Situation Determination with Distributed Context Histories

Graham Thomson, Paddy Nixon, and Sotirios Terzis
Global and Pervasive Computing Group, Department of Computer and Information Sciences
University of Strathclyde, Glasgow, UK

ABSTRACT
Determining the situation within an environment is a key goal of smart environment research. A significant challenge in situation determination is reasoning about open-ended groups of people and devices that a smart environment may contain. Contemporary solutions are often tailored to the specific environment. In this position paper, we present a novel general situation determination framework, that by viewing people and tools as playing roles in a situation, can easily adapt recognition to incorporate the dynamic structure of a situation over time.

INTRODUCTION
Determining the situation within an environment is a key goal of smart environment research. It provides a natural pivot to which users and application programmers can associate behaviours, such that the computing machinery contained within the environment silently and automatically adapts to its inhabitants' behaviours, "invisibly enhancing the world" [1].

The approaches to situation determination offered by the state-of-the-art context-aware infrastructures [2, 3] experience the following drawbacks:

- An expert of the particular environment is required to specify the correlation of the available context information with the situations that occur. Reasoning is performed by large logic programs [3] or Bayesian networks [2], which must be manually constructed and maintained.
- As the amount of available context information and number of situations increases, it becomes increasingly difficult for an expert to decipher and specify correlations.
- The situation specifications will suffer from the subjective bias of the expert who programmed them.
- Recognition is limited to the fixed number of cases programmed by the expert for the local environment, and does not adapt well to the introduction of unrecognised people or devices in the environment, that is, when the structure of the situation is uncertain.

This paper presents a novel approach to situation determination that attempts to address these issues. There is no need of an environment expert, as situations are programmed by example by users themselves. Reasoning is based upon a general situation determination framework that can be applied in any smart environment. By viewing people and tools as playing roles in a situation, recognition can easily adapt to incorporate the dynamic structure of a situation over time.

To illustrate the kind of situations we wish to detect, listed below are examples of typical situations that occur within our department, along with a description of their characteristics.

Project meeting People that are members of a project are assembled in a meeting room.

PhD meeting A PhD student and their supervisor are talking in the supervisor's office.

Conversation Two or more people are talking in the same area.

Presentation An audience is assembled in a meeting room, a projector is running, and presentation software is running. The host introduces the speaker(s), the speaker(s) present, and then the speaker(s) answer questions posed by the audience.

Checking mail A person is working with mail reader, web browser, and document reader tools, with the mail reader tool being used most frequently.

Reading A person is alone at their desk, and the computer or any other tools in the desk area are idle.

Coffee break The time is around either 11 or 4 o'clock, and a group of people are assembled in the staff room, drinking coffee.

Party In the late afternoon or evening, a large group of people are gathered in a conference room, with music playing and a projector displaying photographs.

From considering the situations presented above in addition to other situations from domestic and social scenarios, we propose that a situation can be robustly characterised by the combination of four main aspects:

Figure 1. The situation tree.

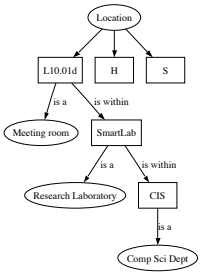

Figure 2. The Location subtree.

- The time at which the situation occurs, as well as its duration.
- Where the situation occurs, and the properties of that location.
- The attributes of the group of people that are present.
- The group of tools that are present, and the manner in which they are being used.

Furthermore, we propose that to accurately identify the elements of a situation that change over time, analysis of only the order and proportion of those elements is sufficient.

In the rest of this paper, we go on to describe the challenges faced when attempting to determine the situation, and present how we deal with these challenges in our approach to situation determination.

REPRESENTING THE SITUATION

An ontology based approach is used to represent the information that characterises a situation. Doing so permits matching at various levels of abstraction, information to be exchanged and interpreted correctly between multiple participants, and the information to be translated to different ontologies that may be used in other environments.

Figures 1 through 6 shows an example ontology describing a small presentation situation. The ontology is structured as a tree. At the root of the tree in Fig. 1, is the Situation class, which contains four other classes - Time, Location, People, and Tools.

