Proceedings of the 4th International Workshop on

Modeling and Reasoning in Context (MRC 2007)

with Special Session on the Role of Contextualization in Human Tasks (CHUT)

Anders Kofod-Petersen
Jörg Cassens
David B. Leake
Stefan Schulz
(Editors)

AUGUST 2007 ROSKILDE UNIVERSITY COMPUTER SCIENCE RESEARCH REPORT #112

Computer Science
Roskilde University
P. O. Box 260
DK–4000 Roskilde
Denmark

Telephone:	+45 4674 3839
Telefax:	+45 4674 3072
Internet:	http://www.ruc.dk/dat/
E-mail:	datalogi@ruc.dk

ISSN 0109–9779

This research report constitutes the proceedings of the *4th International Workshop on Modeling and Reasoning in Context (MRC 2007) with Special Session on the Role of Contextualization in Human Tasks (CHUT)* which is held in conjunction with the 6th International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT 2007), Roskilde University, Denmark, August 2007.

Research reports are available electronically from:

http://www.ruc.dk/dat/

MRC 2007

Fourth International Workshop on
Modeling and Reasoning in Context

with

Special Session on the
Role of Contextualization in Human Tasks (CHUT)

Workshop at the
Sixth International and Interdisciplinary Conference on
Modeling and Using Context
Roskilde University, Denmark, 20-21 August 2007

Editors:
Anders Kofod-Petersen
Jörg Cassens
David B. Leake
Stefan Schulz

Preface

Context sensitive processing plays a key role in many modern IT applications, with context awareness and context-based reasoning essential not only for mobile and ubiquitous computing, but also for a wide range of other areas.

From an intelligent systems perspective, one of the challenges is to integrate context with other types of knowledge as an additional major source for reasoning, decision-making, and adaptation and to form a coherent and versatile architecture. There is a common understanding that achieving desired behaviour from intelligent systems will depend on the ability to represent and manipulate information about a rich range of contextual factors.

The MRC workshop series aims to provide a forum for scientists and practitioners exploring modelling and reasoning issues and approaches for context sensitive systems, from a broad range of areas, to share their problems and techniques across different research and application areas.

This year's workshop is the fourth workshop in the MRC series. This year's workshop is held in conjunction with the Sixth International and Interdisciplinary Conference on Modeling and Using Context (CONTEXT 2007) in Roskilde, Denmark. We received nine submissions for the workshop. Each submission was reviewed by at least three programme committee members. The committee decided to accept five papers for presentation and inclusion in this proceedings. In addition a special track on the Role of Contextualization in Human Tasks (CHUT 2007), containing three papers, was included.

Ye et al. describes how lattice theory can be utilised to organise situations. The low-level context information typically available is often brittle and cannot be used directly in applications without interpretation. Ye at al. demonstrates how lattice theory can represent situations with various degrees of generality, and how backward and forward chaining can be used to identify ongoing situations. Finally, the authors demonstrate how Bayesian Networks can be used to overcome uncertainty when reasoning about situations.

Know et al. reports on the Scatterbox system, which delivers messages to users' mobile devices depending on their context. The aim of this work is to take multiple heterogeneous sources of contextual data and infer the situations that they map to, thus allowing the system to infer when to deliver messages and where. Know et al. further describe a formal definition of context and situations. Finally, they describe the workings of the Scatterbox, as well as an example of how it can deliver messages depending on the user's ongoing situation.

Kofod-Petersen and Petersen describes the use of stereotypes for user models in mobile collaborative ambient intelligent systems. This work focuses on the fact that ambient intelligent systems must include a user model, and that a heterogeneous user group requires specific user models. The authors describe how a stereotype user model can be integrated with an existing knowledge model for ambient intelligent systems. Finally, they illustrate how the stereotypes can be used to construct tasks for mobile collaborative learners.

Hong, Schmidtke and Woo discuss the use of an ontology based context model for combining context models and contextual reasoning. The model focuses on the who, when, where, what, how and why dimensions of a user, which provide the primitive relations for contextual reasoning. The authors deliberate on the syntax and semantics of their logical language used to reason about context. Finally, they evaluate the context model and reasoning mechanism with regards to expressiveness and inferential power.

Vieira et al. report on how a Context-Oriented Model can deal with the problem of modelling context. The authors suggest a three layered model, based on the notion that contextual elements cannot be defined a priori. The three layers consist of meta-model, a domain specific model and application specific instances. Vieira et al. describe how their model can be used in a conference going example.

Brézillon, Brézillon and Tijus describe their work on improving situation awareness in novice drivers. By combining driving situations, drivers' behaviour into scenarios, the authors are able to simulate normal, pre-critical and critical situations. By applying machine learning techniques to questionnaires, Brézillon et al. arrive at four steps in their model of evolution of drivers. These data are used in a case study revolving around crossroads.

Zacarias, Pinto and Tribolet approach enterprise modelling, which normally can represent an organisation's design, by using a context-based approach to discover and model inter-personal work practices. The authors focus on the actions that are executed in work situations, and capture these in an ontology. Zacarias et al. demonstrates the usefulness of their model be applying it to two case studies concerning software development and furniture retailing.

Cruz et al. report on their work on contextualising existing software reuse repositories. The main purpose of this work is to improve the assembling of software assets by improving the semantics of their descriptions in the repository.

We would like to thank the organisers of the Sixth International and Interdisciplinary Conference on Modeling and Using Context 2007 for supporting this year's workshop. In addition, we would like to thank Andrei Voronkov for his fantastic EasyChair system.

June 2007

Anders Kofod-Petersen
Jörg Cassens
David B. Leake
Stefan Schulz

Organisation

Programme Chairs

Anders Kofod-Petersen, Norwegian University of Science and Technology
Jörg Cassens, Norwegian University of Science and Technology
David B. Leake, Indiana University, USA
Stefan Schulz, The e-Spirit Company, Germany

Programme Committee

Agnar Aamodt, Norwegian University of Science and Technology, Norway
Steven Bogaerts, University of Indiana, USA
Patrick Brézillon, University of Paris 6, France
Hans-Dieter Burkhard, umboldt University Berlin, Germany
Lorcan Coyle, University College Dublin, Ireland
Mehmet H. Göker, PriceWaterhouseCoopers, USA
Dominik Heckmann, DFKI, Germany
Eyke Hüellermeier, University of Magdeburg, Germany
Ana G. Maguitman, University of Indiana, USA
Marius Mikalsen, Sintef, Norway
Enric Plaza, IIIA-CSCI, Spain
Thomas R. Roth-Berghofer, DFKI, Germany
Sven Schwarz, DFKI, Germany
Santtu Toivonen, VTT, Finland

CHUT Programme Chairs

Patrick Brézillon, University of Paris 6, France
Charles Tijus, University of Paris 8, France
Juliette Brézillon, University of Paris 6, France

CHUT Programme Committee

Frédéric Adam, University of Cork College, Ireland
Thierry Artières, University of Paris 6, France
Jean-Michel Dalle, University of Paris 9, France
Patricia Delhomme, INRETS, France
Irina Ezhkova, IIAT, Belgium
Avelino Gonzalez, Universiity of Central Florida, USA
Robert R. Hoffman, IHMC, USA
Jacques Naymark, Savoirs-Interactifs, France

Cécile Paris, CSIRO, Australia
Simon Parsons, Brooklyn College, USA
Daniel Racoceanu, SHNU, Singapore
Farida Saad, INRETS, France
Ana Carolina Salgado, UFPE, Brazil
Marcus Specht, Open University, The Netherlands
Mark Suskin, NSF, USA
Roy Turner, Univesity of Maine, USA
Teun Van Dijk, University Pompeu Fabra, Spain

Table of Contents

Modeling and Reasoning in Context

The Role of Contextualization in Human Tasks

Using Situation Lattices to Model and Reason about Context*

Juan Ye, Lorcan Coyle, Simon Dobson, and Paddy Nixon

System Research Group, School of Computer Science and Informatics,
UCD, Dublin, Ireland
juan.ye@ucd.ie

Abstract. Much recent research has focused on using situations rather than individual pieces of context as a means to trigger adaptive system behaviour. While current research on situations emphasises their representation and composition, they do not provide an approach on how to organise and identify their occurrences efficiently. This paper describes how lattice theory can be utilised to organise situations, which reflects the internal structure of situations such as generalisation and dependence. We claim that situation lattices will prove beneficial in identifying situations, and maintaining the consistency and integrity of situations. They will also help in resolving the uncertainty issues inherent in context and situations by working with Bayesian Networks.

1 Introduction

Context-aware computing systems provide adaptive services or behaviours according to different contexts. Context can be sensed from physical sensors, profiled by users, or derived from application or meta-information existing in systems. This context, acquired without any further interpretation, is called *low-level* context. It may be meaningless, trivial, vulnerable to small changes, or uncertain. A system might not necessarily be expected to adapt its behaviour to each and every change of context. If the context is incorrectly reported, or is considered irrelevant to applications, a problem will occur when a system makes a responsive action in reaction to real-time contextual changes [11].

It is difficult to build behaviours that adapt directly to low-level context. It is more attractive that a context-aware system is aware of *situations*, which are external semantic interpretations of context [4]. Compared to context, situations are meaningful, relatively stable, and certain. By abstracting contexts into situations, it is easier to resolve from imperfect context, capture meaningful contextual changes, and make it transparent to add or remove context sources. A meeting detection application (such as when Sensay [12] attempts to detect whether a meeting is taking place or not) should not be overly concerned with

* This work is partially supported by Science Foundation Ireland under grant numbers 05/RFP/CMS0062 "Towards a semantics of pervasive computing" and 04/RPI/1544 "Secure and predictable pervasive computing".

individual pieces of context, such as noise levels; rather it should concern itself with what the actual situation is – in this case, whether or not a meeting is taking place. A meeting situation can be composed with specific contexts: whether there are more than two people in a designated place; whether the current time is during office hours; whether the ambient noise levels are high? When a new type of context is introduced that can influence the situation (e.g., a meeting is scheduled in the calendar), then the situation specification is modified. However, its associated actions (e.g., change the mode of the attendances' mobile phones) will not be affected.

Situations, as an integral unit of semantics, are considered crucial in determining a system's actions. It is beneficial to define system behaviours only on situations, and make any context or contextual change transparent. Therefore, a promising context-aware computing system tends to be situation-aware.

As the study of situations has become popular, a huge number of situations are produced in an *ad hoc* way (for example, those outlined in Section 2). In order to benefit from using situations, it is necessary to analyse the internal relationships between situations. A situation can be decomposed into a set of smaller situations, which is a typical *composition or dependence* relation between situations. One situation can be considered more general than another situation, which is a *generalisation* relation: for example, a meeting situation is considered more general than a conference meeting situation, because the conditions inherent in the conference meeting situation subsume or imply the conditions in a meeting situation. Alternatively, a situation may be required to precede another situation, i.e., there is a *temporal order* between the situations.

Changes in situations may cause the system to adaptively change its behaviour, and in turn, this change in behaviour could lead to the generation of new contexts leading to new situations. Dealing with the rich internal relations between situations requires an efficient approach to organise situations, detect inconsistent situation specifications, and study the dynamic evolution of situations. These challenges are also proposed as future work by Loke [8].

This paper does not aim to provide a novel representation for situations: we use the typical representation – logical predicates. We focus on how to study the characteristics of situations by applying lattice theory. Situation lattices will be used to analyse the relations between situations. They will help to maintain the consistency and integrity when defining situations. This can avoid checking and modification of situation specifications when the errors are detected at runtime. We also study the issue of uncertainty based on situation lattices.

The remainder of this paper is organised as follows: Section 2 introduces the current state of research in studying situations; Section 3 details the design of situation lattices, and provides analysis of their characteristics. Section 4 discusses two approaches for dealing with uncertainties in situation identification. Finally, Section 5 draws a conclusion to the paper and outlines the future direction of this research.

2 Related Work

Past research on context-aware systems placed emphasis on modeling low-level context. More recently, the interest is on how to abstract, represent, and identify situations from the raw context. Early attempts such as Gu *et al.*'s ontology-based model [13] used first-order logical predicates to define situations. These attempts simply composed context and situation with logical operators.

Yau *et al.* analysed the semantics of situations and gave them formal representations [16]. Context is considered as any instantaneous, detectable, and relevant property of the environment, the system, or users. A situation is a set of contexts over a period of time that is relevant to future device actions. A situation can be atomic or composite. An atomic situation is composed of contexts in terms of context operators, including function, arithmetic or comparison operators, and time constraints. The time constraints involve `forAny`, `exists`, `time-stamp`, `offset`, and `interval`. A composite situation is composed of atomic or other composite situations in terms of logical operators and time constraints. This helps application designers to specify situations using formal expressions.

Costa *et al.* [3] studied the classification of situations in terms of their composition. A situation can be an *intrinsic context situation* – that is immediately derived from a single piece of context; it can be a *relational context situation* – that is used to associate multiple pieces of context in a certain relation; it can be a *formal relation situation* – that is defined by applying formal relations between two pieces of context directly, such as greater than, subset of, and distance; or it can be *combined situation* – that is made up from situations.

Loke [8] proposed a novel way of representing situations by decoupling the inference procedures of reasoning about context and situations from the acquisition procedure of sensor readings from context-aware systems. They apply a logic programming approach to characterising situations, which helps the system designer in naturally individuating and identifying situations for an application. It also provides a high level of programming and reasoning situation for the developers.

Thomson *et al.* provided a reusable library of situation specifications that helps to automatically determine situations [14]. They expressed different levels of granularity of a situation through specification inheritance. New specifications are created as variations of existing ones so that the same situation can be interpreted at different levels of abstraction. We apply a similar approach to expressing situations through inheritance, however, the situation lattice we propose is a higher level structure that can be used to organise the specifications and further exploit richer characteristics in situations.

Most of the current work studies the composition of situations and formal representations. However, none of them have proposed a formal mechanism to organise the situations.

3 Situation Lattice

This section examines the application of lattice theory [1] to the organisation of situations so as to study the characteristics of situations, such as generalisation, and dependence.

3.1 Construction of Situation Lattices

Situation lattices are inspired by Woods' use of lattice theory to recognise situations in linguistics [15]. A complete lattice is a partially ordered set where each subset of elements have the least upper bound and the greatest lower bound. The lattice theory is useful studying the structures with partial order. This paper will apply this formal theory to study situations.

Definition 1. *A **situation lattice**, L, is defined as $L = (S, \leq)$, where S is a set of situations and the partial order $\leq$ is a generalisation relation between situations. Each situation is associated with a logical description $l(t_1, \ldots, t_m)$, where t_i is a context predicate, or a basic or composite situation. $s_i \leq s_j$, $s_i, s_j \in S$ defines s_j to be a more general situation than s_i, iff any logical description being satisfied by s_j will be also satisfied by s_i. If two situations have no generalisation relation between them, then they are called **disjoint** situations.*

In S, a unique top situation $s_\top$ is a universally true situation. $s_\top$ is the most general situation so that $\forall s_i \in S$, $s_i \leq s_\top$. Dually, a unique bottom situation $s_\bot$ is a universally false situation. $s_\bot$ is the most specific situation so that $s_\bot \leq s_i$. Each situation in S is a basic or composite situation (except $s_\top$ and $s_\bot$). All basic situations are immediately under the top situation $s_\top$, which are derived from pieces of context through a general mapping function $f(c^*) = s$. The function takes a single piece c or a composition $c^* = c_1 \times c_2 \times \ldots \times c_n$ of context to derive a basic situation. A composite situation is made up of basic or other composite situations and sits under the basic situations in S.

Given $s_i, s_j \in S$, a situation s_j is more general than s_i iff the logical description l_j that is satisfied by s_j will also be satisfied by s_i. That is, $s_i \leq s_j$ implies that a logical description l_i subsumes l_j, labelled as $l_j \sqsubseteq l_i$. The relations $\leq$ and $\sqsubseteq$ have the same meaning w.r.t. the *partial order*. Thus, l_i can be rewritten by substituting l_j for the corresponding part in l_i: $l_i = l_j \wedge l_i^* \wedge l_i'$, where l_i^* is a logical description particular to l_i. l_i' is the residual part of the logical description, for besides s_j there may be other immediately more general situations above s_i.

To simplify the logical descriptions of situations, the specific situations automatically inherit the logical descriptions from its immediately general situations. This approach is used to build and maintain a situation lattice, whose formal proposition is expressed as follows.

Proposition 1. *At the appropriate level of generality of L, a situation s is specified with a logical description l^* that is only particular to itself. Its complete logical description can be obtained from $l = (\bigwedge_{i=1}^{m} l_i) \wedge l^*$, where l_i is the complete logical description of s_i that is the immediately more general situation than s:*

$s \preceq s_i$. In turn, $s_n \leq \ldots \leq s_1$ holds if all of the complete logical descriptions satisfy the following condition: $l_1 \sqsubseteq \ldots \sqsubseteq l_n$.

Figure 1 shows an example of a situation lattice for the meeting scenario discussed in Section 1. The basic situation $s_{highNoise}$ is identified by evaluating the noise degree sensed from the noise sensor and its logical description is `noise(conference_room, greaterThan, 3)`. The context from the positioning sensor is evaluated to identify the basic situation s_{np2}: the number of current people present is over two. The situation s_{mCal} will be true if there is a meeting scheduled for now in a sensed calendar. The situations s_{projOn} and s_{speOn} will be true if the projector and the speaker are turned on respectively.

A composite situation $s_{meeting}$ is used to evaluate whether a meeting is going on, whose logical description is expressed as $l_{mCal} \wedge l_{np2}$. The meeting situation $s_{meeting}$ has two more specific situations: a group meeting s_{gm} and a conference meeting s_{cm}. Each of these situations has inherited the logical description from $s_{meeting}$ and extended it with their particular descriptions. The logical description of a group meeting situation $l_{gm} = l_{meeting} \wedge l_{highNoise} \wedge l_{gp2}$, must satisfy that from another two situations: the noise level is above the third level, and there are at least two group members. The logical description of a conference meeting $l_{cm} = l_{meeting} \wedge l_{projOn} \wedge l_{speOn} \wedge l_{np10}$, must satisfy that from another three situations: the projector and the speaker are in use, and there are more than ten people present. Particularly, the situations s_{gp2} and s_{np10} are more specific situations relative to s_{np2}. s_{gp2} inherits the number requirement on involved people and extends them to the group identities of the people. s_{np10} simply constrains the number requirement on people present – at least ten people. In this lattice, s_1 is the unique top situation, and s_0 is the unique bottom situation.

For any two situations in a situation lattice, the join situation is the most specific situation among their more general situations, whose logical description should contain the common part of their logical descriptions. The meet situation is the most general situations among their more specific situations, whose logical description should contain the conjunction of their logical descriptions.

In whole, the essential characteristic of this situation lattice is the ability to represent situations of various degrees of generality. It explicitly represents the inheritance relationships between corresponding constituents of those situations.

3.2 Analysis of Situation Lattices

Exploring Dependence Relationships Between Situations A dependence relation between situations is discussed in most context modeling research (such as the research of Gu *et al.* [13] and Henricksen *et al.* [7]. We use situation lattices to capture this relationship.

The situation lattice is regarded as a specialisation structure with respect to the generalisation if it is observed downwards from the top down. A situation $s \in S$ is more *general* relative to all its sub-situations. It also can be considered as a dependence structure if it is observed upwards from the bottom. A specific

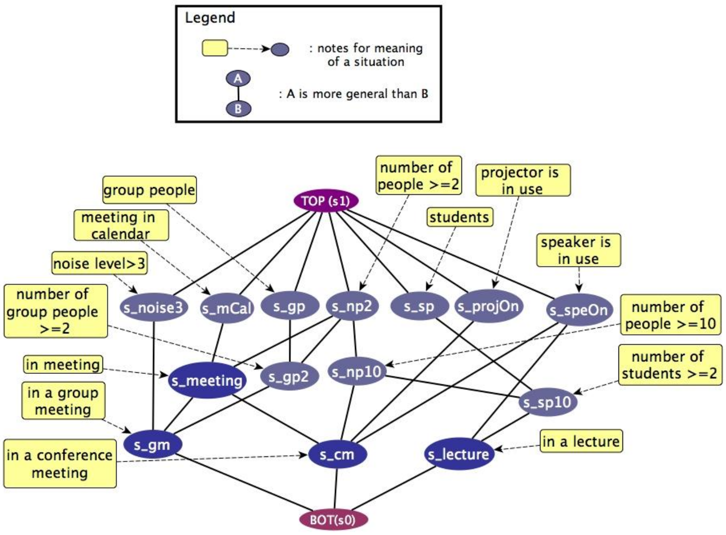


Fig. 1. Meeting situations in a situation lattice

situation can be decomposed into a few of more general situations. Its satisfiability *depends* on the evaluation of the satisfiablity of all its immediately more general situations.

In Figure 1, the satisfiability of a situation $s_{meeting}$ depends on that of its component situations: s_{mCal} and s_{np2}. The satisfiability of s_{mCal} and s_{gp2} depends on that of s_1. The top situation stores all the proper states of a system, and it holds if a system is running properly. Conversely, the bottom situation s_0 stores all the improper states of a system, and it holds if there is anything wrong with the system. Therefore, the bottom situation holds if inconsistent situations are detected. For example, in a given place at a given time, s_0 holds if two situations s_{gm} and s_{cm} are identified by the system.

Maintaining the Consistency and Integrity of Situations Context-aware computing systems typically involve a large quantity of context, based on which a huge number of situations can be created and specified. The question is: how can situations be kept consistent and integral? Consistency means that logical descriptions should be compatible between non-disjoint situations. For instance in Figure 1, it is not possible for a logical description of a group meeting situation s_{gm} to conflict with that of a general meeting situation $s_{meeting}$. Integrity means that logical descriptions should not be satisfied by any two disjoint situations. For example, the logical description that is satisfied by a group meeting s_{gm} in a room should not be satisfied by a conference meeting s_{cm}, or a lecture situation $s_{lecture}$ in that room at the same time.

Once the errors of inconsistency and non-integrity are detected at runtime, the system designers will be forced to rewrite situation specifications. This repetitive checking and modification takes a lot of time, therefore, it would be advantageous if these problems could be spotted and avoided when defining situations. From the top situation, each of its immediately more specific situations should not only satisfy logical descriptions of the top situation, but also contain logical descriptions exclusive from that of other siblings. This checking will be conducted recursively through the whole process of construction. According to Proposition 1, a new situation s is specified in a logical description: $l = \bigwedge_{i=1}^{m} l_i \wedge l^*$, where l_i is the complete logical description of one of its immediately more general situation s_i. If l is evaluated to be false, then there is a part of l^* conflicting with l_i, which implies that s breaks the consistency requirement. The integrity will be checked by comparing l with any logical description l_j of its sibling situations. s is considered as an acceptable situation if its logical expression is different with that of its siblings: $l \neq l_j$.

