcenter design.

The question manager receives a question from the human agent. He selects the most important ones and asks the callcenter to get an answer. If only general knowledge is required, the question will be broadcasted. Otherwise experts who are near will be contacted directly. After a while responses are coming in and when the answer is certain enough, it will be sent to the Human Agent.

We tried to incorporate the challenges we've proposed as follows:
(C1) The Human Agent only sends the most important concepts. But there still may be vast amounts of concepts coming in. Since it's the callcenter's responsibility not to flood people, it needs to select too. Right now this is a simple threshold.
(C2) We assume it's known in which fields a person is expert and which expertise is needed for a concept. Given that information and additional information like age, we calculate the reliability of a response. A database is used to store that information and the location of the persons for easy access.
(C3) When the certainty of an outcome is at least twice as high as the negated outcome, the system decides it's certain enough.
(C4) This hasn't been implemented.
(C5) The urgency is a number between 0 and 1 which we use as a chance to contact an appropriate person.
(C6) SMS Cell Broadcast Service enables messages to be sent to multiple mobile phones located in a certain area. If we don't know who to ask or who are at the location of the disaster, the system will choose to broadcast the questions to a restricted area. After receiving feedback, the system will use background information about that person to calculate the reliability of the answer.

3 System Performance

For the purpose of testing our proposals we have implemented a basic DPN system. It can handle the native top-down construction as well as the new bottom-up construction. On top of this we added the Human agents and the Callcenter.

In order to test our implementation we made a small set of partial world-models, some top concepts and a sensor scenario.

Based on our test results we observed that the systems works as described in the previous chapters. Due to lack of time we could not test the system in great detail.

4 Conclusions

The introduction of Human Agents in a DPN to aid interactive response during a crisis is a good idea. From the performance of the system we see that the Human Agent does its job as it should, and therefore potentially supplying emergency troops with valuable information about the 'scene of crime'.

An emergent property of the system is that the Human Agents can detect a false alarm, once a negative answer to a query about a certain hypothesis is received.

An essential property that is maintained is that the system is very flexible. At runtime, the Concept Manager can create the DPN on the fly: at every given moment, a change in sensor values results in an update of the DPN, which may result in the activation of new Human Agents. The use of a DPN also is a good choice, since its essentially distributed architecture implies that the system will easily survive a crash of one or a bunch of Human- , Fusion- and/or Sensor-agents (the system degradates gracefully).

The problem with the creation of the DPN is the lack of good real-world DPN's and concepts. This problem is being addressed by other people within the DPN group at the time of writing this paper.

5 Improvements & Future work

The behaviour of a Human Agent still is subject to debate. In our system, the utility of a Human Agent is maximal when its hypothesis is true with a probability of 0.5. In this situation, the Callcenter is always queried and the priority of the query is high. This may result in undesirable effects. Consider for example the situation in which a hypothesis is true with a probability of 0.75, which is due to a malfunctioning of one of the sensors: it could be 0.9 if the sensor was functioning right. Still, the '50% hypothesis' has a higher priority, whereas it might be more usefull to turn it the other way around. A big advantage however, is that our Human Agents do not need to have any prior knowledge about the hypothesis that they represent. The above mentioned effect would require a Human Agent to have prior knowledge about the sensors, and which concepts are affected by them, in order to circumvent it.

We have chosen to let the utility of a Human Agent be dependant only on the uncertainty of a hypothesis and the length of the queue of the Callcenter. One might think of other things that can affect the utility like response time, specific properties of the location of the crisis, or allowing the queue to also have a negative effect since in our system a queue of size 1000 still contributes 1/1000 to the utility.

Further development should also focus on optimizing and generalizing the current mechanism of generating the DPN.

In order to optimize the callcenter we need a good way to evaluate how it actually performs. We would like to see a formula that gives a score based on the time it takes to get the right amount of answers, the certainty and how many people have been bothered.

(C1) It should be stored when and how often a person has been called. The higher this number, the higher the penalty of contacting this person again.

(C2) When one question needs expertise in 5 areas, can someone who has expertise in four of five give valuable information? We think so. In order to incorporate this, we propose using conventional Information Retrieval techniques like vectors to represent expertise and the in-product between vectors as a measure for the quality of an answer.

(C3) Our proposition is to associate a concept and the amount of risk you're willing to take. A domain expert should associate an amount of risk with each question. Tests in real life should give some values of how to weight the responses.

(C4) An algorithm which incorporates the wanted behaviour should be developed.

(C5) The chance that one will respond and the urgency should be incorporated. The more urgent it is, the more willing we are to bother people in order to get an outcome as quickly as possible. If you would like to take no risks, then you could already give an outcome after two positive responses.

(C6) An SMS is easy to send, but sending a response is tedious. Should we be satisfied with yes-no questions? Knowing the expertise of everyone near the event is a handy heuristic, but how realistic is it? A lot of privacy concerns are popping up. A talking computer which calls is a friendlier way of communication and could retrieve more information. Instead of asking one question, it could ask to describe what's happening.

The CMA has some overlap with the yellow pages agent. The yellow pages agent also stores information about all concepts that are present within the system. In a final implementation one can decide to combine the functionality of the Concept manager agent and the Yellow pages agent into one distributed agent.

We implemented our proposals in self-made DPN system based on JavaBayes inference graphs the next step would be to implement our proposal into the real DPN software.

References

[1]. N. Vlassis (2003). *Multi-agent Systems and Distributed AI.* University of Amsterdam
[2]. S. Russel, P. Norvig (1995). *Artificial Intelligence, a modern approach.* Prentice Hall
[3]. G. Pavlin, M. Maris, J. Nunnink (2004). *An Agent-based approach to Distributed Data and Information Fusion.* University of Amsterdam
[4]. *Agent Architecture for Distributed Perception Networks* (tracking number: 470)

SMS interface

The system uses SMS messages to interact with human observers. For this project we opened a two way SMS service with Clickatell. We can send and receive messages by uploading and downloading text files form a ftp server. We tested the interaction with a java ftp-client (JFTP, see http://j-ftp.sourceforge.net/). We observed that the SMS service works reliable and fast.