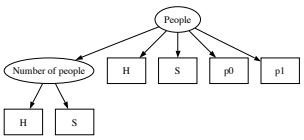

Figure 3. The People subtree.

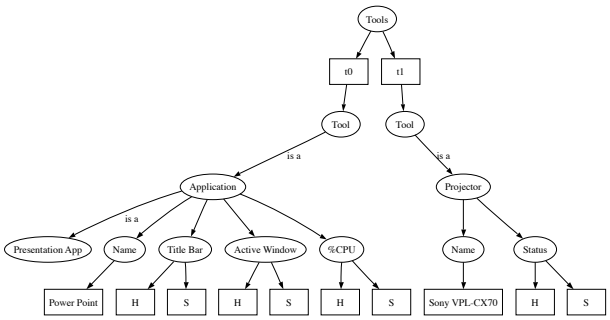

Figure 4. The Tools subtree.

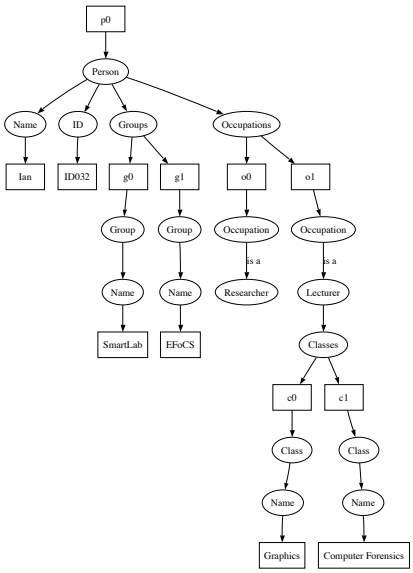

Figure 5. The p_0 subtree.

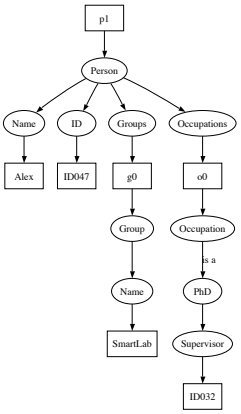

Figure 6. The p_1 subtree.

Time is the simplest aspect in the tree and includes values such as the time of day, day of week, date of month, etc. The tree is not shown in the interests of space.

To represent location, a symbolic, hierarchical model is used. That is, locations are referred to by name, and may be spatially contained in other locations. For example, Fig. 2 shows that 'SmartLab' is contained in 'CIS'. Each location is associated with a class that represents the type of location it is. For example, the location 'SmartLab' is a Research Laboratory.

As there may be a group of people present in a particular situation, each person is given a label, shown as p_0 and p_1 in Figs. 3, 5, and 6. These labels are utilised in matching, as we shall see later. From each person instance stems useful information that will help to identify a situation, such as their occupation, the groups they belong, and so on. For some situations such as a conversation, it is simply the number of people present, not their individual identities that are characteristic. A Number of People class is included to capture this explicitly.

Each tool that is involved with a situation is also given a label, shown as t_0 and t_1 in Fig. 4. Tools can include physical devices such as projector or a whiteboard, as well as computer applications. Computer devices are not regarded as tools themselves, as their use is characterised by the software applications that they host.

For each class in the ontology, its variables shall be either static or dynamic. If a variable is static, it is the value of the variable that characterises a situation. For example, in Fig. 5, 'Ian' is the value of the static variable 'Name'. If a variable is dynamic, it is the change in its value that characterises a situation. For example, in Fig. 4, the variable '%CPU' is dynamic. To capture the change in value, the sequence and histogram of values is stored for a dynamic variable. A sequence captures the order of the values, a histogram captures the proportion of the values. In the figures, dynamic variables can be identified by their Sequence (S) and Histogram (H) siblings.

Recent work has shown success in using dynamic Bayesian networks to classify low-level actions or activities of users, such as holding a telephone handset, or adjusting a thermostat [4, 5]. However, applying such techniques to recognising a situation, where the number of variables is very large and may change continuously (the structure is dynamic), and the situation's duration may be an hour or more, would be impractical. We suggest that such techniques are ideal for generating values of variables in the situation tree. For example, our current implementation uses a hidden Markov model to infer the current location.