Identifying Situations There are two ways of recognising a situation. **Backward chaining** starts with a list of situations and works backward to see whether the available context supports the requirements of any of the situations. Backward chaining is a typical mechanism used in current context-aware computing. To identify a meeting situation, a system will collect all the perceptible context, for example, noise level and the number of people in this room. If the context satisfies the conditions of a meeting situation, then it is identified. This backward chaining is useful only when a situation to be determined is chosen beforehand.

In the situation lattice, the logical description is defined particularly for each situation, and increasingly inherited from its general situations. Backward chaining is carried out by evaluating this incrementally logical description with the given context.

While in many real applications, where there are many possible situations, it is not always practical to locate a situation beforehand. In this case, **forward chaining** should be used: this starts with the acquired context and applies inference rules to arrive at a situation. In this circumstance, faced with a large number of inference rules, it is infeasible to find the rules that match a certain situation by systematically checking each rule. It is necessary to find out a way of reducing the computational load and locating a situation efficiently. In the situation lattice, situations can be shared which avoids repetitive evaluation of situations. The forward chaining does not have the problem of infinite loops in the situation lattices either.

The situation lattice will be suitable for the forward chaining. A system starts by identifying basic situations from the given set of context. Only the logical description l^* particular to a situation will be checked, rather than its complete logical description. If the description is satisfied, the satisfied context will be removed from the original given context set and the chaining will continue checking its more specific situations. In this way, only the minimum descriptions will be evaluated every time without repetition, and the given context set is reduced continually. This will reduce the computation load and improve the efficiency. When a set of most specific situations $\{s_i, \ldots, s_k\}$ are located, the target situation is the join of all these situations.

4 Situation Lattices and Uncertainties

When dealing with real-world context data, there is no guarantee that situations will be identified with complete certainty. The uncertainty of context is subject to sensor failure, noise, delays, disconnected sensor network, infrequent update in response to changes [7]. Context is considered uncertain, if it is

- *incomplete*, when some information is unknown or missing. There may not be enough evidence to determine a the correct situation;
- *imprecise*, when the resolution of the context cannot satisfy the requirement of applications;
- *conflicting*, when there are several inconsistent pieces of information from different sources, which may result in multiple disjoint situations being determined;
- *incorrect or meaningless*, when the information is erroneous compared to the actual state or reality, which may result in an incorrect situation being determined;
- and *out-of-date* when the information is stale and is not updated in response to changes, which may result in an incorrect situation being determined.

Many of these uncertainties are amplified when using inference rules to reason about context, as well as the typical insensitivity of rules to noisy inputs. Another concern is the difficulty in defining and maintaining accurate inference rules. These uncertainties can result in incomplete, inconsistent, and incorrect situations being identified, especially when dealing with real-world context.

4.1 Coarse-Grained Approach to Resolving Uncertainty

A coarse-grained approach is introduced to resolve uncertainties with respect to the characteristics of a lattice. Compared to a specific situation, a general situation has fewer or looser requirements (or conditions). The general situation can be extended to more specific situations by adding requirements in its logical description (e.g., from s_{np2} to s_{gp2}), tightening the constraints (e.g., from s_{np2} to s_{np10}), or uniting with other situations (e.g., from s_{sp} and s_{np10} to s_{sp10}). If some context is too incomplete to support a given situation then this situation cannot be identified. However, its general situations will be checked until the context is satisfied by a situation. Therefore, when a system fails recognising a specific situation, it can loosen the requirement to locate a more general one. If the context is conflicting to each other, it generates some disjoint situations. These situations satisfy different conditions that are difficult to determine which is proper, while the conditions satisfied by all of them are considered correct.

As a result, the join of these disjoint situations will be returned to resolve the inconsistent uncertainty.

The system is kept stable using the coarse-grained approach because it always tends to choose the inviolable situation, even though this is not always the most appropriate situation. In the pathological case, when all of the derived disjoint situations are conflicting, the join of them is the most general situation $s_\top$. That implies the system does not detect any situation and will not take any particular behaviour, so it is considered insensitive to situations or context. However, among the disjoint situations, if uncertainties of context were incorporated into the lattice, it might be appropriate to select the situation with the highest degree of confidence. The system should then carry out the behaviours specified for that situation. This responsive system is more suitable for real-world applications. Consequently, we propose a fine-grained approach to quantify the confidence of generated situations, which helps to determine the situation that is most likely to occur.

4.2 Fine-grained Approach to Resolving Uncertainty

The typical fine-grained approach attempts to quantify the uncertainties underlying situations using probabilities. These probabilities attempt to capture the uncertainty caused by imperfect context and error-prone deriving mechanism. The situation lattice represents the dependence relationship between situations, so a promising approach is to represent the probabilities on both situations and dependence relationships and then reason on them with these probabilities. Bayesian Networks have a causal semantics that encode the strength of causal relationships with probabilities [6].

Bayesian networks are usually used to calculate the probabilities for decision making under uncertainty. A Bayesian network is a directed acyclic graph in which each node represents a variable that can be discrete or continuous, and each arc is the causal relationship between nodes. If there is an arc from a node A to another node B, then A is called a *parent* of B, implying that the variable B is regarded depending directly on A. If a node does not have a parent, then it is called *root*. Each root node is associated with an *a priori* probability. Each non-root node is associated with a conditional probability distribution (CPD). If the variables are discrete, then the CPD is represented with a conditional probability table (CPT) given all possible combination of their parent nodes: $p(x \mid parent(x))$, where $parent(x)$ is a parent set of a node x.

It is obvious that a situation lattice has a very similar structure to a Bayesian network. The lattice can be converted to a Bayesian network in a straightforward manner: each node in a Bayesian network corresponds to a situation, and each arc to a dependence edge. In this Bayesian network, the root nodes are considered the basic situations that are immediately under the top situation $s_\top$. After building the graphical model of Bayesian network, we will assess the prior probability for each root node and the conditional probability for each non-root node. The Bayesian probability of an event is a degree of belief in this event [6] and it can be obtained from the domain expert or observations.

Considering the uncertainty and dynamism, the probabilities will be evaluated by training a set of real data. For the probability of a root node, a simple but straightforward approach is $p(\omega) = \frac{N'}{N}$, where N' is the times that a certain state ω takes place and is recognised, and N is the total number of observations. To simplify the computation, it is assumed that the structure of the model is known and the full observations are possible, so the *maximum likelihood estimate* [9] is applied for the conditional probability distribution. For each non-root node s, one of its discrete state is written as ω, its parent nodes are $s_1, \ldots, s_n$, and one of its conditional probability is calculated as follows:

$$p(s = \omega \mid s_1 = \omega_1, \ldots, s_n = \omega_n) = \frac{N(s = \omega, s_1 = \omega_1, \ldots, s_n = \omega_n)}{N(s_1 = \omega_1, \ldots, s_n = \omega_n)}, \qquad (1)$$

where $N(s = \omega, s_1 = \omega_1, \ldots, s_n = \omega_n)$ is the number that s is recognised in one of its states ω, and all of its parents are in one of its own states ω_i; and $N(s_1 = \omega_1, \ldots, s_n = \omega_n)$ is the number that all of its parents are in one of its own states ω_i. In a meeting scenario example in Figure 1, the prior probability of a root situation s_{np2} is 0.74. For the group meeting s_{gm}, one of its conditional probabilities $p(s_{gm} = true \mid s_{highNoise} = low, s_{meeting} = true, s_{gp2} = true)$ is 0.87. When the noise level is sensed lower than the third level, the meeting situation takes place, and more than two group members are located, the probability of a group meeting is 0.87.

Bayesian inference is the process of updating the probabilities based on the relationships in the model and the recent evidence. The new observation is applied to the model by assigning a variable to a state that is recognised from the observation. Then the probabilities of all the other variables that are connected to this variable will be updated. The new probability is called *posterior* probability that reflects the new levels of belief.

Under the conditional independence assumption, the joint probability distribution is applied to compute the probability of the resultant situations given the causal situations: $p(s_i = \omega) = \prod_{k=1}^{n} p(s_k \mid parent(s_k))$. For example, if the situations s_{np2} is recognised *true* and s_{mCal} is recognised *true*, and other situations are uncertain, then the probability of a meeting situation $s_{meeting}$ is 0.98, and that of a group meeting situation s_{gm} is 0.64.

With Bayesian networks, a system will not only return a more general situation through the above coarse-grained approach, but it will also return a specific situation with the highest possibility. If the highest possibility is beyond the threshold that is specified by a system, the behaviours corresponding to that situation will be carried out.

Up to now, we assume that the Bayesian network is operating on a closed world, with a structure that is known *a priori*. However, in a real-world context-aware system this structure may not be known or will change over time as sources of context are added and removed by the environment. The assumption that inputs are certain is also unrealistic given the inherent uncertainty of context data [7]. The promise of using Bayesian networks with situation lattices is that they could be used to learn the underlying structure of situations, which would make it possible to reconfigure the situation lattice.

5 Conclusion and Future Work

As situations become more and more important, system designers tend to specify a large number of rules to identify various situations in an *ad hoc* way. An efficient approach is expected to organise and manage these situations so that their specifications maintain the consistency and integrity requirements. This paper applies a formal structure using lattice theory to organise situations.

The situation lattice reflects the generalisation relation of situations and captures the dependence between situations. We believe it will be helpful when maintaining the consistency and integrity of situations, however, the involved computation may be huge when faced with lots of situations. This paper only presents a simple situation lattice with a limited number of situations, while we will attempt to design an algorithm to make the checking procedure scalable and efficient. The situation lattice will also be beneficial when identifying the situations using backward and forward chaining approaches. However, the situation lattice only reflects the static structures of situations. We have discussed the dynamic evolution of situations with a fibration theory in earlier work [4]. In the future, we will investigate how situation lattices and fibrations can be made to work together. *Chu Space* [17] is anther interesting worth studying, which may be useful for exploring the temporal order between situations, and for preserving situations' structures during the dynamic evolution.

In dealing with the issue of uncertainty, the situation lattice supports a coarse-grained approach and a fine-grained approach by working with Bayesian networks. Bayesian networks work well if situations are limited to a small number, and if the context sources are relatively fixed. (Gu *et al.* [5] and Ranganathan *et al.* [10] have successfully applied Bayesian networks to deal with uncertainties in contextual reasoning.) However, this assumption is contradictory to the nature of context-aware computing systems. A system may contain thousands of situations so as to satisfy various kinds of customised applications. If a non-root situation node has m parent situations, then the size of its conditional probability table is $2^m \times (m+1)$ (if each variable has only two discrete states). Considering the complexity of situations, the computation of conditional probability tables will be large.

For the acquisition of context, context-aware systems should watch all the potential context in the environment. This is a big issue when applying these systems in reality, and potentially not solvable at the current research stage. In these environments, new context sources often enter and leave. This frequent churn in context sources will quickly render the original Bayesian network useless and require the system to frequently retrain itself. If there are a large number of nodes in the Bayesian network, the cost of training will be prohibitive. What's more, if learning the structure of nodes is required, the NP-hard problem underlying the Bayesian network will become an obstacle [2].

Considering the above disadvantages, we will design the algorithms to optimise the performance of Bayesian networks based on the particular characteristics of context-aware computing systems.

References

1. G. Birkhoff. *Lattice Theory.* Providence, R.I. : American Mathematical Society, 3rd ed edition, 1967.
2. E. Charniak. Bayesian networks without tears: making bayesian networks more accessible to the probabilistically unsophisticated. *AI Mag.*, 12(4):50–63, 1991.
3. P. D. Costa, G. Guizzardi, J. P. A. Almeida, L. F. Pires, and M. J. van Sinderen. Situations in conceptual modeling of context. In *EDOC 2006 workshop proceedings*, pages 6–16, October 2006.
4. S. Dobson and J. Ye. Using fibrations for situation identification. In T. Strang, V. Cahill, and A. Quigley, editors, *Pervasive 2006 workshop proceedings*, pages 645–651. Springer Verlag, 2006.
5. T. Gu, H. K. Pung, and D. Q. Zhang. A Bayesian approach for dealing with uncertain contexts. In *Procs of Pervasive 2004*, 2004.
6. D. Heckerman. A tutorial on learning with bayesian networks. Technical Report MSR-TR-95-06, Microsoft Research, Redmond, Washington, June 1996.
7. K. Henricksen and J. Indulska. Modelling and using imperfect context information. In *Procs of PERCOM'04*, page 33, 2004.
8. S. W. Loke. Representing and reasoning with situations for context-aware pervasive computing: a logic programming perspective. *Knowl. Eng. Rev.*, 19(3):213–233, 2004.
9. K. Murphy. A brief introduction to graphical models and bayesian networks. http://www.cs.ubc.ca/ murphyk/Bayes/bayes.html, 1998.
10. A. Ranganathan, J. Al-Muhtadi, and R. H. Campbell. Reasoning about uncertain contexts in pervasive computing environments. *IEEE Pervasive Computing*, 03(2):62–70, 2004.
11. B. Schilit, N. Adams, and R. Want. Context-aware computing applications. In *IEEE Workshop on Mobile Computing Systems and Applications*, 1994.
12. D. Siewiorek, A. Smailagic, J. Furukawa, A. Krause, N. Moraveji, K. Reiger, J. Shaffer, and F. L. Wong. Sensay: A context-aware mobile phone. In *Procs of the 7th IEEE ISWC*, page 248, 2003.
13. Tao Gu and Xiao Hang Wang and Hung Keng Pung and Da Qing Zhang. An Ontology-based Context Model in Intelligent Environments. In *Procs of CNDS 2004*, pages 270–275, January 2004.
14. G. Thomson, S. Terzis, and P. Nixon. Situation determination with reusable situation specifications. In *PerCom 2006 Workshop Procs*, pages 620–623, 2006.
15. W. A. Woods. Taxonomic lattice structures for situation recognition. In *Proceedings of the 1978 workshop on Theoretical issues in natural language processing*, pages 33–41, 1978.
16. S. S. Yau, D. Huang, H. Gong, and Y. Yao. Support for situation awareness in trustworthy ubiquitous computing application software: Papers from compsac 2004. *Softw. Pract. Exper.*, 36(9):893–921, 2006.
17. G.-Q. Zhang. Chu spaces, concept lattices, and domains. *Electronic Notes in Theoretical Computer Science*, 83, 2004.

Towards Scatterbox: a Context-Aware Message Forwarding Platform*

Stephen Knox, Adrian K. Clear, Ross Shannon, Lorcan Coyle,
Simon Dobson, Aaron J. Quigley, and Paddy Nixon

Systems Research Group, School of Computer Science and Informatics
UCD Dublin IE
stephen.knox@ucd.ie

Abstract. Context-aware systems that rely on mobile devices for user interaction must address the low bandwidth of both communications and more importantly the user's limited attention, which will typically be split between several competing tasks. Content delivery in such systems must be adapted closely to users' evolving situations and shifting priorities, in a way that cannot be accomplished using static filtering determined *a priori*. We propose a more dynamic context-driven approach to content delivery, that integrates information from a wide range of sources. We demonstrate our approach on a system for adaptive message prioritisation and forwarding.

1 Introduction

Rich sources of context data are an integral part of building intelligent pervasive computing applications. Central to pervasive computing is the notion that "technology recedes into the background of our lives" [1]. The classical view of human-computer interaction needs to be extended to include both a cloud of interoperating heterogeneous electronic devices and the ability to interact with one or many when and wherever desired. This extended view requires that disparate devices are able to interoperate easily. This supports enhanced interactions with the user which break away from traditional distribution channels, reaching the user through whichever device they currently have available. As small mobile devices with built-in wireless capabilities such as mobile phones and PDAs become more widespread, they present an ideal opportunity to afford ubiquitous communication services to the user.

Advances in technology in recent years have meant that computing devices have become cheaper, smaller, and more powerful. Mobile devices including mobile phones and PDAs are constantly increasing in utility through the addition of extra sensory equipment like gyroscopes, accelerometers and GPS receivers. In addition, communication devices are beginning to be embedded into everyday objects such as toys, refrigerators and coffee cups [2, 3]. Each of these advances expands both the opportunities

* This work is partially supported by Science Foundation Ireland under grant number 05/RFP/CMS0062, "Towards a Semantics of Pervasive Computing.", grant number 04/RPI/1544, "Secure and Predictable Pervasive Computing.", grant number 03/CE2/I303-1, "LERO: the Irish Software Engineering Research Centre.", and by an EMBARK Scholarship from the Irish Research Council in Science, Engineering and Technology.

for the delivery of ubiquitous communication services but also increases the system complexity in coordinating this cloud of devices.

With the tremendous increase in available contextual data, a context-aware system can determine many facts from the environment that can inform its behaviour e.g., who is present, or what task they are performing. An example of this is a smart meeting room that manages shared projectors [4], or a smart room for elderly people which detects incidents such as falls and reacts to them quickly [5]. These decisions can only be made if there are enough data, of sufficient fidelity, to support them. Therefore there must be many sensory devices, each providing differing inputs, that each contribute to the knowledge of the system as a whole. These devices will often go unnoticed by the user, but they provide valuable information about the user's surroundings and the context of their activities.

We define "context" as any aspect of the environment of a system understood symbolically – or, more concretely, as a measurable component of a given situation. By "situation" we mean a certain composition of various simple and derived contexts that gives rise to pervasive services. These contexts may be simple metrics which can be investigated with instruments, such as a time context (e.g. 18:48 GMT), a location context (e.g. Coffee Area) or a user context (e.g. Bruce), but also range to more complex computations such as a user's current social context, meaning the other users that they are sharing a space with.

A central challenge for pervasive computing is to integrate contextual information in order to best recognise and service the changing situation. Since individual sources of context are inherently error-prone, this cannot be accomplished by focusing on any one source but instead requires a fusion-based approach that can integrate all available information, giving due weight to the fidelity of each source.

This paper describes an approach to situation awareness being prototyped in a system we call Scatterbox, a "moving letterbox" that delivers relevant messages to a user's mobile device based on the context derived from sensors within a pervasive computing environment. The goal of this system is to take multiple, heterogeneous sources of contextual data, and extrapolate the situations that they map to. The characteristics of these situations define whether an appropriate action should be taken — and so whether a certain message should be delivered to a user's phone or PDA, limiting distraction by only sending messages that are timely and relevant. In accordance with the study done by Oulasdvirta *et al.* [6] on the "drastically short term" limited attention span of mobile users, Scatterbox provides short, concise messages requiring minimal attention.

The system we propose monitors a user's e-mail inbox and dynamically forwards messages to him, depending on his situation. This means that he will be notified of the receipt of only important e-mails or messages relevant to his current task when he is in a certain situation, while all other messages will be stored as normal in his e-mail inbox. This provides a tangible demonstrator of a real-time pervasive system which is constantly adapting to changes in the user's situation.

The remainder of the paper is organised as follows. Section 2 describes related research in the area of contextual message forwarding. Section 3 contains a formal description of our approach to the composition of context into situational awareness. Section 4 describes our context acquisition and reasoning methods using a contextual framework, while Section 5 describes how our Scatterbox system has been implemented to work in a physical location. Section 6 outlines our proposed evaluation that will be performed to test the efficacy of Scatterbox using real messages, real users and real context. In the final section, we offer some conclusions and plans for future work.

2 Related Work

Standard message delivery systems such as email and SMS do not take the user's context into account, and can make it difficult for a user to prioritise inputs, as well as lead to irritation due to unnecessary disruption. Context-awareness can be used as an effective augmentation of existing message delivery systems. Filtering messages using context can be used as a means to decrease the disruption that message delivery can cause to a user. This approach to message delivery has the potential to become the cutting-edge in messaging, self-organisation, and content filtering.

Multiple approaches to context-aware message delivery have previously been explored. The Stick-e note architecture [7] is one of the earliest of these. Stick-e notes can be messages or other objects which have contexts attached to them. This architecture incorporates the user's physical environment into service delivery; the principle contexts used are location and time. The notes may be stored on a user's PDA or a static device like a desktop PC. The message or object gets delivered to a user only when they enter the context that is attached to the object. For example, users may be reminded that they have to return a book to somebody when they are at their bookshelf and in the presence of the person from whom they borrowed the book.

Nakanishi *et al.* [8] proposed the Context Aware Messaging Service that uses schedule information, location information and available media to send an incoming message (or call) to users using an appropriate protocol and device. This system uses context to determine whether the message should be sent using e-mail, SMS, and so on, and what device it should be sent to. The authors performed an evaluation of a two month experiment in Tokyo. The results suggest that the use of context to determine what device a message should be sent to made the retrieval of messages more convenient. It also made users feel comfortable to know that another user would not be disturbed by a message if the timing was inappropriate.

Context-aware adaptation has also been introduced to applications with the goal of reducing a user's distractions. Miller *et al.* [9] leverage the Aura Contextual Information Service in order to build distraction-free context-aware applications. In their example, urgent messages should be sent using SMS or IM depending on the situation.

Ho *et al.* [10] also propose the use of context in order to prevent the user from being interrupted by messages, calls, etc. at inappropriate times. They list 11 factors which influence a person's interruptibility at a given moment and complete a user trial which shows interesting results. People are less likely to be disrupted when they are between activities (e.g., between a meeting and going to lunch).

3 Foundations of Context-Awareness and Adaptive Behaviour

The notion of context is key to context-aware computing, and a universally accepted definition has been difficult to realise. Yau [11] defines context as "any instantaneous, detectable and relevant property of the environment, the system or users", which is similar to, but more general than, Dey's [12] definition that context is "any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves". Henricksen [13] proposes to make a distinction between the concepts of *context* and *context modelling* in order to achieve consensus and precision:

- "The context of a task is the set of circumstances surrounding it that are potentially of relevance to its completion."
- "A context model identifies a concrete subset of the context that is realistically attainable from sensors, applications and users and able to be exploited in the execution of the task. The context model that is employed by a given context-aware application is usually explicitly specified by the application developer, but may evolve over time."

We aim to take a step further and formalise the notion of context based on the above definitions. The motivation for such a formalism is the need to derive an appropriate means to model context. From Yau's mention of the properties of the environment; Dey's mention of information about an entity; and Henricksen's mention of the more general term "circumstances", we propose to capture the structure of context in a tuple containing a subject, predicate and object. We can then view a predicate as a means to relate properties, information, or circumstances to an entity. Thus, we can express any detectable or realistically attainable facts relevant to the completion of a task.

Definition 1. *A context is a tuple containing a subject, predicate and object (s,p,o) that states a fact about the subject, where*

1. *The subject is an entity in the environment.*
2. *The object is a value or another entity.*
3. *The predicate is a relationship between the subject and object that defines the domain of the object.*

A context may be either a constant or a variable. A constant context must take on a single value from a domain. A variable context may take on values from a domain. The value of a variable context may change over time, while the domain stays the same.

An environment of a context-aware system can be viewed as a finite number of constant and variable contexts. The values that a variable context may attain come from a well understood domain of variable types including nominal or categorical e.g., times of year {Summer, August,...}, ordinal, quantitative, interval e.g., temperature range {-32, +32}, and ratios. The domain of an object captures the "type" of information, property or circumstance that we may legally relate to the subject using a particular predicate.

Context is very fine-grained for use in developing complex adaptive applications, as it is typically a low-level interpretation of raw sensor data. When defining adaptive behaviour, we consider a holistic view, using situations, rather than the combination of individual bits of context data. A situation is a natural abstraction of context data, providing more realistic points to associate adaptive behaviour with. For example, imagine that a user, Bruce, in a lecture theatre, has a slide set open and there are 35 other people in the room. Bruce is at the front of the room and is talking. This presents a lot of contextual data for us to comprehend in parts, but a context-aware system should be able to condense it and infer that the current situation is a "presentation". Often individual component contexts may change but the situation will be maintained. Other situations will be labeled "meeting", "lunch", etc. This view of contexts and situations proves to be quite versatile.

In order to define situation, we firstly define *situation space*. Given that each predicate restricts an object to a particular domain, we can define the set of variable contexts A with common subject s_a and predicate p_a as $A : \langle s_a, p_a, o_i \rangle$ for all $o_i \in D_i$, where D_i is the domain of o_i. Given two or more sets of contexts, we can define their situation space as the Cartesian product of the sets as follows.

Definition 2. *Given the sets of contexts* $A : \langle s_a, p_a, o_1 \rangle$, $B : \langle s_b, p_b, o_2 \rangle \ldots Z : \langle s_z, p_z, o_n \rangle$ *for all* $o_i \in D_i$*, where* D_i *is the domain of* o_i*, we can define the* situation space $AB \ldots Z$ *as the set* $\{(a_1, b_1 \ldots, z_1), (a_1, b_1 \ldots, z_2) \ldots (a_m, b_n \ldots, z_p)\}$ *where* $a_1 \ldots a_m \in A$*,* $b_1 \ldots b_n \in B \ldots$*, and* $z_1 \ldots z_p \in Z$*.*

A situation space thus captures all possible combinations of two or more contexts. We can thus define a concrete situation as a subset of a situation space.

Definition 3. *A situation is a subset of a situation space.*

Given a situation space AB defined over the variable context sets A and B, we can define the situation AB_1 as $\{(a_1, b_1), (a_3, b_2)\}$. Situations themselves may be used in the construction of more complex situations. For completeness, a base behaviour should be provided in order to cover all possible circumstances. For a more concrete example, imagine we have the context sets over the domains $Months$ and $Weather$. We can define the situation space for $Weather \times Month$ and define situations like $SummerSun$ or $SpringWind$.

Adaptive behaviour is achieved by creating points in the system called *adaptation points*. When one of these is reached, the system exhibits a corresponding behaviour. An adaptation point could be as simple as a context variable taking on a certain value or as complex as an elaborate situation composition. In general, we are selecting states of the system where something useful should happen. We may wish to set a user's phone profile to silent when she is in a meeting, for example, or set her Instant Message (IM) status to away when she is out on lunch.

4 Context Acquisition and Modelling

A context-aware system collects relevant context data from the environment and then acts appropriately based on this information. Our context-aware applications are built on top of Construct [14], a distributed fully decentralised open-source platform supporting the building of context-aware, adaptive, pervasive and autonomous systems[1]. Such systems interact through manipulation of a common data model rather than through the piecing together of services. Construct provides applications with a uniform view of context information regardless of how it was derived. Construct takes care of collating and distributing context data around a network. Applications can then query the nearest Construct node for data that their service relies on.

4.1 Sensors

Sensors are used to gather context data from the environment. We have categorised our sensors into two types: physical and virtual. Physical sensors directly detect characteristics of the environment e.g., sensors for location data obtained from Ubisense[2] and Bluetooth spotters. Virtual sensors obtain information from the Internet, local network or local computer, e.g., sensors for obtaining calendar information, monitoring computer activity and collating syndicated data feeds.

In this project we utilise the following physical and virtual sensors:

- **Ubisense Location Sensors** poll a Ubisense location tracking system which has been set up throughout our research lab and tracks special tags that users can carry around with them. The sensor calculates the x, y, and z coordinates of an individual with a peak granularity of 30cm in 3D space.
- **Bluetooth Location Sensors** poll for a user's designated Bluetooth-enabled device. A number of statically positioned "base stations" have been positioned throughout the lab. Due to Bluetooth's limited range, the set of static devices within the system which can connect to a mobile Bluetooth device allow us to infer a range of possible positions for the device.
- **Calendar Sensors** monitor a list of published iCal and vCal calendars for data about a user's appointments and location (e.g. room number) and availability for a period of time (e.g. during the next week).
- **Computer Activity Sensors** determine whether an individual is located at a computer by checking if they are logged-in and active at that terminal.

Data from each of these sensors is fed into a connected Construct node, from where it is disseminated to other nodes in the system. The Construct middleware is being complemented by the development of an uncertainty framework [15]. This performs aggregation of context and deals with inconsistencies that may arise when the same type of data is produced from different sensors (e.g., location sensors providing data that says a person is in two places at once).

4.2 Modelling Context, Situations and Behaviour

In order for the heterogeneous nodes running Construct to be able to interpret and communicate about context and situations, we need a means to model them.

[1] The Construct home page is located at http://construct-infrastructure.org/

[2] Learn more about Ubisense at http://www.ubisense.net/

We model context and situations as ontologies in the Web Ontology Language (OWL)[3]. Our reason for choosing OWL is that it fits our mathematical description of context and situations from Section 3 most appropriately. Entities and context categories are modelled as OWL classes. Their attributes correspond to context variables and constants from the previous section. For example, the Person class has constants such as name and e-mail and variables such as location. In order to create a person, an instance (or individual in OWL terminology) is made of this class and values are assigned to its attributes. Variable contexts may be changed frequently and timestamped by sensors.

We create situation spaces as OWL classes in a similar way. Their attributes are the context variables or constants which the space encapsulates. In order to create concrete situations, we create instances of the situation spaces by assigning values or intervals to the contexts. This is similar to composing context variables and creating a subset of the resulting situation space as described in the previous section.

Adaptation points can be viewed as augmented virtual sensors which consist of a context or situation and a corresponding behaviour. The virtual sensors asynchronously poll Construct to monitor whether their situations have been realised. They then execute their behaviours and possibly introduce more contextual information into the network. The prescribed behaviours may require context information to function e.g., if we wish to route a message to a user, we must first choose a device, using location information, to send it to. As mentioned in Section 3, for completeness adaptation points should be specified for all subsets of a situation space. One means to approach this requirement is to introduce a default behaviour on a subset of a situation space and specialise the behaviour for other, more significant subsets.

The expressiveness of our mathematical model for situations leads to some problems regarding conflict, however. By allowing situations to be composed of other situations, and adaptation points to be defined on elements as fine-grained as individual contexts, we must decide what behaviour the system should exhibit in occurrences where situation definitions overlap. Some possible approaches to this problem are given in Ye *et al.* [16].

5 Scatterbox

In order to demonstrate our approach to situation determination and use of context in pervasive systems, we are developing Scatterbox, a context-aware message forwarding platform. Scatterbox forwards certain incoming e-mails to users in a pervasive environment based on their context. The user's context is found by tracking his location and monitoring his daily schedule. This context data is accessed through Construct, and situations are identified based on this data. As messages arrive, Scatterbox forwards them to subscribed users should their situation warrant it.

We now describe in detail how Scatterbox is implemented. We illustrate the situations and behaviours that the application uses, describe our mode of message delivery, and show the use of the application with an example case study.

[3] The OWL specification is located at http://www.w3.org/TR/owl-features/

5.1 Situations and Behaviour

The situation spaces that we use for Scatterbox are *EmailRelevance*, *Meeting*, and *PresenceAtInbox*. *EmailRelevance* is dependent on factors such as the situation of the user, the sender of the e-mail, the e-mail content and the user's calendar information. In order for Scatterbox to know when a situation has occurred, it must poll Construct regularly for newly acquired context, and match the new context to the situation spaces.

For the Scatterbox application, we need only define one adaptation point, *relevantEmail*, which reacts by sending a summary message to the user's Bluetooth device. This behaviour can be defined as *sendMessage(user, msg)*. This command can be entered into Construct, distributed, and picked up by a node within range of the user's Bluetooth device. An application running on the Construct node then sends the message to the device.

5.2 Message Delivery

Bluetooth wireless technology is built into most devices that we carry around every day, like mobile phones, PDAs and other portable devices, which means Scatterbox is transparent, reliable, and scalable.

Transmission of messages is accomplished through Bluetooth's Push protocol. The Bluetooth capabilities of most mobile phones and similarly-powered devices are typically limited to a transmission range of approximately 10 metres. Each device can be uniquely identified within the system. If a user's Bluetooth device is in range of a Bluetooth-enabled Construct node, a message can be routed to the node and pushed to the mobile device. The message can then be accepted or rejected by the user. In Section 6 we show how we will use this acceptance or rejection to drive our evaluations.

5.3 Use case

Consider the case of Bruce, a user who has a meeting scheduled after lunch with one of his students. The meeting is to take place in his office, and Bruce is currently upstairs in the coffee area having lunch with one of his colleagues. The system's contextual inputs at this point are:

- The current time of day.
- Schedule information from Bruce's calendar application (which includes both the time it takes place and the room it takes place in).
- Bruce's current position in the building.

Many calendaring applications will offer the ability to e-mail the user a reminder of an appointment such as this meeting before the meeting is to take place. However, in this case an e-mail reminder will be of no use, as Bruce is not near his computer. Scatterbox routes messages that would normally arrive in a user's e-mail inbox to their mobile device, if they are away from their computer and the message is deemed to be contextually relevant by the system. In the case of calendar appointments, it is not a requirement that an e-mail with an appointment reminder is received; the user's calendar is sufficient, as contextual data can be extracted from it directly.

Shortly before a meeting is scheduled to occur, the system will query Bruce's location. When it notices he is not in the prescribed location, a message will be pushed to his phone to remind him of the meeting.

Variants of this situation that would not lead to a message being sent would be:

- had Bruce been in his office at the time the meeting was to start.
- had the student that Bruce was to meet also been up in the coffee area at the time the meeting was to start. In this case a message will not be sent, as Bruce's social context overrides the location information.

5.4 Situations in Scatterbox

To demonstrate how contextual information can be collected about Bruce, we will use the sensors mentioned in Section 4.1. Each sensor is modelled using an ontology, which describes how that sensor's data should be interpreted. This gives uniformity to the data within the system, so in the event of a query like: "What room is Bruce in?", it is a simple matter of making a single query based on the location ontology, rather than multiple queries for each separate type of sensor data.

Scatterbox continuously seeks context data using queries such as the following:

- Select last location of Bruce.
- Select upcoming entries from Bruce's calendar.
- Select sensor readings from Bruce's computer activity sensor.

Bruce's situation can be determined from the responses to these queries. For example, a Construct node could return:

- "Boardroom, 3.02pm"
- "Meeting with head of school, Boardroom, 3.00pm"
- "Inactive"

From these results, it is inferred that Bruce is in a meeting and not in his office. He therefore should only be sent a message if it is classified as being important, relative to his situation.

Every situation is defined by the following criteria: the values context data must hold for the situation to be realised, and the resultant behaviour. We allow the user to define their own message filters, which take the form of standard e-mail filters, such as filtering by sender, recipient, or keywords. For message classification, Scatterbox takes an approach based on common spam filtering techniques, such as white-lists, black-lists, and simple keyword classification. These filters are entered into Construct in the form of RDF. From that point on, they are viewed by Construct as being additional context data.

The data going into Construct is checked against a set of ontologies. This piece of data can then be associated with something in the environment. In the case of a Bluetooth reading, the datum is seen to be an attribute of a *Bluetooth device*. This Bluetooth device ontology is associated with the *Person* ontology with the $hasCellPhone$ relation. These relations allow Scatterbox to see whether a person is in the environment,

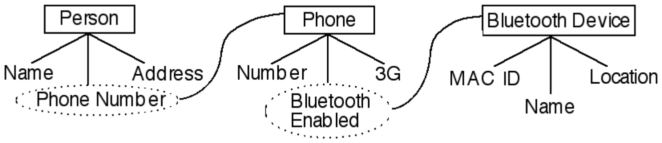


Fig. 1. An example of how ontologies are traversed.

where they are, and consequently determine their situation. This traversal of ontologies is illustrated in Figure 1.

Following on from the example above, Bruce must define how Scatterbox is to decide which messages are appropriate for each situation. This involves naming a situation and stating who he would accept messages from in that situation. An empty list implies no messages should be delivered. Secondly, he has the option of defining a set of keywords which have to appear in incoming messages for them to be deemed important enough for delivery. Finally, Bruce defines which criteria indicate a situation. He does this by creating instances of situation spaces in an ontology editor, for example, as described in Section 5.1.

From this point on, Scatterbox scans incoming e-mails and, using keyword classification, assigns e-mails to particular situations. It is assumed that the user will already have adequate spam protection preventing unwanted messages from reaching their e-mail inbox.

6 Evaluation

In order to evaluate Scatterbox's effectiveness in using context to determine which messages to forward, and the accuracy and usefulness of these messages, we have designed a user test. This is a more complex problem than standard spam filtering as we assume that context determines the level of tolerance a user has at any time for reading a message. We believe that given the situation, the willingness of a user to read a message changes in a predictable manner [10]. Therefore an evaluation of this system must account for the context that was used to support the decision to pass the user a message.

We will perform the evaluation with a number of real users and their real e-mail. The transactions take the following form: messages are sent to the user via Bluetooth's Push protocol. The user's phone then gives them the option of accepting the message or rejecting it. This user feedback is logged by the Scatterbox application running on a Construct node. The feedback is then used to determine the appropriateness of Scatterbox's decision to send the message to the user.

Our evaluation will estimate the utility of context-aware message filtering by quantifying both the number of unwanted messages that are sent to a user (false positives), and the number of important messages that are not sent to the user (false negatives). Feedback elicited from rejection of unwanted messages indicates false positives, but an alternative approach is needed to account for false negatives (messages which should have been sent but were not). For the purposes of evaluation we capture this form of feedback by occasionally sending messages to the user that Scatterbox does not believe the user will want to see. Our evaluation will count rejections of these messages as true negatives (and thus appropriate behaviour) and acceptances of these messages as false negatives (i.e., inappropriate). We believe that it is more important for Scatterbox to avoid false positives than false negatives — since the user will always check their inbox eventually — and that our evaluation will bear this out. (Interestingly this is the opposite of what one would expect from the perspective of spam filtering.)

Further to performing a live evaluation, we propose to amass a data set of real context and behaviours over the course of this user test and use this to perform a number of offline evaluations. By gathering all relevant context features with Scatterbox's decision to send messages and the user's feedback, we will be able to investigate which features were most important in predicting correct behaviour using simple feature selection [17]. By examining how changes in context affect situation and behaviour, we will be able to examine the ability of Scatterbox to consistently and smoothly respond to its environment.

7 Conclusions and Future Work

We developed a system, Scatterbox, that determines a user's situation by composing numerous sources of contextual data. This system can then intelligently forward useful messages to a user's mobile device based on their current situation. These messages comprise both important e-mails that the user has received while they were away from their computer, and also contextual notifications of important events, such as meeting reminders derived from the user's calendar. The system notices changes in a user's situation, and reacts accordingly.

Scatterbox was created using a context infrastructure (Construct), which has been used for context reasoning in many other application areas, such as in location awareness for health care [18], smart homes [19], and recommender systems [20].

We have also designed an evaluation of this system, which uses the rate of rejection of messages to determine whether the system classified the situation correctly and if it acted appropriately.

The next step in Scatterbox's development is the addition of Machine Learning techniques to correct its distribution algorithms in a similar way to that done by intelligent spam filtering techniques. Research is also ongoing regarding the use of truth maintenance to reduce the need for continually resolving recurring inconsistencies within a data store.

References

1. Mark Weiser. The Computer for the 21st Century. *Scientific American*, 265(3):94–104, September 1991.
2. Emmanuel Munguia Tapia, Stephen S. Intille, Louis Lopez, and Kent Larson. The design of a portable kit of wireless sensors for naturalistic data collection. In *4th International Conference on Pervasive Computing, Dublin, Ireland, May 7-10, 2006*, pages 117–134, 2006.

3. Lars Erik Holmquist, Friedemann Mattern, Bernt Schiele, Petteri Alahuhta, Michael Beigl, and Hans-W. Gellersen. Smart-its friends: A technique for users to easily establish connections between smart artefacts. *Lecture Notes in Computer Science*, 2201:116–122, 2001.
4. Tim Finin Harry Chen and Aravind K. Joshi. A context broker for building smart meeting rooms. In *The Knowledge Representation and Ontology for Autonomous Systems Symposium*, AAAI Spring Symposium, March 2004.
5. Vincent Rialle, Nancy Lauvernay, Alain Franco, Jean-Franois Piquard, and Pascal Couturier. A smart room for hospitalised elderly people: essay of modelling and first steps of an experiment. *Technology and Health Care*, 5(7):343 – 357, January 1998.
6. Antti Oulasvirta. The fragmentation of attention in mobile interaction, and what to do with it. *interactions*, 12:16–18, 2005.
7. Jason Pascoe. The stick-e note architecture: Extending the interface beyond the user. In *IUI '97: Proceedings of the 2nd international conference on Intelligent user interfaces*, pages 261–264, New York, NY, USA, 1997. ACM Press.
8. Yasuto Nakanishi, Takayuki Tsuji, Minoru Ohyama, and Katsuya Hakozaki. Context aware messaging service: A dynamical messaging delivery using location information and schedule information. *Personal Ubiquitous Comput.*, 4(4):221–224, 2000.
9. Nancy Miller, Glenn Judd, Urs Hengartner, Fabien Gandon, Peter Steenkiste, I-Heng Meng, Ming-Whei Feng, and Norman Sadeh. Context-aware computing using a shared contextual information service. In *Pervasive'04, "Hot Spots"*, Vienna, April 2004.
10. Joyce Ho and Stephen S. Intille. Using context-aware computing to reduce the perceived burden of interruptions from mobile devices. In *CHI '05: Proceedings of the SIGCHI conference on Human factors in computing systems*, pages 909–918, New York, NY, USA, 2005. ACM Press.
11. Stephen S. Yau, Dazhi Huang, Haishan Gong, and Yisheng Yao. Support for situation awareness in trustworthy ubiquitous computing application software. *Software: Practice and Experience*, 36(9):893–921, July 2006.
12. Anind Dey. Understanding and using context. *Personal and Ubiquitous Computing*, 5(1):4–7, 2001.
13. Karen Henricksen. *A Framework for Context-Aware Pervasive Computing Applications*. PhD thesis, The School of Information Technology and Electrical Engineering, University of Queensland, September 2003.
14. Graeme Stevenson, Lorcan Coyle, Steve Neely, Simon Dobson, and Paddy Nixon. Construct — a decentralised context infrastructure for ubiquitous computing environments. In *IT&T Annual Conference, Cork Institute of Technology, Ireland*, 2005.
15. Simon Dobson, Lorcan Coyle, and Paddy Nixon. Hybridising events and knowledge as a basis for building autonomic systems. *IEEE TCAAS Letters*, 2007. To appear.
16. Juan Ye, Adrian K. Clear, and Simon Dobson. Towards a formal semantics for pervasive adaptive systems. *Computer Journal*, 2007. To Appear.
17. Huan Liu and Hiroshi Motoda. *Feature Selection for Knowledge Discovery and Data Mining*. Kluwer Academic Publishers, Norwell, MA, USA, 1998.
18. Lorcan Coyle, Steve Neely, Paddy Nixon, and Aaron Quigley. Sensor aggregation and integration in healthcare location based services. In *First Workshop on Location Based Services for Health Care (Locare'06), Nov 28 2006, Innsbruck Austria*, 2006.
19. Lorcan Coyle, Steve Neely, Gaëtan Rey, Graeme Stevenson, Mark Sullivan, Simon Dobson, and Paddy Nixon. Sensor fusion-based middleware for assisted living. In *Proc. of 1st International Conference On Smart homes & heath Telematics (ICOST'2006) "Smart Homes and Beyond"*, pages 281–288. IOS Press, 2006.
20. Lorcan Coyle, Evelyn Balfe, Graeme Stevenson, Steve Neely, Simon Dobson, Paddy Nixon, and Barry Smyth. Supplementing case-based recommenders with context data. In *1st International Workshop on Case-based Reasoning and Context Awareness at ECCBR*, 2006.

Learning at your Leisure: Modelling Mobile Collaborative Learners

Anders Kofod-Petersen and Sobah Abbas Petersen

Department of Computer and Information Science,
Norwegian University of Science and Technology,
7491 Trondheim, Norway
{anderpe|sap}@idi.ntnu.no,
http://www.idi.ntnu.no/

Abstract. Advances in ubiquitious and mobile technologies have facilitated learners to continue their learning outside their classrooms, when and where they desire. Learners are now able to access their learning resources and interact with their peers and teachers through technology. The design and creation of such learning spaces pose many challenges. The learner's context defines the needs of the learner at any time. To meet the needs of the learner, a set of services must be available anytime and anywhere. To establish the learner's context, a model of the learner is essential. This paper focuses on modelling the learner and proposes the use of stereotype modelling to determine the context of the learner and to propose a set of services to the learner to support her learning process. A scenario describes a mobile collaborative learner and the modelling concepts are described using an example.

1 Introduction

Advances in ubiquitious and mobile technologies have facilitated learners to continue their learning outside their classrooms, when and where they desire. Recent research has shown that learners desire to continue their learning processes outside of their classrooms and combine learning with leisure [1] or learn spontaneously, in response to recent, current or imminent situations, but without any time being set aside for it [2]. The term mobile learning has become popular in recent times to denote learning that is conducted while the learner is on the go or when the learner is mobile. Many authors refer to mobile learning as supporting learning via mobile devices such as handheld PDAs or mobile phones, where access to learning material is provided via the mobile device. We consider mobile learning as the learning that takes place or the learning support that is provided to a learner when the learner is mobile, independent of the technology. Mobility is often considered in terms of time and physical space or location [3]. Hence, when the learner is outside the classroom and classroom hours, the learner is mobile and needs access to the learning resources such as books and literature. Similarly, access to people that support the learning process is very important.