SITUATION DETERMINATION

In our approach, situations are captured by the users themselves. We envisage a user running a client that requests a label for the situation, as well as the start and end times. During this period, the situation tree for the current environment (the context history) is recorded. Currently, we define the environment to be the room the user is in.

When determining the situation, we are comparing a situation tree that reflects the current state of the environment to a situation tree that has been previously captured and labelled. We shall refer to these as an example situation and the current situation respectively.

For each class defined in the ontology, there exists a comparison function that returns the similarity of two instances of that class. In this way, the similarity instances can be given the most appropriate score according to their class. For example, as the string 'Alex' is an instance of the class Name, it would have a greater similarity when compared to 'Alexander' than to 'Alan'.

It is the open-ended number of entities that makes situation determination challenging, specifically matching groups of entities. When a situation contains an open-ended group of objects, its structure is dynamic. As a situation is strongly characterised by the groups of people and tools it contains, an efficient method to compare groups is required. Consider the presentation application tool that is part of the Tools group in Figure 1. It is PowerPoint, and let's assume its dynamic properties show it is running lightly continuously. In an example situation it may be OpenOffice Impress that is running lightly continuously. What we are interested in is finding the tool that is playing the *role* of a presentation application running lightly continuously. Similarly in a PhD meeting situation, within the group of people present a person plays the role of a supervisor, while another plays the role of a student. Any group can be viewed as a set of roles. The problem of matching a group is then finding which members of a group from the current situation best fit the roles defined by the group in an example situation.

In Figs. 3, 5, and 6 the two instances of Person are labelled p_0 and p_1. These labels identify the role that that person plays. To compare a group, we have to use a more sophisticated compare function than that described previously. To illustrate, we shall consider the case of matching two People groups. Let ex.P be the People group from an example, and c.P be the People group from the current situation. The algorithm for matching groups is then:

1. For each ep_i in $ex.P$ and cp_j in $c.P$, compute the similarity of ep_i and cp_j, $sim(ep_i, cp_j)$.

2. Construct all possible mappings from the elements of $c.P$ to $ex.P$ where either every element of $ex.P$ is mapped to by a single, distinct element from $c.P$, or every element of $c.P$ maps to a single, distinct element in $ex.P$.

3. Calculate the score for each mapping by taking the average of the $sim(ep_i, cp_j)$ scores of each map within it.

4. The mapping which has the highest score gives the score for the group as a whole, as well as the optimal mapping of roles.

When comparing large groups, step 2 may become prohibitive. In cases where each member of a group contains only static properties, the computational cost can be reduced. To achieve this, each group member is given a unique id. When two group members are compared, their unique ids are compared first. If the unique ids match, we can shortcut the rest of the comparison of the two members as we know we have an exact match.

Members identified by their unique id can be confidently assigned to a particular role. Then, it is only the group members whose unique id is not recognised that must be assigned, substantially reducing the number of mappings that must be constructed.

Matching 'in vivo'

When an example situation is captured, the information in the situation tree is recorded over a length of time. When situation matching is performed, it must be done *as the situation is unfolding*. If only a single situation tree were used to capture an entire situation, we could only accurately match it at the end of the situation. Therefore, situation trees are captured periodically throughout the duration of a situation. The sequences and histograms of dynamic variables store the changes in value of the variable from the beginning of the situation up to the end of the period.

Distributed matching of situation fragments

So far we have looked at the situation tree as a whole. The information contained within the tree shall come from several different sources. It is therefore undesirable, and unnecessary, to have to collate the information in one place to perform matching.

In our approach, each person and tool has a corresponding software agent. The agent observes the information in, and performs matching on, the fragment of the situation tree that represents the person or tool. No single agent has a view of the entire situation tree, it can only see its own fragment. For example, in Fig. 5, the agent representing Person 'Ian' would see only the subtree starting at p_0, and in Fig. 4 the agent representing the Power Point instance, would see only the subtree starting at t_0. There is also an agent that represents the current environment, which we refer to as the situation server (SS).

When a tool or person enters a new location, its agent alerts the SS for that location to its representative's presence. The communication links between the SS and each agent in the environment form a star topology. Each agent stores locally applicable fragments of example situations. The agent compares the current situation fragment to each example fragment, making a list of all (fragment label, score) pairs. This list is then sent to the SS.