They could be the teacher, peer learners as well as any other people that could support the learning process.

Providing access to learning resources and facilitating interaction among learners is an important support for the learning process. To be able to provide such support anytime and anywhere, we envisage the learners' environment being capable of extending this support using technology. Several university campuses have embarked on providing easy access to learning resources for its students by creating wireless hotspots around the campus. Several cities in Europe have launched projects to transform the city into a wireless zone, facilitating easy access to resources and people. The city of Trondheim's Wireless Trondheim initiative is one such project[1]. There is a trend towards ambient intelligence and intelligent environments.

The IST Advisory Group in European Union (ISTAG) defines ambient intelligence as human beings surrounded by intelligent artefacts, supported by computing and network technology embedded in everyday objects. More importantly, the environment should be aware of the presence of a person, perceive the needs of this person and respond intelligently to them in a relaxed an unobtrusive manner [4]. To respond to the needs of the user, there is a need for a user model.

When using systems that require user models to adapt its services some considerations as to the nature of these models and the domain in question must be made. In domains where the user group is highly homogeneous canonical user models are likely to be the best option, whereas in domains where the user group is highly heterogeneous specific user models are likely to be the preferred option. We have earlier suggested an ambient intelligent architecture and implementation where a one-to-one coupling between the system instance and the user exists [5]. In this case the system builds an implicit user model over time, by storing experienced situations as cases and utilising case-based reasoning as the means of adapting its behaviour to the user's idiosyncrasies.

We have previously suggested how to use this ambient intelligent system in a mobile collaborative learning domain [6]. However as users of an ambient intelligent system in this setting presumedly is much less persistent, thus not allowing the case-based reasoning system enough time to adapt its behaviour, some explicit user model is required.

The work presented here suggests the use of stereotype modelling as the means of realising an explicit user model in an ambient intelligent setting. The services that are required by the user are delivered by one or more service providers. Similar to a model of the user and her needs, there is a need for a model of the services and the service providers. A detailed discussion of this is beyond the scope of this paper.

The rest of the paper is structure as follows: First and overview of what constitutes a mobile learner is given. This is followed by the background for the work presented here, as well as selected related work. Thirdly, a scenario containing a mobile language learner sets the stage for the use of ambient intelligent systems.

[1] http://www.tradlosetrondheim.no/

Fourthly, a description of how stereotypes are used to model a learner is given. This is followed by a short example revisiting the scenario. The paper ends with a summary and pointers to future work.

2 Mobile Collaborative Learner

Language learning is an area where the learner has the need to complement the classroom learning with experiences outside of the classroom. In language learning, there is a strong component of informal learning that takes place outside the classroom through interactions with the society that complements and reinforces the formal learning that takes place in classrooms, see Figure 1. Learners often experience situations where they would like to continue their learning processes as they go about their daily lives and while they are mobile. An exploratory study of language learners' use of technology have shown that learners desire to combine learning with leisure and entertainment [1]. For example, a TV program may stimulate them to learn new words in a particular subject area or they may feel the need to learn the appropriate usage of a word or a phrase that appears in a conversation.

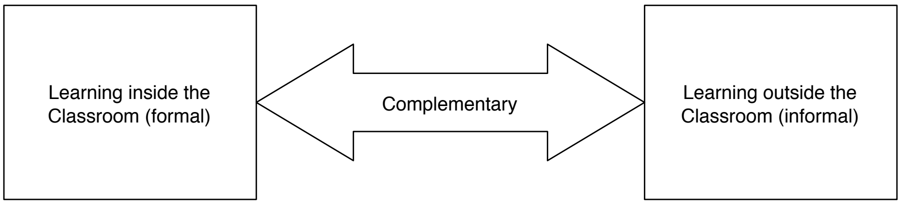

Learning inside the Classroom (formal)
Complementary
Learning outside the Classroom (informal)

Fig. 1. Complementary Language Learning Situations

Learning as a social activity has been the discussion of several books and articles (e.g. [7] and [8]). Many scholars agree that learning is most effective when it takes place as a collaborative rather than an isolated activity and when it takes place in a context relevant to the learner [9]. In our work, a socio-constructivist approach to language learning is considered where learning is supported by collaboration and the interaction with others [10]. In collaborative language learning, the social construction of knowledge occurs through the learners? interaction with other learners, teachers and various communities that support the learning process. Similarly, conversation has been identified as an important aspect of learning where learning is considered as a continual conversation with other learners and teachers [11]. This is particularly important in language learning where the learning is strongly influenced by situations [12] and culture [13]. It is important for a language learner to learn in an appropriate cultural context and to interact with communities that exist in the cultural setting and practise the language with native speakers, fluent speakers and peers. A learner that is mobile and collaborates with other people as a part of her learning process is a mobile collaborative learner.

3 Background and Related Work

To achieve environments such as the ones described in the ISTAG report [4], several different paradigms and technologies must be integrated. The complementary use of a number of technologies is essential, drawing from their strengths and characteristics. Such scenarios describe an ambient intelligent environment. The concept of ambient intelligence has been described as humans being surrounded by intelligent interfaces supported by computing and networking technology that is embedded in everyday objects such as furniture, clothes and the environment [4]. The environment should be aware of the presence of a person (the user) and perceive the needs of the person and adapt and respond intelligently to these needs. We see ambient intelligence as a combination of a number of paradigms; ubiquitous computing [14], pervasive computing [15] and artificial intelligence [16], see Figure 2. The ubiquitous computing aspect addresses the notion of accessibility of the technology, where the technology and connectivity is available through everyday objects that are in the user's environment. Artificial intelligence techniques provide the context awareness to establish the user's needs and the appropriate response and the pervasive computing aspect supports the architectural aspects to realise the situation.

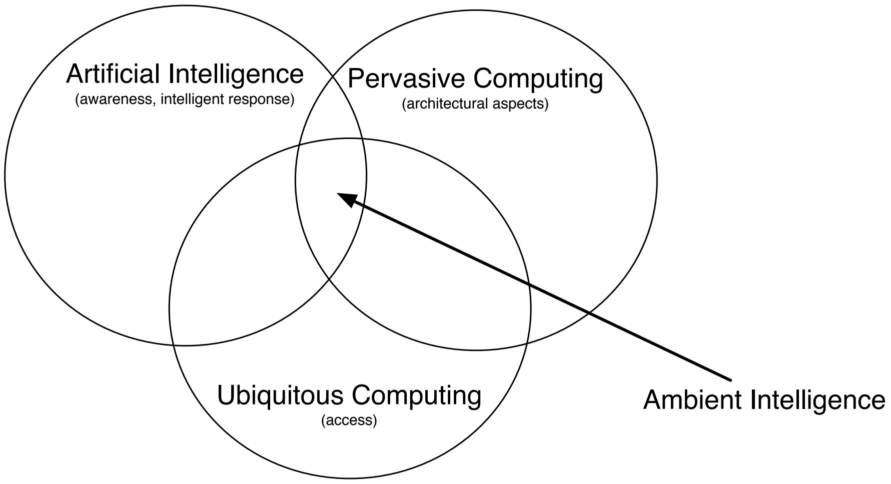


Fig. 2. Paradigms Related to Ambient Intelligence

Similar to the paradigms, several different technologies are required to achieve the desired effects. Mobile technologies, such as mobile phones and handheld devices, provide access while on the move; personal technologies, provide the means of accessing the appropriate content when desired, i.e. the personalisation and contextualisation of the content for the user; embedded and ambient technologies, such as shared displays, and interactive technologies, such as interactive white boards, are becoming popular in different environments for collaborative work and learning. An example of the use of shared displays in an office environment is described in [17] and in a learning environment in [18].

A combination of technologies is often required to obtain the appropriate set of services that are required. An example of blended technologies is described in [18], where an interactive shared display in combination with email, SMS and other applications are proposed for collaboration among teacher trainees during their practice period. In [19], they describe the combined use of interactive television with SMS for language learners while watching TV.

We have earlier introduced an ambient intelligent architecture [5] and demonstrated how an implementation can support context awareness and context sensitivity in a hospital ward domain [20,21]. The existing system proposes a subjective perspective on context and situations, which is well suited for domains where the user is a long term user, giving the system a possibility to adapt to the specific user's idiosyncrasies. However, as discussed in [22] the use of an ambient intelligent system in domains where users are non-persistent poses some new challenges to how users are perceived and modelled. Specifically, in domains where users are persistent no explicit user model is required as it is implicitly gathered over time by observing the user's behaviour. In domains where users are non-persisten, such as a tourist domain or mobile learner domain, it is not possible to apply a costly knowledge acquisition process regarding a users, and it is impossible to learn the specific user's behaviour over time. However, to allow an ambient intelligent system to perceive its users' needs and respond intelligently to them [4], some sort of explicit user model is required.

Jayaputera et at. describes the eHermes system, which is a a multi-agent system for adaptation of content delivered to users based on their context [23]. The context model used encompasses the capabilities of the user's device and the bandwidth available. The main idea is, based on user request, to generate plans and execute them in run time. Jayaputera et at. approaches the problem of user modelling by employing stereotypes, in the tradition of Rich [24]. The authors use a "User Profile" to describe the specific user and a "Device Profile" to describe the devices available. The stereotypes contain two parts: The classification part and the predictive part. The classification part contains context independent information, such as the user's demographical information and device information; and context dependent information, such as the user's current location and the user's current environment. The predictive part contains predictions about the user's needs, preferences and behaviour of their devices. When a user or a device enters eHermes for the first time, the user or device is mapped to an existing stereotype, which is used for personalisation.

4 Scenario

Astrid is a German student that is new in Trondheim. She has just started the "Norwegian for Beginners" course at the university and is hoping to get to know the city of Trondheim and learn some Norwegian as soon as possible. The city of Trondheim has a number of services to support newcomers to the city and to help them in learning Norwegian. Astrid has registered to these services and provided information about her interests such as history and outdoor life.

Astrid walks around the city hoping to learn some new Norwegian words and to learn more about the city. She has her mobile phone with a camera and voice recording capabilities and she activates the services that she has registered on to. She is presented with an augmented map of the city that helps her to find audio traces that others have left behind. She arrives at the city square where there is a farmers' market on Saturdays. She is intrigued to see the different kinds of berries and uses the glossary service via her mobile device to learn the Norwegian names for the products in the farmers' market. When the glossary service is activated, she is also informed that a certain chapter in her Norwegian textbook relates to the subject that she's looking up in the glossary service and she is asked if she wishes to view that chapter. As she walks around the square, her current location on the augmented map is highlighted and her mobile phone gives an audio signal to inform her that audio files that have been left behind by previous visitors are available in that location. She activates the audio trails to listen to the experience of previous visitors. She is very excited to hear that one of the previous visitors had actually been picking some cloudberries[2], just outside Trondheim and had left directions to that place. Astrid leaves an audio recording of her impressions of the city centre.

She receives a message on her mobile phone that there will be a concert by a local group in the square starting in 10 minutes. She notices that several people have started gathering around a stage on one side of the square. Astrid moves towards the group to find a place with a good view of the stage. She is keen to share her city experience with her classmates from the Norwegian class. She activates a service so that other users of the system and who are in the vicinity would know that she is in the area. In a short while, another German student Helga sends her a message saying that they could meet up.

5 Learner Modelling

As aforementioned, an explicit user model has not been a focus in our work so far. However, as the nature of the mobile learner dictates that our system moves from a subjective perspective on the world and its user, to a more objective view, an explicit user model is required. The work presented here suggests the use of stereotype modelling in the tradition of Rich [24] as the means of constructing a user model.

[2] Cloudberry (Rubus chamaemorus) is a wild berry that is abundant in marshes in mid-northern Norway

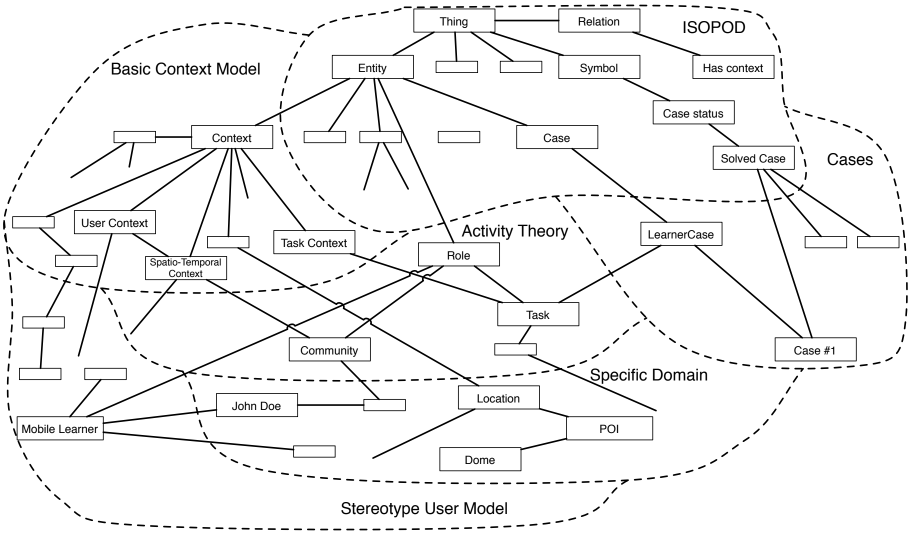


Fig. 3. Knowledge Model Layers

The user model is an integrated part of the system's knowledge model. This model is depicted in Figure 3. With the addition of the user model, the knowledge model now contains six different areas all interconnected in a multi-relational semantic net. The `ISOPOD` model contains the basic constructs required to explicate the rest of the model, as well as relations and concepts required for the case-based reasoning to function [25,26]. The `Basic Context Model` contains a meronomy of knowledge that is used to structure the perceived data into a coherent structure [27,5]. The `Activity Theory` part contains the knowledge captured through the socio-technical analysis required to model ongoing activities [28]. The `Cases` contains perceived situations represented as episodes in the case-based reasoning framework. Finally, the `Specific Domain` contains factual knowledge about the domain of discourse. The figure does not give credit to the multi-dimentional nature of the knowledge model since it does not carry well in two dimensions.

The `Stereotype User Model` is a directed acyclic graph in a "generalisation of" hierarchy, see Figure 4. The root node of the graph contains the most general stereotype (`any-person`), and the leaf nodes the most specific. Each of the stereotypes contain a set of *facets* with a *value* and *rating*. Following Rich, each of the facets' values are in a linear scale ranging from -5 to 5, where a positive value indicates that the stereotype is positive to the facets, and a negative value indicates that the stereotype is negative to the facet. The ratings range from 0 to 1000 indicating the degree of certainty in the facet-value pair. Except for the `any-person` stereotype, no stereotype contains all facets.

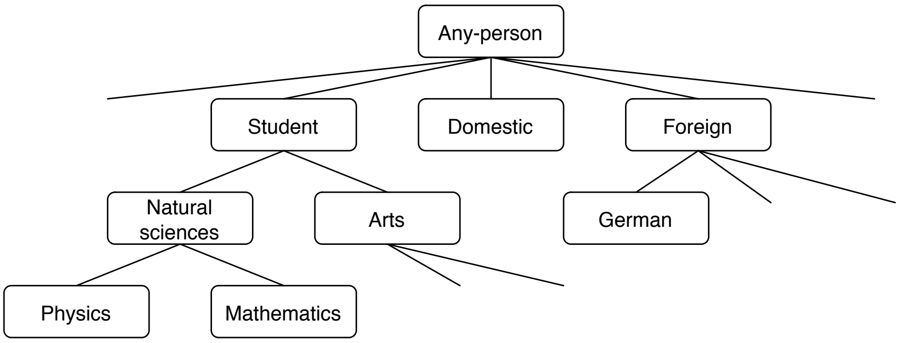


Fig. 4. Hierarchy of Stereotypes

The specific user model, know as the *User Synopsis (USS)* in Rich, contains a specific model for each of the individual users. This model is summary of the different stereotypes that is found relevant for the user. In addition to the facets, values and rating triplet, each tuple also contains a justification; that is the name of the stereotype that has supplied the specific triplet.

Initially this model is constructed partly by supplying information that the user has submitted before entering the course, such as sex, age, nationality and major; and partly by allowing the user to fill in a short questionnaire before instantiating the service. This questionnaire contains questing related to the person's interests besides the academic, such as a interest in history, concerts, or outdoor life. All of the information supplied by the user function as a way of activating triggers for the stereotypes. For an example, being a female, German physicist with a interest in the outdoors, might trigger the `female`, `german`, `physics student`, and `outdoor person` stereotypes.

For the specific user model to function the tasks that the ambient intelligent system is to perform must be modelled in accordance with the user model. That is, each of the possible task has facets and values assigned to them.

The initial stereotypes are constructed based on available knowledge about students attending language courses at the learning facility. Likewise, the assignments of facets and values to the different tasks are based on a combination of the curriculum required for the language students, and prior experiences from the learning facility and the local tourist centre.

6 An Example

To demonstrate the sequence of events that takes place when a new user approaches the system. Lets revisit Astrid who is learning Norwegian as part of her visiting the university in Trondheim.

Initially Astrid has filled in the necessary form when she applied for the Norwegian course. From this form we learn that Astrid is doing her master degree in physics, she is German and female. As aforementioned this information will trigger stereotypes such as `physics student`, `master student`, `german` and `female`. In addition she fills in a short questionnaire when signing up for the Wireless Trondheim service. Here she tells the system that she is interested in history and outdoor life, which triggers stereotypes such as `history person` and `outdoor person`. Once the initial specific user model has been constructed the following sequence is executed:

1. Select suitable sequence of tasks for the user model
2. Construct sequence of actions based on the stereotype modelling
3. Populated the sequence of actions through activation of a virtual enterprise
4. Ask the user to verify the plan. Return to 1 until the user is satisfied

The system can now construct a sequence of tasks that the ambient intelligent system and user are to carry out together. These tasks might be interacting with the locals, suggesting relevant cultural events, experiencing the outdoors and supplying dictionary services. There are many possible actions that might execute any give task. For an example, a map service might be supplied by any number of commercial actors. Therefore, the system can now, taking into account any stereotype given constraints, construct a sequence of actions that will execute all the required tasks. Finally, the virtual enterprise that are to cooperate in executing all the actions are gathered.

When the sequence of actions has been constructed it is presented to the user. The system might have selected some unsatisfactory stereotypes leading to a sequence of actions that the user does not agree with. It is now possible to lead a type of conversation where the system suggests new action sequences until the user is satisfied, thereby also updating the specific user model,

7 Discussion and Future Work

This paper has described why a user model is a requirement for an ambient intelligent system to function. It has further demonstrated how stereotype modelling is a promising candidate for quickly constructing user models with minimal effort from the user. However, some issues are still to be resolved.

Contrary to the original Grundy system described by Rich, and ambient intelligent system does not necessarily has the privilege of being able to conduct a conversation with the user. More often the primary interface between the system and its users are behavioural interfaces. Here users can observe the results of the system though its behaviour and influence the system through their behaviour. Likewise, the system can observe the effects of its behaviour by observing its users' behaviour, thereby reasoning on the success of its behaviour. Thus, to adapt the specific user model the system has to monitor the users' behaviour and look for triggers that might reinforce the existing user model, or in case of unexpected behaviour adapt its model of the specific user to include previously non important stereotypes.

Further, the nature of an ambient intelligent system dictates that is must adapt to the user continuously. Currently we have not defined how the populated sequence of actions might change if the user changes her mind or something unexpected happens. The main challenges here does not lie in adapting the specific user model based on behavioural input from the user. Rather, the challenge lies in restructuring the virtual enterprise constituting the populated sequence of actions, in particular when dealing with commercial partners. Some measures for sharing profit and cost between the affected partner must be conceived. We are currently investigating ways of agreeing on common business models between commercial partners in ambient intelligent systems. However, no conclusive results have been achieved yet.

Acknowledgements

Parts of this work has been supported by Accenture Innovation Lab Norway and the MOTUS2 project, funded by the Norwegian University of Science and Technology.

References

1. Thornton, P., Sharples, M.: Patterns of technology use in self-directed japanese language learning projects and implications for new mobile support tools. In: Proceedings of the International Workshop on Wireless and Mobile Technologies in Educations (WMTE). (2005)
2. Eraut, M.: Non-formal learning and tacit knowledge in professional work. The British Journal of Educational Psychology **70** (2000) 113–136
3. Cooper, G.: The mutable mobile: Social theory in the wireless world. In: Wireless World, Social and Interactional Aspects of the Mobile Age. Springer Verlag (2001)
4. Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., Burgelman, J.C.: Istag scenarios for ambient intelligence in 2010. Technical report, IST Advisory Group (2001)
5. Kofod-Petersen, A., Mikalsen, M.: An architecture supporting implementation of context-aware services. In Floréen, P., Lindén, G., Niklander, T., Raatikainen, K., eds.: Workshop on Context Awareness for Proactive Systems (CAPS 2005), Helsinki, Finland, HIIT Publications (2005) 31–42
6. Petersen, S.A., Kofod-Petersen, A.: Learning in the city: Context for communities and collaborative learning. In: Proceedings of the 2nd International Conference on Intelligent Environments (IE06), Athens, Greece, The Institution of Engineering and Technology (2006)
7. Wenger, E.: Communities of Practice: Learning, Meaning, and Identity. Cambridge University Press (1998)
8. Lave, J., Wenger, E.: Situated Learning: Legitimate Peripheral Participation. Cambridge University Press (1991)
9. Godden, D.R., Baddeley, A.D.: Context-dependent memory in two natural environments: on land and underwater. British Journal of Psychology **66** (1975) 325–331

10. Vygotsky, L.S.: Mind in Society. Harvard University Press, Cambridge, MA (1978)
11. Sharples, M.: Learning as conversation: Transforming education in the mobile age. In: Proceedings of the Conference on Seeing, Understanding, Learning in the Mobile Age. (2005)
12. Ogata, H., Yoneo, Y.: Knowledge awareness for a computer-assisted language learning using handhelds. International Journal of Continuous Engineering Education and Lifelong Learning **14** (2005) 435–449
13. Tang, R.: The place of "culture" in the foreign language classroom: A reflection. The Internet TESL Journal (1999)
14. Weiser, M.: The computer for the 21st century. Scientific American (1991) 94–104
15. Satyanarayanan, M.: Pervasive computing: Vision and challenges. IEEE Personal Communications **8** (2001) 10–17
16. Russel, S.J., Norvig, P.: Artificial Intellignece: A Modern Approach. 2nd edn. Prentice Hall (2002)
17. Divitini, M., Farshchian, B.A.: Shared displays for promoting informal cooperation: an exploratory study. In Darse, F., Dieng, R., Simone, C., Zacklad, M., eds.: Cooperative Systems Design – Scenario-based Design of Collaborative Systems, IOS Press (2004) 211–226
18. Morken, E.M., Divitini, M.: Blending mobile and ambient technologies to support mobility in practice based education: the case of teacher education. In: Proceedings of the 4th world conference on MLearning MLearn 2005, Cape Town, South Africa. (2005)
19. Fallakhair, S., Pemberton, L., Griffiths, R.: Dual device user interface design for ubiquitous language learning: Mobile phone and interactive television (itv). In: Proceedings of IEEE International Conference on Wireless and Mobile Technology for Education (WMTE). (2005)
20. Cassens, J., Kofod-Petersen, A.: Using activity theory to model context awareness: a qualitative case study. In: Proceedings of the 19th International Florida Artificial Intelligence Research Society Conference, Florida, USA, AAAI Press (2006) 619–624
21. Kofod-Petersen, A., Aamodt, A.: Contextualised ambient intelligence through case-based reasoning. In Roth-Berghofer, T.R., Göker, M.H., Güvenir, H.A., eds.: Proceedings of the Eighth European Conference on Case-Based Reasoning (ECCBR 2006). Volume 4106 of Lecture Notes in Computer Science., Ölüdeniz, Turkey, Springer Verlag (2006) 211–225
22. Kofod-Petersen, A.: Challenges in case-based reasoning for context awareness in ambient intelligent systems. In Minor, M., ed.: 8th European Conference on Case-Based Reasoning, Workshop Proceedings, Ölüdeniz / Fethiye, Turkey (2006) 287–299
23. Jayaputera, G., Alahakoon, O., Cruz, L.P., Loke, S.W., Zaslavsky, A.B.: Assembling agents on-demand for pervasive wireless services. In Mahmoud, Q.H., ed.: Wireless Information Systems, ICEIS Press (2003)
24. Rich, E.: Users are individuals: Individualizing user models. International Journal of Man-Machine Stuides **18** (1983) 199–214
25. Aamodt, A.: A knowledge-intensive, integrated approach to problem solving and sustained learning. PhD thesis, University of Trondheim, Norwegian Institute of Technology, Department of Computer Science (1991) University Microfilms PUB 92-08460.
26. Aamodt, A.: Knowledge-intensive case-based reasoning in creek. In Funk, P., Calero, P.A.G., eds.: Advances in case-based reasoning, 7th European Conference, ECCBR 2004, Proceedings. (2004) 1–15

27. Kofod-Petersen, A., Mikalsen, M.: Context: Representation and reasoning – representing and reasoning about context in a mobile environment. Revue d'Intelligence Artificielle **19** (2005) 479–498
28. Kofod-Petersen, A., Cassens, J.: Using activity theory to model context awareness. In Roth-Berghofer, T.R., Schulz, S., Leake, D.B., eds.: Modeling and Retrieval of Context: Second International Workshop, MRC 2005, Revised Selected Papers. Volume 3946 of Lecture Notes in Computer Science., Edinburgh, UK, Springer Verlag (2006) 1–17

Linking Context Modelling and Contextual Reasoning[*]

Dongpyo Hong, Hedda R. Schmidtke, Woontack Woo[**]

GIST U-VR Lab.
Gwangju 500-712, Korea
{dhong,schmidtk,wwoo}@gist.ac.kr

Abstract. In this paper, we discuss a novel perspective on ontology-based context modelling that makes it easy to combine context models and contextual reasoning mechanisms. On the context modelling side, we outline our idea of a user-centric context model based on the six fundamental context parameters of *who*, *when*, *where*, *what*, *how*, and *why* (5W1H); on the contextual reasoning side, we introduce syntax and semantics for a simple logical language and sketch a tableau mechanism for reasoning. The model-theoretic semantics for this logical language is like the context model based on the parameters of 5W1H. With the common semantics, it is then easy to show the link between context modelling and contextual reasoning.

1 Introduction

For any research on context-aware, intelligent computer systems, a fundamental question is how to represent context. As perspectives of research and targeted types of context-awareness differ, the specific properties of representations of context also differ (cf. the range of perspectives surveyed in [3]). However, the concept to be represented, that is *context*, is the same; and we can expect that there are interfaces and mappings between different types of *representations of context*. Thus, we present an approach to describe such an interface for the example of a representation of context from the area of *context modelling* and a representation of context from the area of *contextual reasoning*.

A definition of context that has been accepted widely in the area of context-aware applications has been given by Dey and Abowd [6]: "Context is any information that can be used to characterise the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves." A problem with this definition is that it does not differentiate between context and information about context. A more recent definition by Bardram [1] better reflects the fact that *context* exists outside a representation system: " 'Context'

[*] This research is supported by the UCN Project, the MIC 21st Century Frontier R&D Program in Korea.

[**] Corresponding author

refers to the physical and social situation in which computational devices are embedded." A *context model* can then be conceived of as a data model suitable for storing information about the context of a certain interaction event.

Mobile context-aware computing has to cover issues of sensor reliability, ad hoc network communication, software development support, reasoning and inference, usability, and privacy management. Many of the main differences between approaches can accordingly be traced back to emphasis on one or the other aspect, such as sensor-fusion in [21], networking in [20], and development of context-aware applications in [11]. Most recently, ontology-based approaches [22, 18, 9] have gained importance to answer the demands of heterogeneous application environments. The key idea of ontology-based context modelling is that applications using the context model also have to agree on a common ontology, that is, a set of basic concepts defined in a formal language, which developers can use to specify application specific concepts. Application concepts, being founded upon the same basic concepts, can then be used for communication between different applications.

When human beings reason or communicate about objects and events in the environment, they usually abstract from certain aspects, and reason within a context. When we reason about space, for instance, we may use *to the West of* as a transitive relation (cf. the calculus in [15]). This assumption is valid as long as we suppose a sufficiently small local area of context, as *to the West of* is globally a cyclic relation: Denmark is to the West of Korea, Korea is to the West of Canada, and Canada is to the West of Denmark; within the local context of a city or country, in contrast, *to the West of* can be used in the same manner as *to the North of*, i.e. as if it was a transitive, acyclic relation.

Research in *contextual reasoning* investigates how such locally valid theories can be connected and how inferences can be made across context boundaries: Benerecetti, Bouquet, and Ghidini [2] investigate the basic principles and tasks of contextual reasoning. They use the metaphor of *context as a box* containing the contextualised logical sentences together with a list of parameters and values for these. These parameters establish the link between the contextualised sentences and the surrounding or neighbouring boxes. In [2], three dimensions of context dependence are distinguished: contextualised representations can be *partial*, *approximative*, or *perspectival* representations. The three dimensions of context dependence can be illustrated with respect to the parameter of spatial context: changing from a global to a partial view of space can be identified with spatially focusing on a sufficiently small local area; a change of spatial perspective can be identified with a change of reference frames; and changing the degree of approximativeness can be related to coarsening and refinement of spatial granularity. The focus of this article is on the aspect of *partial* representations.

The key idea of our research was to link a user-centric context model for context-aware applications (Sect. 2), which can provide a system with access to sensory information, and a logic-based contextual reasoning mechanism (Sect. 3) with a common semantics that has its roots in six fundamental parameters of context: both representations of context are understood as describing circumstances of a certain *interaction* as a proposition stating that someone (*who*) interacts somehow (*how*) and for a certain reason (*why*) with something (*what*) at a given time (*when*) and place (*where*). We sketch how contextual information retrieved from sensors can be stored and accessed with the context model and how information stored in the context model can be translated into the logical language (Sect. 4).

2 User-centric Context Model

Regarding context model, there have been many research activities from artificial intelligence to mobile, ubiquitous, and pervasive computing. For instance, Chen and Kotz [4] gave a survey of various definitions of context and its applications, particularly in the area of mobile computing. Dey and Abowd [6] explored context from the area of context-aware computing. However, most context models take an application centred perspective. From the perspective of human-computer interactions in contrast, we are more interested in how contextual information is perceived by users rather than by devices, services, or applications. Thus, we define context as user-centric context, that is, described from the perspective of the user: *User-centric context* is represented as a sequence of explicit and implicit information that occurs in a user's interactions with applications.

Each sequence of the proposed context model consists of a set of WHO, WHEN, WHERE, WHAT, HOW, and WHY (for short: 5W1H) which can provide the bridge into the ontology and thus into contextual reasoning (Sect. 3). Sensory information, inferred information, and information entered by a user are sorted accordingly into one of the six categories. Table 1 shows what each category represents in a sequence. Explicit information is the user's direct inputs

Table 1. Context categories

Category	Description	Examples
WHO	Basic user information	name, gender, birthday
WHEN	Time	time stamp, time of day, season
WHERE	Location	coordinate (x,y), place, region
WHAT	Relevant objects	applications, services, commands (application dependent)
HOW	Ongoing processes	signals from sensors, e.g. current activity of the user (sensor dependent)
WHY	Users' intentions	stress, emotion, future events from a schedule

(e.g., from keyboard, mouse, pen) and signals from sensors (e.g., acceleration, temperature, GPS). Implicit information is information inferred from explicit information. In order to turn sensory raw data into information required by services, they have to be accumulated, processed, and integrated [12]. According to the state of processing, user-centric context can be categorised as *preliminary*, *integrated*, and *final context*.

Preliminary Context (PC) stores primitive data from sensors and primitive features from the data.

Integrated Context (IC) contains accumulated preliminary contexts and inferred information, particularly from sensor-fusion.

Final Context (FC) is the context representation received from and sent to applications. Both a user's expectations (e.g., privacy concerns) and the application's demands for information have to be regarded whenever information is sent to an application.

For example, explicit user input can be the birthday of the user like `birthday` $= x$ in the preliminary context. In the integrated context, it can be expressed as `age` $= f(x, d)$ where $f()$ is a function for counting the number of years between two dates or time stamps.

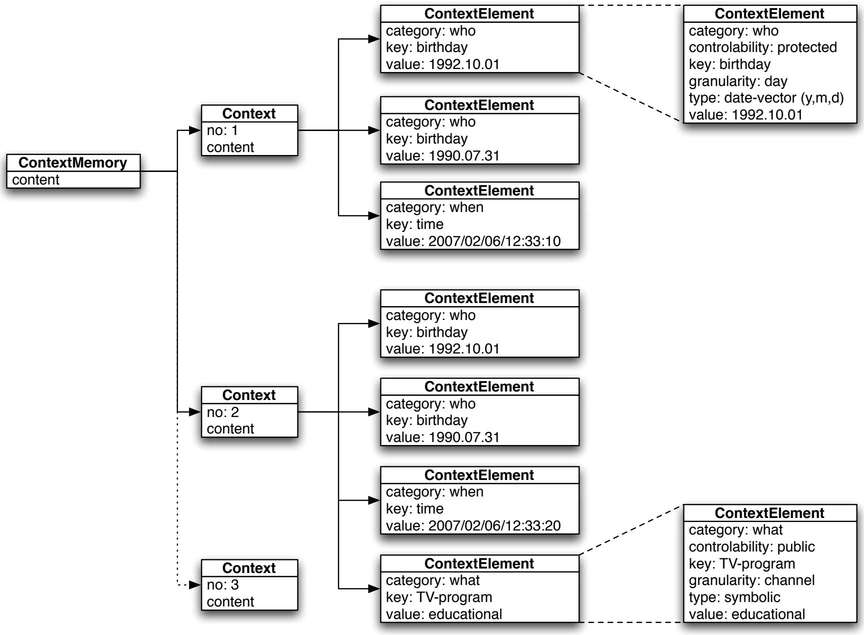


Fig. 1. *Context* objects (whether PC, IC, or FC) consist of *context elements* (here simplified with detailed views for two examples) and are collected into *context memory*.

The proposed context model consists of three parts as follows (Fig. 1).

ContextElement is a basic type in the proposed context model and consists of six attributes: *category* (one of 5W1H), *controlability*, *key*, *granularity*, *type*, and *value*.

Context objects store all information available at a certain time and thus contain a full description of a context as it is retrieved from sensors. The Context class is implemented as a set of contextual elements. For accessing elements in a context object, we can use the category together with the key.

ContextMemory stores context objects as long as they are necessary, because context-aware applications might need to access not only current but also previously collected contextual information. It is implemented as an ordered collection based on time and provides different search facilities.

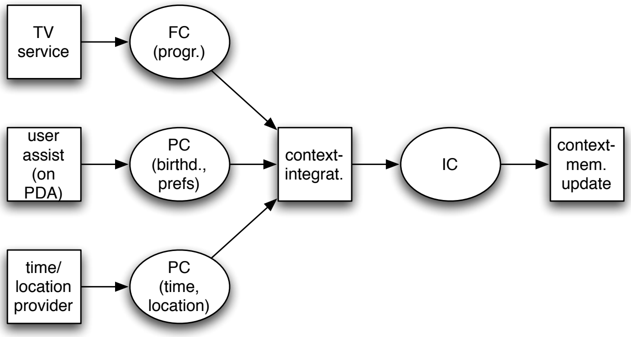


Fig. 2. Context memory updates

Figure 2 illustrates the processing of context in our application framework [12] for a simple example. Context objects (PC and FC) are produced from three sources: a TV service provides information about the currently shown program; the user assistant is a program running on users' PDAs that provides information about users to the environment according to privacy demands; thirdly, a local beacon issues the name of its location and a time stamp signal. Since sensor fusion mechanisms or handling of uncertainty are not necessary in this simple example, context integration can simply unite all PCs it receives between two time stamps into one integrated context object. Finally, the collected information is stored in context memory. Figure 1 shows part of context memory in two consecutive contexts: in context 1, two users are present, each disclosing information about their birthday; in context 2 10s later, the same data about users is broadcast, but additionally the TV service broadcasts that currently an educational program is shown.

If we consider the context objects to be simply collections of pieces of information, the additional information from the category of 5W1H is only an additional sorting criterion. The link into the ontology and thus to contextual reasoning is only established if it makes a semantic difference to which category a certain entry belongs. As an example, compare the *who*-element that contains the birthday of a user with the *when*-element containing the current time in Fig. 1. Above, we mentioned that the *who*-elements provide information about who are the agents of an interaction event, whereas *when*-elements specify the time of interaction. From a programming perspective, both context elements contain values of date/time data type. Ontologically however, the *birthday*-element must indicate users, and only the *time*-element specifies time. With set theoretical semantics we can realise this demand by stating that each *who*-element describes a set of users – in the example: the set of users whose birthday is on the given date – whereas the *when*-element describes a portion of time, i.e. a set of time points. Given such a mapping of the data structures to standard set-theoretical semantics, we can encode our ontology in a logical language in a straight forward manner.

3 Reasoning about Context Objects

Ontologies support three major issues in the development of context-aware ubiquitous computing applications [18]: discovery and matchmaking, interoperability, and context-awareness. Ontology languages based on description logics (DL), such as OWL (used, e.g., by [22, 9]), particularly support formulation of taxonomic knowledge, which is required for the first two tasks. Reasoning about the classical domains of context, in particular space and time however, requires additional expressive power. Other approaches to ontology-based context modelling use first order logic [18, 9], or F-Logic [22] for additional expressiveness. Ranganathan et. al [18] believe this need for more expressiveness to be caused by space and time being quantitative domains. However, research in qualitative reasoning has shown that space and time can be reasoned about efficiently with qualitative representations [19], and that users are more comfortable with qualitative than with quantitative interfaces to, e.g., spatial knowledge [7].

Qualitative spatial, temporal, and taxonomic knowledge has been handled previously in separate specialised logical languages, and combined languages are only recently being explored [13]. To see that it is not trivial to add, e.g., spatial relations into taxonomic knowledge consider the integration of the RCC-relations [17] in SOUPA [5]: the formal specification[1] expresses, for instance, only that *proper part* and *part* are transitive relations, but not that *part* is a reflexive relation and *proper part* is irreflexive; or that any proper part of a region is also a part of that region. Moreover, the SOUPA specification of space[2] adds a relation *spatiallySubsumes*, which is supposed to provide spatial containment reasoning [5, Sect. 3.1.5]. However, its relation to the RCC-relations, or whether it is reflexive or irreflexive is not specified.

Following a similar approach as [13], we characterise a specialised logical language that combines reasoning about the specific domains. Our main idea for translating knowledge stored in the context model into a format suitable for logical reasoning is to use *context terms* corresponding to the *context objects* from the context modelling side as the atomic units for the logical language. Each context object and therefore also each context term is understood as describing

[1] http://pervasive.semanticweb.org/ont/2004/06/rcc

[2] http://pervasive.semanticweb.org/ont/2004/06/space

circumstances of a certain *interaction* by a proposition stating that someone (*who*) interacts somehow (*how*) and for a certain reason (*why*) with something (*what*) at a given time (*when*) and place (*where*). To simplify the discussion, we only present the logical framework for reasoning about four parameters of an interaction, namely the spatial, temporal, object-related taxonomic, and agent-related taxonomic parameters.

The syntax of the logical language is similar to that of a description logic. In description logics we have two types of expressions: concepts and formulae, where only concepts are recursively defined. Similarly, the logical language defined in this paper consists of the recursively defined context terms, denoting circumstances of an interaction, and formulae. The set of context terms is defined based on a set of atomic context terms as the smallest set that fulfils:

1. All atomic context terms and the special symbols $\top$ (the maximal or trivial context), and $\bot$ (for the empty or impossible context) are context terms.
2. If c and d are context terms then the complement $\neg c$, the sum $(c \sqcup d)$, and the intersection $(c \sqcap d)$ are also context terms.

In comparison to concepts in description logics, we currently do not allow constructions involving quantification. Instead, we encode some of the necessary functionality into different operators for generating formulae out of context terms. A *context formula* is formed from two context terms with one of five operators: if c and d are context terms, then $c \sqsubseteq d, c \sqsubseteq_{\text{who}} d, c \sqsubseteq_{\text{what}} d, c \sqsubseteq_{\text{when}} d, c \sqsubseteq_{\text{where}} d$ are context formulae to be read as summarised in Tab. 2 below. A *contextual knowledge base* (CKB) is defined as a set of context formulae.

We can now specify the semantics of this simple language. For representing the four aspects, the domain of discourse consists of quadruples of subsets of four distinct sub-domains: $U = 2^{U_{\text{A}}} \times 2^{U_{\text{O}}} \times 2^{U_{\text{T}}} \times 2^{U_{\text{S}}}$, where U_{A} is the set of all users, U_{O} is the set of all objects, U_{T} is the set of all temporal points, and U_{S} is the set of all spatial points, i.e. the area of the domain. Any context term c is interpreted as a quadruple $\langle a, o, t, r\rangle \in U$ corresponding to a sentence: "some members of the group of *agents* a interact with some *objects* in o at a *time* in t somewhere in the *region* r." The interpretation function I maps the context terms to elements of U. The special symbols $\top$ and $\bot$ and the context term operators are interpreted in the following way:

$$I(\top) = \langle U_{\text{A}}, U_{\text{O}}, U_{\text{T}}, U_{\text{S}}\rangle \text{ and } I(\bot) = \langle \emptyset, \emptyset, \emptyset, \emptyset\rangle,$$

$I(c \sqcap d) = \text{and}(I(c), I(d))$, with *and* being the piecewise intersection:

$$\text{and}(\langle a_1, o_1, t_1, s_1\rangle, \langle a_2, o_2, t_2, s_2\rangle) = \langle a_1 \cap a_2, o_1 \cap o_2, t_1 \cap t_2, s_1 \cap s_2\rangle$$

$I(c \sqcup d) = \text{or}(I(c), I(d))$, with *or* being the piecewise union:

$$\text{or}(\langle a_1, o_1, t_1, s_1\rangle, \langle a_2, o_2, t_2, s_2\rangle) = \langle a_1 \cup a_2, o_1 \cup o_2, t_1 \cup t_2, s_1 \cup s_2\rangle$$

$I(\neg c) = \text{comp}(I(c))$, where *comp* is the piecewise complement:

$$\text{comp}(\langle a, o, t, s\rangle) = \langle (U_{\text{A}} \setminus a), (U_{\text{O}} \setminus o), (U_{\text{T}} \setminus t), (U_{\text{S}} \setminus r), \rangle$$

The basic relation underlying the semantics of the operators is the relation of *containment* ($\subset$) as shown in Tab. 2. With the interpretation function I and the

Table 2. Syntax and semantics of operators. The semantics is given with respect to two context terms c and d with $I(c) = \langle a_c, o_c, t_c, r_c \rangle$ and $I(d) = \langle a_d, o_d, t_d, r_d \rangle$.

Syntax	Semantics	Reading
$c \sqsubseteq_{\text{who}} d$	is true, iff $a_c \subset a_d$	c is socially a sub-context of d
$c \sqsubseteq_{\text{what}} d$	is true, iff $o_c \subset o_d$	c is conceptually a sub-context of d
$c \sqsubseteq_{\text{when}} d$	is true, iff $t_c \subset t_d$	c is temporally a sub-context of d
$c \sqsubseteq_{\text{where}} d$	is true, iff $r_c \subset r_d$	c is spatially a sub-context of d
$c \sqsubseteq d$	is true, iff $a_c \subset a_d$, $o_c \subset o_d$, $t_c \subset t_d$, and $r_c \subset r_d$	c is a sub-context of d

domain U defined, entailment from CKBs can be derived in the standard way. A structure $\langle I, U \rangle$ is a model for a formula ϕ, iff ϕ is true in $\langle I, U \rangle$. We obtain five variants for the semantic concept of satisfiability: we call a context term c *spatially (temporally, conceptually, socially) satisfiable* iff a structure $\langle I, U \rangle$ exists in which $c \sqsubseteq_{\text{where}} \perp$ ($c \sqsubseteq_{\text{when}} \perp$, $c \sqsubseteq_{\text{what}} \perp$, $c \sqsubseteq_{\text{who}} \perp$) does not hold; c is *strongly satisfiable* iff it is spatially, temporally, conceptually, and socially satisfiable.

We suggest a simple tableau mechanism (cf. [8] for an introduction and overview) for reasoning with CKBs. In order to ask a question, such as whether $c \sqsubseteq_{\text{where}} d$ holds in a CKB, we ask whether the context term $(c \sqcap \neg d)$ is spatially unsatisfiable. More exactly, we ask whether $q \sqsubseteq_{\text{where}} (c \sqcap \neg d)$ entails $q \sqsubseteq_{\text{where}} \perp$, for an arbitrary new context term q (for *query*). We can then start the tableau algorithm with the CKB and a set of query formulae as given in Tab. 3.