When the SS has received a list from each agent, it must combine these fragment scores into a score for that situation tree as a whole. The list from a Person agent will include the score of the agent's representative against each Person role in a situation, likewise for a list from a Tool agent. Based on these lists, the SS executes the group score / role assignment algorithm for the People and Tools groups. The SS then computes the scores for Location and Time and combines these to produce the total scores for each situation. The SS then sends a message back to each agent in the environment, indicating the highest scoring situation.

FUTURE WORK

In this paper it has been assumed that people in the same location are in the same situation. This will not always be the case. For example, in an open plan work area, some people may be involved in a conversation, while others may be working at their desk. Recognition could be extended to differentiate between these. Furthermore, some situations such as 'Journey home' will be characterised by a sequence of locations. In this case it may be inappropriate for a Person agent to attempt collaborative determination at each location.

As the number of situation examples increases, each agent will have a greater number of fragments to match. After an example situation is captured, a clustering phase could be introduced to reduce the total number of situation fragments.

In some situations, such as outdoor situations, it may not be possible to host a dedicated SS. In such cases, the agents involved in a situation would participate in an election protocol to identity the most suitable agent to act as the SS.

We are currently experimenting with our approach on simulated data, for which the initial results look promising. We are also evaluating different approaches to combining examples of the same situation, as well as opportunities for unsupervised learning. We are also working on the development of the necessary matching agents, which will provide a prototype system allowing us to experiment with real-time situation determination.

REFERENCES

1. M. Weiser. The computer for the 21st century. *Scientific American*, pages 94–104, September 1991.
2. A. Ranganathan, J. Al-Muhtadi, and R. H. Campbell. Reasoning about uncertain contexts in pervasive computing environments. *IEEE Pervasive Computing*, 3(2):62–70, 2004.
3. H. Chen, T. Finin, and A. Joshi. A context broker for building smart meeting rooms. In *Proceedings of the Knowledge Representation and Ontology for Autonomous Systems Symposium, 2004 AAAI Spring Symposium*. AAAI, March 2004.
4. Matthai Philipose, Kenneth P. Fishkin, Mike Perkowitz, Donald J Patterson Dirk Hahnel, Dieter Fox, and Henry Kautz. Inferring Activities from Interactions with Objects. In *IEEE Pervasive Computing: Mobile and Ubiquitous Systems*, volume 3, pages 50–57. IEEE, 2004.
5. Kenneth P. Fishkin, Bing Jiang, Matthai Philipose, and Sumit Roy. I sense a disturbance in the force:

Unobtrusive detection of interactions with rfid-tagged objects. In *Ubicomp*, pages 268–282, 2004.

Author Biographies

Graham Thomson is a PhD student in the Department of Computer and Information Sciences at the University of Strathclyde. He received his BSc in Software Engineering from the University of Strathclyde. Contact him at the Univ. of Strathclyde, Dept. of Computer and Information Science, Livingstone Tower, 26, Richmond Street, G1 1XH Glasgow, Scotland; Graham.Thomson@cis.strath.ac.uk.

Sotirios Terzis is a lecturer in the Department of Computer and Information Sciences at the University of Strathclyde. He received his PhD in the Computer Science Department at Trinity College Dublin. His research interests include context-awareness and trust management in pervasive computing systems. He received his BSc and MSc with a specialization in distributed systems from the University of Crete. He is a member of the ACM, the IEEE Computer Society, and the British Computer Society. Contact him at the Univ. of Strathclyde, Dept. of Computer and Information Science, Livingstone Tower, 26, Richmond Street, G1 1XH Glasgow, Scotland; Sotirios.Terzis@cis.strath.ac.uk.

Paddy Nixon is professor of computer science at the University of Strathclyde where he leads the Global and Pervasive Computing Group. He received his BS and PhD in computer science from Liverpool University and his MA from Trinity College Dublin. Contact him at the Univ. of Strathclyde, Dept. of Computer and Information Science, Livingstone Tower, 26, Richmond Street, G1 1XH Glasgow, Scotland; paddy@cis.strath.ac.uk.