Table 3. Examples for questions to the CKB. The query context term q does not appear anywhere else in the CKB. The formula $c \sqsubseteq d$ is expanded to $\{c \sqsubseteq_{\text{where}} d, c \sqsubseteq_{\text{when}} d, c \sqsubseteq_{\text{what}} d, c \sqsubseteq_{\text{who}} d\}$. Note that the semantic concept of satisfiability of a context term cannot be expressed within the logical language itself, since negation of a formula cannot be expressed.

Question	Formula	Negated Query Set Q
Is a context as described by the term c satisfiable?	no positive form	$\{q \sqsubseteq c\}$ expanded: $\{q \sqsubseteq_{\text{where}} c, q \sqsubseteq_{\text{when}} c, q \sqsubseteq_{\text{what}} c, q \sqsubseteq_{\text{who}} c\}$
Do the interactions of c take place in the region of d?	$c \sqsubseteq_{\text{where}} d$	$\{q \sqsubseteq_{\text{where}} (c \sqcap \neg d)\}$
Do the interactions of c involve objects of d?	$c \sqsubseteq_{\text{what}} d$	$\{q \sqsubseteq_{\text{what}} (c \sqcap \neg d)\}$

The algorithm starts with the set $T = \{CKB \cup Q\}$ containing as the only branch $CKB \cup Q$. In every step, we expand a branch $S \in T$: we first check whether S is a closed branch, that is, whether it contains for some operator $\sqsubseteq_m$, either $q \sqsubseteq_m \bot$, or both $q \sqsubseteq_m c$ and $q \sqsubseteq_m \neg c$ for some context term c. If S is closed, it is removed from T. If $T = \emptyset$, the tableau is closed, and the query has been proved. As long as there is still an open branch $S \in T$, we select a formula ϕ from S and modify S according to the rules given in Tab. 4. In case of the β-rules, we replace S with two branches. If a branch in T cannot be closed and no rule is applicable, the query has been disproved.

Table 4. Simple rules for basic contextual reasoning. For each rule $\sqsubseteq_m$ signifies one of $\sqsubseteq_{\text{who}}, \sqsubseteq_{\text{where}}, \sqsubseteq_{\text{when}}, \sqsubseteq_{\text{what}}$, so that we obtain a total of $4 * 10 = 40$ rules.

Name	Input (ϕ in branch S)	Output ($S' = S \setminus \{\phi\}$)
q-intro	$c \sqsubseteq_m d$ (with $c \neq q$)	$S' \cup \{q \sqsubseteq_m (\neg c \sqcup d)\}$
$\bot$-elim	$q \sqsubseteq_m c \sqcup \bot$	$S' \cup \{q \sqsubseteq_m c\}$
$\bot$-abs	$q \sqsubseteq_m c \sqcap \bot$	$S' \cup \{q \sqsubseteq_m \bot\}$
$\top$-elim	$q \sqsubseteq_m c \sqcap \top$	$S' \cup \{q \sqsubseteq_m c\}$
$\top$-abs	$q \sqsubseteq_m c \sqcup \top$	$S' \cup \{q \sqsubseteq_m \top\}$
$\neg$-elim	$q \sqsubseteq_m \neg\neg c$	$S' \cup \{q \sqsubseteq_m c\}$
α-rule 1	$q \sqsubseteq_m (c \sqcap d)$	$S' \cup \{q \sqsubseteq_m c, q \sqsubseteq_m d\}$
α-rule 2	$q \sqsubseteq_m \neg(c \sqcup d)$	$S' \cup \{q \sqsubseteq_m \neg c, q \sqsubseteq_m \neg d\}$
β-rule 1	$q \sqsubseteq_m (c \sqcup d)$	two branches $S' \cup \{q \sqsubseteq_m c\}$ and $S' \cup \{q \sqsubseteq_m d\}$
β-rule 2	$q \sqsubseteq_m \neg(c \sqcap d)$	two branches $S' \cup \{q \sqsubseteq_m \neg c\}$ and $S' \cup \{q \sqsubseteq_m \neg d\}$

4 Discussion

With both the context model and the context reasoning mechanism described, we can sketch how the two components can be linked. The proposed context model provides us with the necessary data model to store and process sensory data in a common format, in our approach the context elements. All data collected about the same situation together yield the context object. We therefore can state that the context object represents the situation as it is *perceived* by the system. Contextual reasoning comes in, as these perceptions are assigned a *meaning* by sorting them into the conceptual scheme provided by application ontologies: we might, for instance, classify a certain range of sensory values from a physiological sensor as signalling a critical health condition. With this classification we link the perception into our representation of the world. When the percept is classified, it becomes a new fact in the CKB.

Thus, we can compare the context model with a representation of perceptions, whereas contextual reasoning handles representations of knowledge. The gap between context modelling and contextual reasoning can then be identified as the well-known grounding problem of how knowledge is anchored in perception [10].

Context modelling, designed to support the generation of more and more abstract classifications – from *preliminary context* retrieved from sensors to *integrated context* required for triggering services –, can be understood as an effort to bridge this gap from the sensory or perceptual side.

Application ontologies can be encoded in our framework as consistent sets of sentences, i.e., as axiomatic systems formulated in the logical language. They form the static basis of the CKB in a running context-aware system. Additionally, the context-aware system can dynamically expand the CKB with information from the context objects. Every newly constructed context object, the `currentContext` object, can be translated into a context term that can be sorted into the (taxonomic, spatial, and temporal) hierarchies encoded in the CKB, in order to be available for reasoning. Classification of context terms requires two steps, first, *algorithmic classification*, a step of abstraction in context processing, and second, *ontology-based classification*, using the reasoning mechanism. Consider, for instance, a context-aware media centre in a smart home environment, such as the ubiTV application [16].

Algorithmic classification In the first step, an application developer has to provide methods to convert collections of context elements into context terms, thus assigning a meaning to them with respect to the semantics of the logical language. What does it mean, for instance, if there are two context elements with the *birthday*-key and different values in a context, as in Fig. 1 (p. 4)? Since the elements are in the *who*-domain they have to be interpreted as denoting groups of users. An interprperspicuousetation of this context object that makes sense is to assume that the group of users in this context is contained in the set of all persons whose birthday is on one of the two dates, i.e., as the union of the groups of users described by each context element. The result of algorithmic classification is a description of the current context in terms of the application ontology, in the example, a classification of users according to age and of TV-programs according to content might be relevant for the TV-application:

$$\begin{aligned}\texttt{currContext} &\sqsubseteq_{\text{who}} \text{Teenager}\\ \texttt{currContext} &\sqsubseteq_{\text{what}} \text{EducationalTVProgram}\end{aligned}$$

Ontology-based classification After algorithmic classification has been used to generate a description of the situation in terms of the application ontology, we can further reason about the situation within the logical framework. With the following statements from an application ontology, for instance,

$$\begin{aligned}\text{EducationalTVProgram} &\sqsubseteq_{\text{what}} \text{TVProgram},\\ \text{TVProgram} &\sqsubseteq_{\text{what}} \text{NeedsSoundResource} \sqcap \text{NeedsPictureResource}\end{aligned}$$

the media centre can conclude that the current context requires sound resources

$$\texttt{currContext} \sqsubseteq_{\text{what}} \text{NeedsSoundResource}.$$

5 Conclusion and Future work

We discussed a novel perspective on ontology-based context modelling that makes it easy to combine context models and contextual reasoning mechanisms. On the context modelling side, we outlined our idea of user-centric context modelling; on the contextual reasoning side, we introduced syntax and semantics for a simple logical language. The expressiveness of this first, simple language does not go beyond propositional logics and further extensions are a focus of ongoing works. However, even this simple language already supports reasoning about main parameters of context (who, when, where, what) in a unifying and perspicuous way, and is sufficient to encode, e.g., the location hierarchies proposed by Leonhardt [14].

From the context modelling perspective, this paper illustrates how a logical query language with a clear semantics can be used to provide contextual reasoning capabilities as well as model-theoretic semantics to a context model. From the contextual reasoning side, the context model provides a way for grounding knowledge-based reasoning about context in sensory or perceptual data about the real world.

The paper provided only an overview of our approach. Details, e.g., regarding how context integration is performed on the context modelling side, or the proof of formal properties of the logical language have not been shown. However, one of the key benefits of our approach to ontology-based context modelling is that questions regarding completeness and soundness of the logical framework can be asked and answered much easier and clearer than in the conventional approach to ontology design based on DL/OWL; and even more so, since several prominent approaches to ontology-based context modelling are formulated using a combination of DL with more expressive logical languages. We conclude that context-aware applications can benefit from specific context logics, and a focus of ongoing works is the detailed investigation of more expressive logical languages for specifying context ontologies. Among the desiderata are, particularly, extensions for representing linear orders, such as the temporal *before*, in order to move towards a fully-fledged contextual reasoning mechanism [2].

References

1. J. E. Bardram. The Java context awareness framework (JCAF) - a service infrastructure and programming framework for context-aware applications. In H.-W. Gellersen, R. Want, and A. Schmidt, editors, *Pervasive Computing, Third International Conference*, pages 98–115, 2005.
2. M. Benerecetti, P. Bouquet, and C. Ghidini. Contextual reasoning distilled. *Journal of Experimental and Theoretical Artificial Intelligence*, 12(3):279–305, 2000.
3. P. Brézillon. Context in problem solving: A survey. *The Knowledge Engineering Review*, 14(1):1–34, 1999.
4. G. Chen and D. Kotz. A survey of context-aware mobile computing research. Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College, November 2000.

5. H. Chen, F. Perich, T. Finin, and A. Joshi. SOUPA: Standard ontology for ubiquitous and pervasive applications. In *First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services*, 2004.
6. A. K. Dey and G. D. Abowd. Towards a better understanding of context and context-awareness. In *Workshop on The What, Who, Where, When, and How of Context-Awareness*, 2000.
7. M. J. Egenhofer. Spatial SQL: A query and presentation language. *IEEE Transactions on Knowledge and Data Engineering*, 6(1):86–95, 1994.
8. M. Fitting. *First-Order Logic and Automated Theorem Proving*. Springer, 1996.
9. T. Gu, H. K. Pung, and D. Q. Zhang. A service-oriented middleware for building context-aware services. *Journal of Network and Computer Applications*, 28(1):1–18, 2005.
10. S. Harnad. The symbol grounding problem. In *Encyclopedia of Cognitive Science*. Macmillan and Nature Publishing Group, 2003.
11. K. Henricksen and J. Indulska. Developing context-aware pervasive computing applications: Models and approach. *Pervasive and Mobile Computing*, 2:37–64, 2006.
12. D. Hong, Y. Suh, A. Choi, U. Rashid, and W. Woo. wear-UCAM: A toolkit for mobile user interactions in smart environments. In *Embedded and Ubiquitous Computing*, pages 1047–1057. Springer, 2006.
13. O. Kutz, C. Lutz, F. Wolter, and M. Zakharyaschev. $\mathcal{E}$-connections of abstract description systems. *Artificial Intelligence*, 156(1):1–73, 2004.
14. U. Leonhardt. *Supporting Location Awareness in Open Distributed Systems*. PhD thesis, Imperial College, London, UK, 1998.
15. G. Ligozat. Reasoning about cardinal directions. *Journal of Visual Languages and Computing*, 9:23–44, 1998.
16. Y. Oh, C. Shin, S. Jang, and W. Woo. ubi-UCAM 2.0: A unified context-aware application model for ubiquitous computing environments. In *UbiCNS*, 2005.
17. D. Randell, Z. Cui, and A. Cohn. A spatial logic based on region and connection. In *Third International Conference on Knowledge Representation and Reasoning*, pages 165–176. Morgan Kaufmann, 1992.
18. A. Ranganathan, R. E. McGrath, R. H. Campbell, and M. D. Mickunas. Ontologies in a pervasive computing environment. In *Workshop on Ontologies and Distributed Systems (part of IJCAI)*, 2003.
19. J. Renz. Qualitative spatial and temporal reasoning: Efficient algorithms for everyone. In *Twentieth International Joint Conference on Artificial Intelligence*, pages 526–531, 2007.
20. B. N. Schilit, N. I. Adams, and R. Want. Context-aware computing applications. In *Workshop on Mobile Computing Systems and Applications*, pages 85–90. IEEE Computer Society, 1994.
21. A. Schmidt, M. Beigl, and H.-W. Gellersen. There is more to context than location. *Computers and Graphics*, 23(6):893–901, 1999.
22. T. Strang, C. Linnhoff-Popien, and K. Frank. CoOL: A context ontology language to enable contextual interoperability. In J.-B. Stefani, I. M. Demeure, and D. Hagimont, editors, *4th International Conference on Distributed Applications and Interoperable Systems*, pages 236–247. Springer, 2003.

Towards a Generic Contextual Elements Model to Support Context Management

Vaninha Vieira[1, 2], Patrick Brézillon[2], Ana Carolina Salgado[1] and Patrícia Tedesco[1]

[1] Center of Informatics, Federal University of Pernambuco, PO Box 7851, Recife, PE, Brasil
[vvs,acs,pcart]@cin.ufpe.br
[2] LIP6, University of Paris 6, 104 Avenue du Président Kennedy, 75016 Paris, France
[vieira,brezil]@poleia.lip6.fr

Abstract. Context management solutions propose the separation of context manipulation tasks from the application's business features. An important concern when managing contextual elements is how to structure and represent them. Context models in literature, generally, propose the formalization of specific contextual elements related to a domain and/or application. However, to be effective and reused by different systems, a context manager should abstract away domain or application particularities. In this paper, we propose COM (*Context-Oriented Model*), an approach to deal with the context modelling problem through the separation of generic context concepts from domain/application particular concepts. COM is a top-down modelling approach that can be used as the base context model to generic context managers. We believe that this approach can also serve to support developers of context sensitive systems in modelling the context usage into their applications.

1 Introduction

Context is what underlies the characterization of entities enabling to understand and differentiate situations. The term *Contextual Element* (CE) refers to pieces of data, information or knowledge that can be used to define the context [1]. Context management aims at providing solutions, such as models and services, to assist the acquisition, representation, processing, maintenance and manipulation of CEs, separating context-related tasks from the applications' business [1]. Generic context managers are of interest since they enable reuse and reduce the complexity associated in building context-sensitive systems, which are more complex than traditional systems exactly because of the additional context management needs.

An open topic in context management is to identify approaches and techniques to support the specification and representation of the contextual elements. Generic models are of interest since many different systems may benefit from them. However, existing proposals of context models, generally, are quite simplistic since they pre-define specific CEs related to concepts such as Person, Time, Location, Device and Activity [2;3], which can be, usually, identified by acquisition interfaces like sensors. We believe that the development of a generic context model should not start by specifying *a priori* which

CEs should be considered in a static and strict manner, since context is very dependent on the domain and application.

In this light, this paper proposes COM (*Context-Oriented Model*), a modelling approach for CE representation, based on the need to specify a generic context model for our domain-independent and reusable context manager, the CEManTIKA (*CE Management Through Incremental Knowledge Acquisition*) [1;4]. It separates the context modelling task in three steps: (i) formalize in a high level layer the context related concepts that will orient the context management; (ii) identify, for a given domain, the contextual elements that should be considered; and (iii) instantiate the contextual elements values according to an application usage.

The idea is to define generic concepts in the same way, for example, that object-oriented systems or aspect-oriented systems do. Generic concepts define the way the system should be modeled and thus allow the development of a mechanism that understands the systems that were built using that paradigm. Our approach proposes to bring this idea to the development of context-sensitive systems. We think that to achieve the generality and reuse of context-related solutions it is necessary to change the way developers think about their systems by explicitly identifying the context related functionalities. This model is our first step in this direction.

The rest of the paper is organized as follows: Section 2 presents our definition of context and a discussion about context modelling and related works; Section 3 introduces a motivating scenario to illustrate the model instantiation; Section 4 details the COM model and the context-related concepts of the high level layer; Section 5 exemplifies its use through the instantiation of the scenario illustrated in Section 3; finally, Section 6 points out some final considerations and further work.

2 Background

2.1 Our Definition of Context

Our work is based on two classical definitions of context. The first states that context is any information that can be used to characterize the situation of an entity (e.g. person, place, object, application) [5]. The second indicates that context is always related to a focus and that at a given focus the context is the aggregation of three types of knowledge: Contextual Knowledge (CK), External Knowledge (EK) and Proceduralized Context (PC) [6]. Focus means an objective or a step in a task, a problem solving, or a decision making. The focus enables to separate knowledge that is relevant or that is not relevant to determine the context. The CK is the known relevant part, while the EK is the unknown or irrelevant part in the context. The PC is the knowledge effectively used in the focus to support the task at hand; it is composed of a subset of the CK that is assembled, organized and instantiated to address the focus, along with the rationale that was used to achieve the instantiated CK.

We use the term *Contextual Element* (CE) to refer to pieces of data, information or knowledge that can be used to define the context [1]. This separation is necessary because contextual knowledge comprises what is in the users' minds and thus is too abstract. To treat context computationally it is important to make this distinction between contextual data (e.g. location coordinates, identification and temperature), contextual information (e.g. weather is hot, nearby person) and contextual knowledge (e.g. understanding about the weather behavior in different regions).

2.2 Context Modelling and Related Works

An important issue in context-sensitive systems is the identification of the CEs that must be considered and how they should be represented to ease the acquisition and reasoning. With the advance of context-aware computing, there is an increasing need for developing formal and generic context models to facilitate context representation, sharing and semantic interoperability of heterogeneous systems [7]. A generic context model should take into account aspects such as the distinction between the context model and the application domain and ways to maintain a sharable context model that enables the communication between different systems.

There are several attempts to define and use context for computational purposes and different approaches are being experimented, such as key-value pairs [8], markup schemas [9], object-oriented models [10], ontologies [11] and topic maps [12]. Ontologies appear as a promising approach since they enable knowledge sharing between human and software agents, easy knowledge reuse between systems, and can be easily used by inference engines for reasoning. Topic maps also seem interesting because they can be used to organize large sets of information building a structured semantic link network over existing resources [12]. This network allows easy and selective navigation to the requested information. An interesting characteristic of topic maps is that topics can have relationships (associations) with each other and topics can play different roles in different associations. In our work we are using a combination of ontologies and topic maps to represent the CEs to achieve better results in terms of CE definition and navigation.

We observed that existing modelling proposals are based on pre-defined and limited sets of contextual information, in general related to a specific domain, or that can be acquired through sensors. They establish a static and pre-defined set of entities and model the context directly as static attributes associated to these entities. We believe that it is impractical to imagine a generic and reusable context model if we start by establishing and limiting the set of elements that is going to be modeled. Context is a dynamic and complex concept and the CEs are strongly dependent on the domain and application. So, it is not reasonable to imagine that a single system analyst or even worse the analyst of a generic context manager is able to specify and define *a priori* all CEs that are relevant to an application that not even exists.

The major difference of our proposal is that we do not intend to establish a fixed structure for the CEs related to a specific problem/application. We propose a meta-model through what the CEs will be modeled during the context-sensitive system analysis and development. Another approach for context representation, more similar to ours, was proposed by Bucur et al. [13]. They combine the generality of ontologies with the complexity inspired by object oriented models, modelling two ontologies: a domain ontology and another that is the description of the context attributes managed by the system. The difference between our proposal and theirs is that we based our model in a definition well known and accepted in the context literature, which gives more theoretical foundation to support our modeled concepts. Also, we consider in the model the dynamicity of the context manipulation through the focus changing.

3 Motivating Scenario

To motivate and simplify the explanation of the model concepts we will consider the following scenario, that we will have in mind to exemplify the model concepts and its instantiation in a Mission Planning Support System.

Patricia is a researcher at a University located in the city of Recife in Brazil. Recently, she had a paper accepted to the *Context-07*, a conference that will take place at Roskilde University, in the city of Roskilde, Denmark. Roskilde is very near to Copenhagen, which for travelling matters is a city of reference indicated by the event organizers. The event will be held in the period of 20 to 24 august 2007. After receiving the good news about the paper, Patricia has to prepare her mission, taking care of things like the transport and hotel reservation. Another important point is that she has a PhD student under her direction doing a stage in a laboratory in Paris. So, to Patricia it is important to accommodate all these requirements (go to the conference, attend to interesting workshops and meet her student at Paris) to choose the best options for travelling and accommodation. Since it is her first time going to Denmark and she knows absolutely nothing about Roskilde, it will be really interesting to Patricia to have a system that supports her while preparing her mission by suggesting itineraries, hotels and travelling companies that are adequate to her needs. The system developer must identify the involved CEs and the CEs that are relevant to the different phases (focus) of a mission preparation.

4 The Context-Oriented Model (COM)

The COM model is divided in three layers (Fig. 1): the *upper layer*, which characterizes the generic context management related concepts and can be qualified as a "meta-model" since it is used for creating individual models; the *middle layer* that defines the domain-specific context-related concepts in accordance with the upper layer; and the *lower layer*, which represents the instantiation of the domain concepts according to a specific application, incrementally acquired during the system usage.

The idea behind the separation into these three layers is that if we define the context specific concepts in a high level domain-independent layer, than these concepts can be managed in a generic manner, without worrying about the domain particularities. So, the mechanics of the context manager will be applied over the upper layer concepts. A compatible context-sensitive system must model the application concepts as instantiation of the upper layer concepts. Existing context and domain ontologies, for example, can be used in the middle layer.

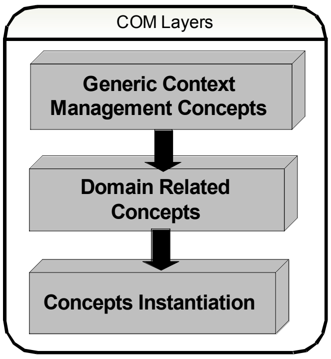


Fig. 1. Interaction between the three layers in the COM model

This paper focuses in discussing the upper layer concepts, which are detailed in the next subsections. The other layers will be discussed in Section 5 through the instantiation of the model according to the motivating scenario described in Section 3.

4.1 Concepts Specification

An illustration of the generic context management concepts and their properties and relationships are presented in Fig. 2. The model is centered on five main concepts: *Entity*, *Contextual Element*, *Focus*, *Rule* and *Action*; and three derived concepts: *CEF-Set*, *RF-Set* and *Proceduralized Context*. The main concepts must be specialized by the application/domain analyst while the derived concepts are built by the context manager based on the main concepts and guided by the focus.

4.1.1 Entity

An entity is anything in the real world that is relevant to describe the domain. For example, in the scenario in Section 3, some entities are Person, Hotel, Mission, Location, TransportType, FlightCompany, TrainCompany and Event. The entities definition can be imported from existing sources such as a domain ontology. An Entity is defined through: a *name*; a *type* (e.g. an entity Missionary is of type Person, an entity Country is of type Location); and an *URI* that permit to link the entity to an external resource that contain further knowledge, such as its OWL description file. The Entity has an association *isCharacterizedBy* with the concept Contextual Element, meaning that one Entity is characterized by one or more CEs.

4.1.2 Contextual Element (CE)

A Contextual Element, as defined before, may represent a contextual data, information or knowledge, and is used to characterize entities. Examples of CEs for the entity *Mission* include: *location*, *initialDate*, *endDate, duration, whoPays* and *officialReasons*; and for the entity *Hotel* include: *location*, *numberOfStars*, *category*, *minimunPrice* and *numberOfRoomsAvailable*.

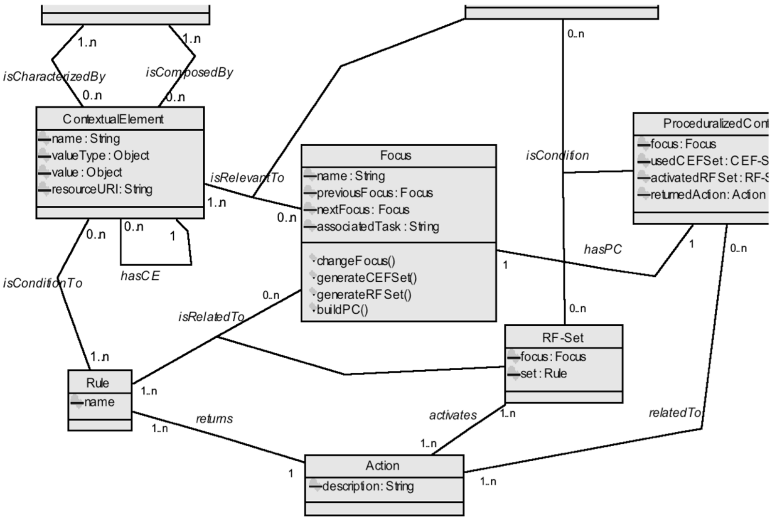


Fig. 2. Specification of the generic context management concepts using UML Notation

The attributes that identify a CE are *name*, *valueType*, *value* and *resourceURI*. The *valueType* indicates the expected value for the CE. For example, the CE *location* has as valueType an entity Location, the CE *numberOfStars* expects as value type an object of type Integer, while the valueType for the CE *category* is an entity Category, which contains the instances {economic, executive, first class}. The attribute *value* signifies the ascribed instance of the CE and will be filled by the context-sensitive application during its usage. The attribute *resourceURI* identifies a link to an external resource that describes the CE, such as an OWL or RDF file.

A CE may be associated to one or more CEs, creating a hierarchical structure between the contextual elements. For example, the CE *officialReasons* may have as possible values the set {*presentation*, *experimentation*, *meeting*}, meaning that the official reasons for a mission may be one of these. In its turn, *presentation* is itself a CE that has as value types the subset {*seminar*, *conferencePaper*}. This relation between CEs reinforces the hierarchy and dependency.

The CE concept has two associations with the Entity concept. The first points out that an entity is characterized by one or more CEs, and the second implies that a CE may be composed of one or more entities. The latter is important in situations where a CE associated to an Entity X needs to make a reference to an Entity Y. For example, the CE *distance* associated to entity Hotel needs to make a reference to another entity, such as Mission, indicating the distance between a Hotel and a Mission location.

4.1.3 Rule and Action

The Rule concept was included in the model to explicitly represent the rules associated to the CEs, necessary to produce contextual information from contextual data and also to support the building of the Proceduralized Context in the focus. The Rule is identified by a *name*, it has one or more conditions, which are represented by the association *isConditionTo* with the Contextual Element concept, and it has one returning action, represented by the association *returns* with the Action concept.

The Action is represented by a *description* and exists in function of a rule. We modeled the action separately from the rule to easy the modelling the context dependent actions that could be implemented by the systems. In this light, the rules are specified having in mind the possible and desired actions.

4.1.4 Focus

The focus is a central and important concept in our model, since the context is always related to a focus. We define the focus as an objective to be achieved, such as a task in a problem solving or a step in a decision making. It is used to identify clear points of time and space that the context is all about. The focus allows the context manager to determine what CEs should be used and instantiated, since it determines the relevancy of a CE in a specific situation. In the scenario of Section 3, examples of focus are *Define the mission*, *Book Hotel*, *Book Transport* and *Make Payment*.

The focus is identified by a *name*. Since it is related to objectives to solve a problem or to execute a task, we modeled it as a sequence of foci, where each element has references to the *previousFocus* and the *nextFocus*. Another attribute is the *associatedTask* that may be a problem, a decision making or a task, and can contain a textual description or an URI referencing an external resource that describes the task.

The associations for the concept focus are: *isRelevantTo* with the concept Contextual Element, showing that the CE is relevant to that focus. Similarly the concept Rule has an association *isRelatedTo* showing that the rule is related to the focus. The *hasPC* indicates that a focus has a related Proceduralized Context.

Context is a dynamic construction that evolves with the focus. As the focus changes the set of CEs that must be considered change accordingly [6]. In this way the Focus concept is the one who guides the generation of the derived concepts (CEF-Set, RF-Set and Proceduralized Context). This is done by the methods associated to the Focus: *changeFocus*, shows that the focus changed pointing out that it is necessary to review the current context; *generateCEFSet*, which marks the building of the CEF-Set according to the current focus and its associated CEs; *generateRFSet*, to build the RF-Set according to the relevant Rules for the focus; and *buildPC*, which indicates the procedures to construct the PC for the focus.

4.1.5 CEF-Set

To identify and manipulate the CEF-Set (*Contextual Elements in the Focus Set*) we use the principles of the mathematical theory of sets. A set is a collection of abstract objects. So, a CEF-Set is a collection of CEs that are relevant and must be considered in a Focus. It is dynamically generated and will be continually rebuild when a new focus arrives. So, as stated in the definition below (*CEF-Set (f)*) an CEF-Set related to a focus f is comprised by a set of tuples (c, v) such that c is a Contextual Element, v is a value for c, being another Contextual Element or a literal (e.g. string, integer or date), and c has an association of relevancy with the focus f.

```
CEF-Set (f) = {(c, v) | isContextualElement(c) AND
isValueOf(v, c) AND (isContextualElement(v) OR isLiteral(v))
AND isRelevant (c, f) }
```

For example, in the scenario of Section 3, the entity Hotel is characterized by CEs such as: *location*, *architecturalStyle*, *yearOfFoundation*, *executiveManagerName*, *numberOfStars* and others. In the focus *Book Hotel* the CEF-Set will contain only the CEs *location* and *numberOfStars*, since the other CEs are totally irrelevant for the problem of recommending hotels for Patricia's mission.

4.1.6 RF-Set

The RF-Set (*Rules in the Focus Set*) has a similar functionality as the CEF-Set, but it determines the relevant *rules* that should be activated in the focus. The Rule concept models all rules in the system related to the defined CEs. However, in a focus only a subset of all rules must be, in fact, activated. Thus, the definition *RF-Set (f)*, below, indicates that an RF-Set in a focus *f* is formed by a set of tuples (*r, a*) such that *r* is a Rule, *a* is an Action, *a* is a result of the execution of the rule *r*, and *r* has an association is related to the focus *f*.

```
RF-Set (f) = {(r,a) | isRule(r) AND isAction(a) AND
isResultOf(a, r) AND isRelatedTo(r, f)}
```

4.1.7 Proceduralized Context (PC)

The PC contains the CE that should effectively be used to support the current focus, and the rationale that has enabled the context manager to identify these CEs. It is constructed based on the contextual elements in the CEF-Set and the selected set of inference rules (the RF-Set). The PC is composed of an explanation related to the processing of the instantiation of the inference rules with the elements in the CEF-Set as conditions and the returned actions.

```
PC (f) = {[(c, v),(r, a)] | isInCEFSet(f,(c,v)) AND
isInRFSet(f,(r,a)) AND isConditionOf(c,r) AND return(r,a)}
```

The definition *PC (f)*, above, states that the PC in a focus *f* is formed by a set of tuples [(*c, v*), (*r, a*)] such that (*c,v*) belongs to the CEF-Set, (*r, a*) exists in the RF-Set, *c* is a condition for *r*, and the execution of *r* with the tuple (*c*, *v*) return the action *a*.

4.1.8 Final Considerations about the COM Model

This section presented the general context related concepts defined for COM. We based our model in the same idea existent when we think about an object oriented model or a relational based model. These modelling techniques define the main structure that enables the building of a computer system that fits each paradigm. If we are going to develop an object oriented system our world must be represented according to classes, properties attributes and objects (instances). When modelling a relational system, the world should fit in the concepts of tables, rows and columns. So, for us, when thinking about modelling a context-based system we must think in terms of entities, contextual elements, focus, rules and actions. And the usage of context is done based on focus and proceduralized contexts.

An important concept in the set theory is the *Universal Set*, which means the set of all elements and subsets we are interested in. In our model it is what we mean by the domain being considered. Another concept of the set theory is the Complement of a Set, which contains all the remained elements in the Universal Set not considered in a set. The context model proposed by Brézillon and Pomerol [6] discussed in Section 2.1, in which this work is based on, includes the concept of External Knowledge (EK). In our model the EK in a focus may be achieved by applying the complement of a set over the CEF-Set. So, all elements that are a Contextual Element and that is not on the CEF-Set for a given focus is considered External Knowledge.

5 Example of Use – Instantiation of the Model

Fig. 3 shows an overview of the instantiation of the main concepts according to the scenario presented in Section 3, illustrating the interaction between the three layers presented in Fig. 1. In the example, three entities were defined: *Person*, *Misson* and *Hotel*. The entity Person has the CE *location*. The entity Mission has the CEs *location*, *duration* and *whoPays* (that uses the entity Person to indicate that the value for the whoPays for a mission will depend on who the missionary is). The entity Hotel has as CEs *location* and *distance* (who uses the entity Mission to indicate that the distance of the Hotel is relative to the location of the mission).

The foci *BookHotel* and *MakePayment* indicates different phases related to the mission preparation. The focus Book Hotel uses the CEs *location, duration* and *distance*. The CE *whoPays* is not relevant to BookHotel since the focus is related to finding hotels for a missionary taking into account elements such as: the distance between the hotel and the mission location and the mission duration (to verify the availability of rooms). The focus Make Payment indicates how the missionary should conduct the payment of the mission. For example, if *whoPays* for the mission is a travel agency that has a contract with the missionary department, then the payment should be done by a payment order emitted by the department secretary to the agency.

The main concepts are used to produce the derived concepts that will enable the context manager to build the PC related to a focus. The role of the derived concepts in the PC building is illustrated in Fig. 4 from the execution of selected rules (RF-Set) over relevant CEs (CEF-Set) according to the example shown in Fig. 3.

The CEF-Set in the focus BookHotel contains the instantiated CEs identified as relevant: *location*, *duration* and *distance*. The RF-Set contains the relevant rules for the focus: *EconomicHotelRule, IsClientRule, IsNearRule* and *HasRoomRule*. The execution of these rules may produce new CEs that will increase the original CEF-Set. For example, the rule *IsClientRule* instantiated a CE *isClient* for the entity-instance *Person:Vaninha* related to the entity-instance *Hotel:Ibsens*. For the sake of clarity we separate in the CEF-Set illustrated in Fig. 4 the previously known CE (upper part of the CEF-Set) from the new inferred CE (lower part).

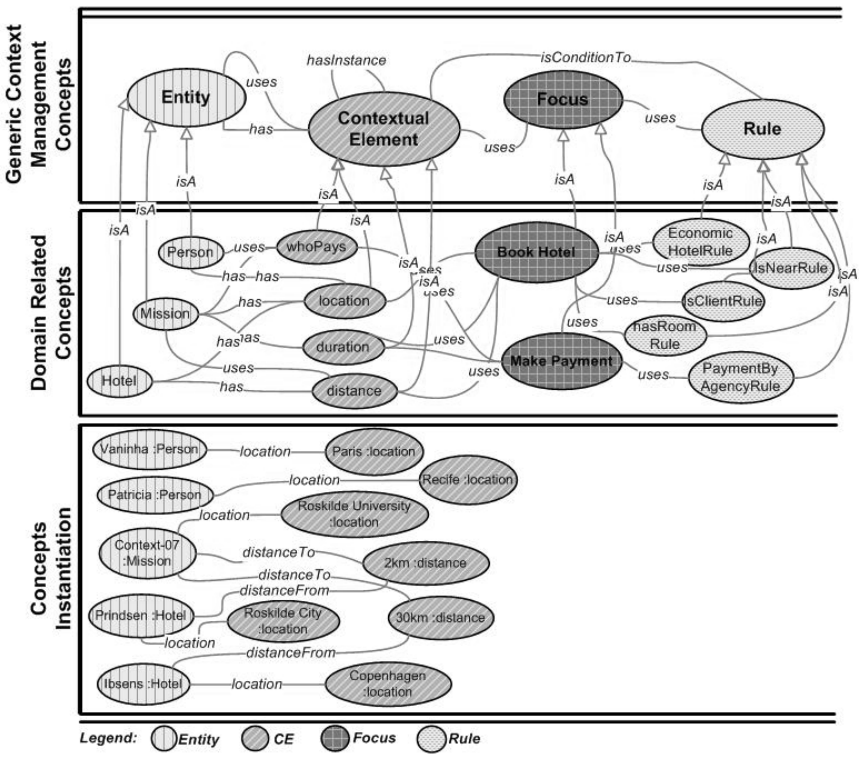


Fig. 3. Illustration of the COM Model layers interaction and the main concepts instantiation

The PC contains the set of the rules executed with the respective instantiated CEs and the final returned actions. For example, the entities *Prindsen* and *Ibsens* passed in the rule *EconomicHotelRule*, while only *Prindsen* passed for the *IsNearRule,* but the *IsClientRule* identified that the *Vaninha* is already a client of the *Ibsens*, and both hotels were returned by the *HasRoomRule* application. The recommended hotels list includes the *Prindsen* for the missionaries *Vaninha* and *Patricia*, however the *Ibsens* was also indicated to *Vaninha* since she was classified as a client of this hotel. With all this rationale contained in the PC, the context-sensitive system may, for example, show the list of recommended hotels for the two missionaries, with the explanation of how it arrived to that list. Each missionary can then apply their own criteria and select the hotel they finally want to book.

This is a small example of how the COM model works, that illustrates how it can be instantiated for a specific domain and application, shows the relation between the different layers and discusses the generation of derived concepts from main concepts. The objective of our work is to implement the CEManTIKA context manager in ways to enable an easy instantiation of the main concepts by domain/application specialists, and to implement the mechanics that permit the generation of the derived concepts.

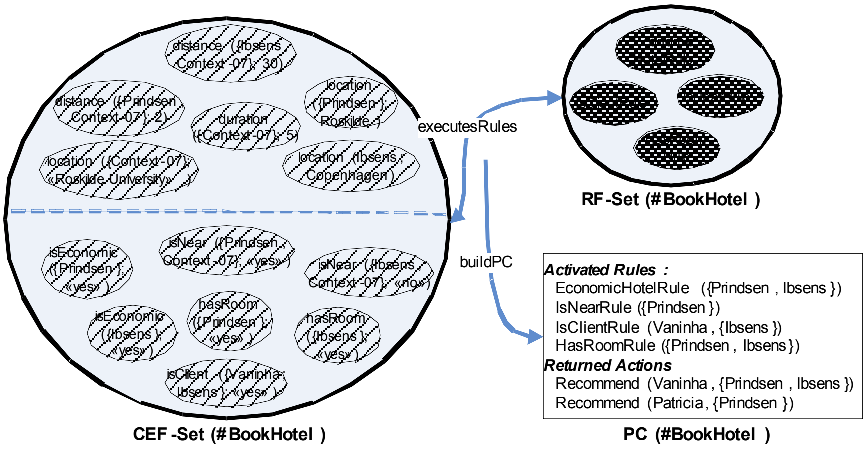

distance ({Ibsens Context-07}; 30)
distance ({Prindsen Context-07}; 2)
duration ({Context-07}; 5)
location ({Prindsen}; Roskilde)
location ({Context-07}; «Roskilde University»)
location (Ibsens; Copenhagen)
isEconomic ({Prindsen}; «yes»)
isNear ({Prindsen Context-07}; «yes»)
isNear ({Ibsens Context-07}; «no»)
isEconomic ({Ibsens}; «yes»)
hasRoom ({Prindsen}; «yes»)
hasRoom ({Ibsens}; «yes»)
isClient ({Vaninha; Ibsens}; «yes»)
executesRules
buildPC
CEF-Set (#BookHotel)
RF-Set (#BookHotel)
Activated Rules :
EconomicHotelRule ({Prindsen , Ibsens })
IsNearRule ({Prindsen })
IsClientRule (Vaninha , {Ibsens })
HasRoomRule ({Prindsen , Ibsens})
Returned Actions
Recommend (Vaninha , {Prindsen , Ibsens })
Recommend (Patricia, {Prindsen })
PC (#BookHotel)

Fig. 4. Derived concepts of COM model used to build the PC in the focus BookHotel.

6 Conclusions and Further Work

This paper presented COM (*Context-Oriented Model*) an approach for context representation that proposes the separation of the context management concepts from domain and application concepts. We assumed the hypothesis that it is impossible to imagine a context model that is at the same time specific and generic, since context is extremely domain and application-dependent. Thus, we propose that developers of context-sensitive systems rethink the way of modelling their systems including the context management phases in the system building process and considering the context related concepts when specifying the system functionalities.

Currently, we are working on the implementation of the proposed model using ontologies and topic maps. For us, the approach of topic maps seems ideal since all concepts (in the upper, middle or lower level) can be represented as topics and freely linked with one another through associations. This enables a richer approach and ease the knowledge incremental acquisition providing a flexibility in the contextual elements representation without a rigid hierarchical format. Ontologies enable the formal specification of the concepts, such as entities and contextual elements, and ease the reuse of existing solutions. We believe that this work is a first step in a new approach to model and develop context-sensitive systems as well as the integration between these systems and context managers.

Acknowledgments. Authors want to thank CNPq and CAPES for their financial support, and the first author thanks UFBA for its support.

References

1. Vieira, V., Tedesco, P., Salgado, A. C., Brézillon, P. "Investigating the Specifics of Contextual Elements Management: The CEManTIKA Approach". In: CONTEXT 2007, LNAI 4635, Roskilde, Denmark (2007), pp. 493-506.
2. Bulcão Neto, R. F., Pimentel, M. G. C. "Toward a Domain-Independent Semantic Model for Context-Aware Computing". In: Proceedings of the 3rdIW3C2 Latin American Web Congress, IEEE Computer Society, Buenos Aires, Argentina, (2005), pp. 61-70.
3. Belotti, R., Decurtins, C., Grossniklaus, M., Norrie, M. C., Palinginis, A. "Modelling Context for Information Environments". In: Workshop on Ubiquitous Mobile Information and Collaboration Systems (UMICS), CAiSE 2004, Riga, Latvia (2004).
4. Vieira, V. "The CEManTIKA Project Homepage" (2007) In: http://www.cin.ufpe.br/~vvs/cemantika/, Accessed in 03/2007.
5. Dey, A. K., Abowd, G. D. "Towards a Better Understanding of Context and Context-Awareness". In: Proceedings of the CHI 2000 Workshop on The What, Who, Where, When, and How of Context-Awareness, The Hague, Netherlands (2000).
6. Brézillon, P., Pomerol, J.-C. "Contextual Knowledge Sharing and Cooperation in Intelligent Assistant Systems", Le Travail Humain, PUF, Paris, v. 62, n. 3 (1999), pp. 223-246.
7. Gu, T., Pung, H. K., Zhang, D. Q. "A Service-Oriented Middleware for Building Context-Aware Services", Elsevier Journal of Network and Computer Applications (JNCA), v. 28, n. 1 (2005), pp. 1-18.
8. Zimmermann, A., Lorenz, A., Specht, M. "Applications of a Context-Management System". In: Proceedings of the CONTEXT-2005, Paris, France (2005), pp. 556-569.
9. Indulska, J., Robinsona, R., Rakotonirainy, A., Henricksen, K. "Experiences in Using CC/PP in Context-Aware Systems". In: Proceedings of the 4th International Conference on Mobile Data Management (MDM2003), v. LNCS 2574, Melbourne/Australia (2003), pp. 247-261.
10. Henricksen, K., Indulska, J. "Developing Context-Aware Pervasive Computing Applications: Models and Approach", Pervasive and Mobile Computing Journal (2005).
11. Vieira, V., Tedesco, P., Salgado, A. C. "Towards an Ontology for Context Representation in Groupware". In: Proceedings of the 11th International Workshop on Groupware (CRIWG'05), Porto de Galinhas, Brasil (2005), pp. 367-375.
12. Power, R. "Topic Maps for Context Management". In: Proceedings of the 1st International Symposium on Information and Communication Technologies - Workshop on Adaptive Systems for Ubiquitous Computing, Dublin, Ireland (2003), pp. 199-204.
13. Bucur, O., Beaune, P., Boissier, O. "Representing Context in an Agent Architecture for Context-Based Decision Making". In: Proc. of the International Workshop In Context Representation and Reasoning, CRR-05, Paris, France (2005).

Improving driver's situation awareness

Juliette BREZILLON[1], Patrick BREZILLON[1], Charles TIJUS[2]

[1]104 avenue du Président Kennedy, 75016, paris, France
{Juliette.Brezillon, Patrick.Brezillon}@lip6.fr
[2]2 rue de la Liberté, 93526 St Denis, France
{tijus@univ-paris8.fr}

Abstract. Initial training, which concludes by a driving license, is insufficient because new drivers do not know how to contextualize the learned procedures into effective practices. Our goal is to improve the driver's Situation Awareness, i.e. the way in which the driver perceives events in the environment, and the projection of their status in a close future. More precisely, our concern is the way in which the driver evaluates the criticality of a situation. First, we model drivers' behavior along two approaches, that is, local and global approaches, in order to have a driver model as exhaustive as possible. Second, we model drivers in a twofold representation, a situation space (an objective representation given by a lattice) and a behavior space (a subjective representation given by a contextual graph). Scenarios connect the two representations. We present in this paper the results of a specific study on driver classification based on the two approaches.

1 Introduction

Car driving is a complex activity that needs practical experiments to be safe. Initial training ends on a driving license that is often insufficient because the young driver does not know how to contextualize the learned procedures in effective practices. As a consequence, novice drivers are proportionally more involved in accident than experienced drivers [9]. [15] estimated that up to 70 % of the novice driver's errors were attributable to inexperience. Based on 1000 novices' crashes analysis, [18] conclude that inexperience was the major factor in 42 % of these accidents. Inexperience concerns several aspects of drivers' cognition, but the main factor of novice drivers' errors is an inadequate mental representation of the driving situation.

Driver's decision making is not based on an objective state of the world, but on a mental model of the driving task and the conditions in which this task is accomplished. This mental model is a « circumstantial representation » [23] built in a working memory from perceptive information extracted in a scene, and from permanent knowledge stored in the long-term memory. This representation provides a meaningful and self-oriented interpretation of the reality, including anticipations of potential evolutions in the current driving situation. This corresponds to the driver's Situation Awareness, according to [12]'s definition of this concept: "The perception of the elements in the environment within a volume of time and space, the comprehension of their meaning, and the projection of their status in the near future."

Moreover, this mental representation is “action-oriented” (i.e. the driver is an actor not a witness). It constitutes an Operative Image (i.e. a functionally deformed view of the reality [20]. Once built, such mental models generate perceptive expectations, guide the road environment exploration and the new information processing, orientate decision making and, lastly, determine all driving behaviors carried out by the driver [3]. Thus, mental representations are a key element of the driver’s cognition. An erroneous representation means, potentially, decision-making errors and unsafe driving actions. [2] illustrate the effect of inexperience at different levels of situation awareness, including information perception, driving situation understanding, and anticipation.

Hereafter, the paper is organized as follows. In the second part, we present the context of our work, the problem we aims to solve, and the methodology we use. In the third part, we present the tools we use along each approach (machine learning and cognitive sciences). In the fourth part, we conclude and present the data used. In the fifth part, we present the current state of the project and our results on our driver modeling. In the last part, we present the perspectives.

2 Presentation

2.1 Related works

Our work is based on the GADGET's methodology [13]. The GADGET project, acronym for "Guarding Automobile Drivers through Guidance Education and Technology", is a European project about road safety. It aims to assess traffic safety measures on driver behavior; analyze the influence of in-car safety devices, various road environments, education and training programs, safety campaigns, and legal measures (including enforcement) on driver behavior.

There are the three hierarchical levels – the strategic, tactical and operational level, and a fourth level is added concerning “goals for life and skills for living“. The levels also have been divided into three dimensions concerning knowledge/skill, risk increasing factors and self-assessment. The highest level refers to personal motives and tendencies in a broader perspective. This level is based on knowledge like lifestyle, social background, gender, age and other individual preconditions have an influence on attitudes, driving behavior and accident involvement. The idea in the hierarchical representation is that both failures and successes on a level affect the demands on lower levels. Thus driver's behavior must be analyzed on all these levels and not at the operational level only.

We postulate that the discrepancy between the theoretical training, which is validated by the driving license, and the effective training by driving alone (the learning-by-doing) is mainly due to a lack of support in the phase of contextualization of the theoretical training in real life situations, i.e. how to apply effectively general knowledge in a number of specific and particular situations.

Our third assumption is that works like GADGET methodology can be revisited at the light of the notion of context and mainly the process of contextualization.

A fourth assumption is that a decision support system would benefit of drivers’ experience by incrementally record drivers’ good and bad practices. Thus a system will be later able to identify the real driver's behavior, which determine a path in the situation space to allow the driver to return to a normal situation and correct behavior, and propose a scenario to support driver training.

[16] shows that it's better to learn from other people's errors than from their successes. Their results provide some support for the hypothesis that it is better to learn from other people's errors than from their successes. That's why we based our driver's typology on driving's errors.

2.2 The ACC Project

The work presented in this paper is ascribed in the ACC project (French acronym that stands for "Context-based Support Driving"). The project is presented at: www-poleia.lip6.fr/~jbrezillon (in French). The objective is to allow drivers to improve their situation awareness by simulation and by allowing drivers to learn from their drawbacks. More specifically, we want to help the driver to identify "pre-critical" situations, i.e. situations where it is yet possible to avoid the critical situation, to make the right decision and thus return to a normal situation instead of the critical situation.

We choose to lead our study from the viewpoint of a car driver instead of the usual observer's viewpoint. We thus focus more on drivers' behavior and his interpretation of the situation at hand than on the situation itself. We know that the analysis will be partial, incomplete and subjective and will lead to deal with a large number of contextual cues.

The three main elements are the situation, the driver's behavior and the scenario. Situations and behaviors are represented in different spaces, and scenarios connect the two spaces. Thus, a driver has a unique representation in the two spaces. A simple intermediate situation in the situation space is a normal situation. A leaf situation is a critical situation (e.g. a collision) or the last normal situation considered in a scenario. Drivers' behaviors are represented in the formalism of Contextual Graphs in which practices (i.e. scenario applications). Finally, a scenario is the crossing of a series of situations by a car driver. Scenarios are represented in the situation space by a tree structure. A node in the scenario tree corresponds to a pre-critical situation, i.e. a situation in which the driver has two options, a bad (resp. good) one leading to a critical (resp. normal) situation.

The overall organization relies on a typology of the drivers and data of real driving situations. Once the driver's position in the typology is known, a simulated scenario of real driving scenarios with driving problems is selected. Each scenario is adapted to the driver's learning through errors handling and errors feedback to improve situation's awareness.

A situation is a scene with a set of characteristics. The context of the situation (the situation dressing) defines some external variables (e.g. it is raining), which impact the situation characteristics.

We associate global methods resulting from machine learning and local methods resulting from cognitive sciences. The statistical training aims to model driver's classes whereas the latter relies on a cognitive modeling of drivers' behaviors. The global approach aims to model the driver from numerous data of low level (e.g. movement of eyes when driving). The goal is to process by generalization and abstraction to obtain more conceptual information (e.g. definition of classes of drivers based on real drivers' behaviors). The local approach aims to model each driver at the cognitive level that concerns the highest levels.

The association of the two approaches, the global and local approaches, allows a more complete modeling the driver at all the levels of the matrix proposed in the GADGET methodology. Thus we solve some problems found in literature, e.g. like some studies that analyze the driver at one level at a time, (e.g. the tactical level.)

3 The Tools

3.1 Questionnaire

We thus design the questionnaire in order to develop, organize and structure items of the GADGET matrix but respecting the hierarchical levels of the matrix. The questionnaire is online at http://www-poleia.lip6.fr/jbrezillon/questionnaire/ (in French). Advertisements for the questionnaire was targeted to associations concerned by accident, for retrieving points on the license, but also car schools, police, insurances, and automobile companies.

3.2 Machine learning

The process of learning based on the statistical distribution of information in a dataset is used in a class of computational models in cognitive science and psychology to describe human behavior. It is also used in computer science when using data to make predictions. We have selected two tools for their adequacy to our problem:

1. A Hidden Markov Model (HMM) is a statistical model where the system being modeled is assumed to be a Markov process with unknown parameters, and the challenge is to determine the hidden parameters from the observable parameters. The extracted model parameters can then be used to perform further analysis, for example for pattern recognition applications. In a hidden Markov model, the state is not directly visible, but variables influenced by the state are visible. Each state has a probability distribution over the possible output tokens. Therefore, the sequence of tokens generated by a HMM gives some information about the sequence of states.
2. A conditional random field (CRF) is a kind of discriminative probabilistic model most often used for the labeling or parsing of sequential data, such as natural language text or biological sequences. Similarly to a Markov random field, a CRF is an undirected graphical model in which each vertex represents a random variable whose distribution is to be inferred, and each edge represents a dependency between two random variables. In a CRF, the distribution of each discrete random variable Y in the graph is conditioned on an input sequence X.

3.3 Cognitive sciences

The methods coming from cognitive sciences considered are:

1. The STONE engine [22] is an automatic tool for structuring knowledge in a Gallois Lattice. It allows making a typology of descriptors and a typology of drivers. STONE Engine starts from input descriptors and relationships between descriptors and builds a tree of descriptors that are structured as a semantic set of dimensions. In addition, by attaching to each category what is specific to this category, the set of descriptors, which particularize each category, provide the best description of drivers of this category.
2. Contextual Graphs are a context-based formalism for representing knowledge and reasoning [6]. This formalism allows modeling the different ways in which an individual accomplishes a task. A driving situation represents the different possible scenarios for this « situation solving ». A path in this graph represents a driver's behavior in the driving situation, taking into account the different contexts considered by the user during the situation solving.

3.4 A common method: PCA

A PCA (Principal Components Analysis [1]) is a common method of machine learning and cognitive sciences. It's a technique for simplifying a dataset, by reducing multidimensional datasets to lower dimensions for analysis.

Technically speaking, PCA is a linear transformation that transforms the data to a new coordinate system such that the greatest variance by any projection of the data comes to lie on the first coordinate (called the first principal component), the second greatest variance on the second coordinate, and so on. PCA can be used for dimensionality reduction in a dataset while retaining those characteristics of the dataset that contribute most to its variance, by keeping lower-order principal components and ignoring higher-order ones. Such low-order components often contain the "most important" aspects of the data (called variables hereafter). But this is not necessarily the case, depending on the application.

4 The Data

An INRETS team at Arcueil has a simulator (embarked equipment in a car and 180-degree screen) to analyze driving situation, such as the reaction time. In terms of our approach, it is possible to simulate normal, pre-critical and critical situations.

We work now with seven associations and the French national police for the exploitation of the questionnaire on the Web. The main reasons are: large quantity of attribute for defining the driver model on several levels, it is an innovating, inexpensive, fast and effective method, it allows to collect more ecological data and offers more relevant statistical analyses, facility in finding participants and leaving the framework of the laboratory and subjects belonging to the university, a larger spectrum of subjects using Internet. Members of such associations are directly concerned by road safety, have already been involved (directly or not) in an accident, are young drivers, students, retired persons, etc.

The questionnaire is composed of 162 questions, most of them requiring a binary answer (yes or no). For the study presented in this paper, we worked on the first 166 answers, and retained as correct 419 of them. Note that the questionnaire aims to instantiate 132 variables.

5 Results

5.1 The questionnaire

Method. The questionnaire is based on the extended version of the GADGET matrix and concerns 61 variables and 162 questions. The results are based on 419 relevant answers to that questionnaire. We found 15 classes, by doing a principal composant analysis to reduce the 61 variables to 3, and we classify new data, thanks to agglomerative methods. We identify for each class the variables that represent the best the class. These variables have a specific value in a class and another value in the others classes. After, we determine in each class the variables that are related to risky behaviors. We then obtain a driver typology that is errors-based. Finally, we analyze driving behavior evolution according to the drivers' age. We wanted to know if young drivers present specific errors different from those of old drivers.

Results. We identify four steps in the evolution of the driving behaviors with the age (see figure 1):

- Discovering step: it's the step in which drivers discover what driving is, thus errors made at this step concern mainly a lack of competence for driving (as information overload, no evaluation of the necessity of a trip, no respect of the safety margins, etc.)
- Risk step: experience coming with driving, the driver looks then for his competences limits by taking risks, thus errors made at this step concern mainly risks (as personal driving style, the no respect to driving rules, etc.)
- Stable step: the driver has found and kept his driving style, and the errors made in this step are quite similar to the previous one.
- New driving style step: driver's competences decrease with the age; The driver becomes less and less self-confident; The errors made at this step concern the new way to drive (e.g. stressed, not realistic self-evaluation and drive for another reason than go somewhere – which appear at this step).

Figure 2 shows that there exists specific errors according to the age of the driver. Young drivers make competence errors by their lack of experience. Later, drivers make risky errors, searching their personal driving style. After, their behavior stays stable. Once older, drivers make errors because there is a shift between their previous way to driving few years ago and the current one. The main problem is a problem of information processing.

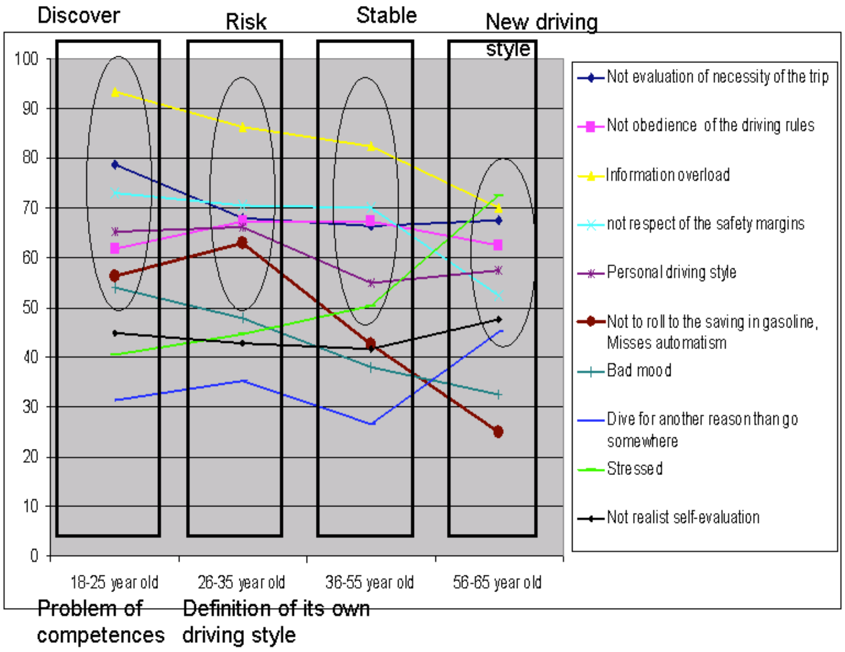


Fig.1. Evolution of the driving behavior among time

5.2 Case study

After discussion with our partners, we take a real traffic situation—a simple crossroad--and try to analyze all the driving situations that can happen. We assume only two cars arriving to the crossroad. We select the viewpoint of the driver of car A (coming from the bottom on Figure 3), and analyze all the options, first, according from where is coming the car B (from the left, the right or in front of car A), and second, according to the movement of the two cars (turn left, straight ahead, or turn right) at the crossroad. We model all the behaviors by contextual graphs (see below).

In the retained traffic situation, each road has a "give way" sign. This means that the rule is "priority to the car coming from your right."

Our modeling is based on:

- the texts of law: given the "theoretical" behavior of the driver
- a tree of situations: given the theoretical behaviors of the drivers

- the results of the questionnaire: given effective behaviors of the drivers

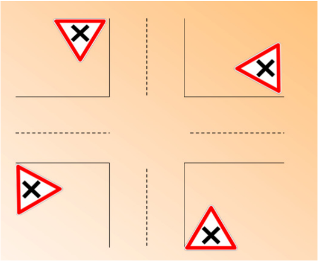


Fig. 2. The crossroad

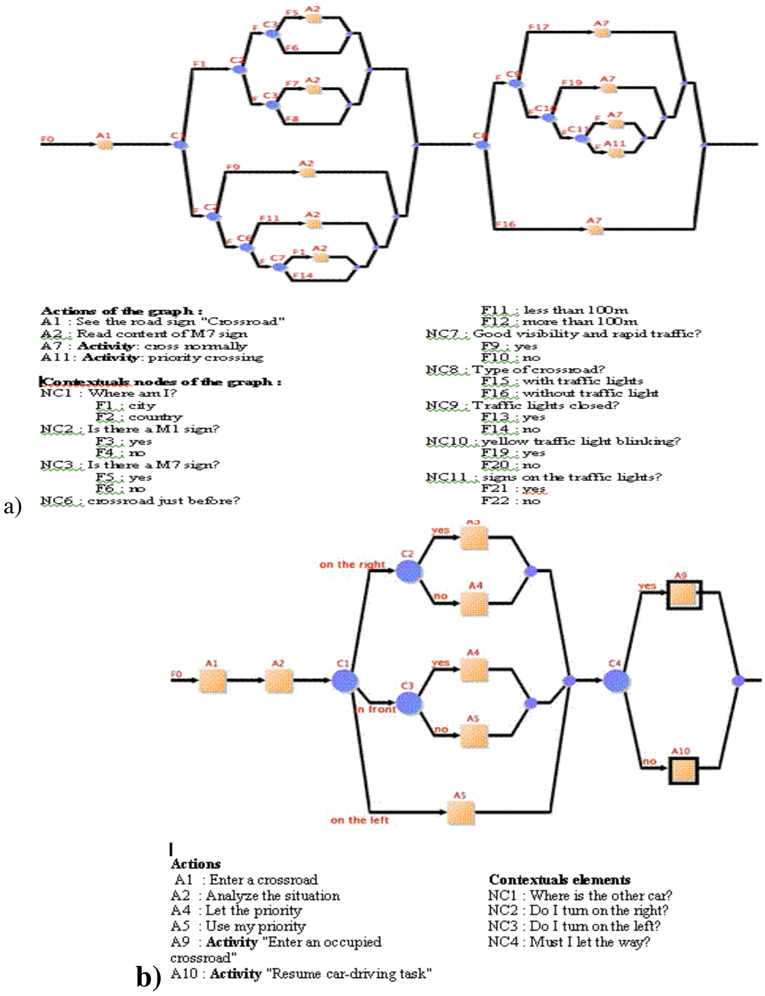
Dressing of the situation. Dressing a situation is to make its context explicit, i.e. to instantiate all the needed contextual cues. Thus, we make the following assumptions:

- The situation occurs during the day, at a normal time to drive.
- It happens in vacation, so drivers are supposed to be relaxed.
- It's not raining.
- There is no snow, or fog, or glaze in the road.
- The road is in good state.
- The car has a correct status.
- There are no pedestrian or animals that come to cross the road.
- This place that is known by the two drivers, i.e. it's not a unknown crossroad, or in a foreign country.
- The crossroad is in the countryside clearly visible by all drivers.
- The two cars can be stopped, arriving slowly, or arriving rapidly at the crossroad.
- The cars are owned by drivers.

Role of context. Most of our assumptions of the previous section concern contextual elements that are often let implicit, although they are more or less related to the driving task. [7] defines context as what constrain the driving task without intervening in it explicitly. Thus context is relative to a focus (the situation in the previous section), which allows distinguishing the contextual knowledge from the external knowledge, the former being more or less related to the focus (e.g. all the known contextual cues serving for the dressing situation). For example, the contextual cue “It is raining” will be used in the driving task as “Reduce speed” (normally).

A situation typology. This crossroad can leads to 27 initial traffic situations, according from where is coming car B and where are going the two cars.

Model of the theoretical behavior. Figure 3 shows the two successive parts of the theoretical model of drivers: (1) the analysis of the situation, and (2) the application of the decision made. Figure 3a represents the theoretical behavior of the driver that can be established from laws and the highway code. Since the crossroad has no special priority, the law defines the “theoretical” behavior as "to yield the emerging passage to the vehicles of right-hand side, by having a special vigilance and a deceleration adapted to the announced danger." There are however some restrictions. First, trams have priority, and, second, if the topology of the crossroads obliges it, a special road sign indicating the distance and/or crossroad topology is added to the road sign "Give way". The theoretical behavior defined by the law is thus to check that the roadway which it will cross is free, to circulate with speed all the more moderate as the conditions of visibility are worse, in the event of need, to announce its approach, must engage in an intersection only if its vehicle does not risk to be immobilized and to prevent the passage of the vehicles circulating on the other ways.



Actions of the graph :
A1 : See the road sign "Crossroad"
A2 : Read content of M7 sign
A7 : **Activity**: cross normally
A11 : **Activity**: priority crossing

Contextuals nodes of the graph :
NC1 : Where am I?
F1 : city
F2 : country
NC2 : Is there a M1 sign?
F3 : yes
F4 : no
NC3 : Is there a M7 sign?
F5 : yes
F6 : no
NC6 : crossroad just before?
F11 : less than 100m
F12 : more than 100m
NC7 : Good visibility and rapid traffic?
F9 : yes
F10 : no
NC8 : Type of crossroad?
F15 : with traffic lights
F16 : without traffic light
NC9 : Traffic lights closed?
F13 : yes
F14 : no
NC10 : yellow traffic light blinking?
F19 : yes
F20 : no
NC11 : signs on the traffic lights?
F21 : yes
F22 : no

Actions
A1 : Enter a crossroad
A2 : Analyze the situation
A4 : Let the priority
A5 : Use my priority
A9 : **Activity** "Enter an occupied crossroad"
A10 : **Activity** "Resume car-driving task"

Contextuals elements
NC1 : Where is the other car?
NC2 : Do I turn on the right?
NC3 : Do I turn on the left?
NC4 : Must I let the way?

Fig. 3. a) Theoretical behavior, b) Effective behaviors

Model of effective behaviors. Figure 3b represents the effective drivers' behaviors in the same situation from a study of drivers in real conditions. We analyzed what can happen concretely that was not planned by the law. First, the car's driver, which has not the priority, cannot respect it and enters the crossroad, because for instance, the car's driver thinks he has time to pass before the other car, or simply didn't see it. He can realize that he's making a mistake and decides to stop in the middle of the crossroad. The other car tries to avoid it. Moreover, the two car's drivers can break down. If a car's driver breakdown, the other driver will have to wait until the other starts again and leave the crossroad, or decides to overtake it. If he overtakes, the first car can start again and realize the other car is in front of him and try to avoid him.

Our approach in the study of driver's behaviors. Briefly, we use a very simple example to illustrate the interest of associating global and local methods. The crossroad has the form of a "T." The car B is coming from the right of car A but have a "Give way" sign and car A just goes straight ahead [8]. Figure 5 represents the different evolutions of this initial situation.

The first scenario ("1" in Figure 5) corresponds to the normal situation. Car-A's driver goes ahead and car-B's driver waits until car-A had passed and then turns right after it.

In the second scenario ("2" in Figure 5) car-B's driver goes ahead a little just to reach the road marking. There are several hypotheses for this. For example, the driver thinks to have time to realize his operation (turn right before car-A) but abandon the idea after a while. Another reason could be that the driver wants to see behind car-A if any other vehicle arrives. Car-A's driver reduces speed, observes car-B driver's behavior, and, as car-B does not move anymore, it crosses the road on the right, but carefully. After car-A is passed, car-B's driver turns right, and the second scenario meets the first scenario (see Figure 5).

In the third scenario ("3" in Figure 5) car-B's driver goes ahead until the mark on the pavement and decides to operate before car-A arrives. Conversely, car-A's driver has a different interpretation of the situation and anticipates that the other driver would stop at the mark on the pavement. However, car-A's driver takes care of the risk of a dangerous situation and understands quickly the purpose of car-B's driver when car-B goes ahead. Thus, car-A's driver has the time for breaking. As we assume that there is no other vehicle behind car-A, its driver can break easily without risk for eventual cars behind him. Car-A's driver break and stop (or at least reduce sufficiently its speed), let car-B's driver finish to turn, let some respectable distance between them and go ahead, after car-B.

A path may be associated with one or several scenarios. A given scenario may appear on several paths (i.e. a driver may express different behaviors at different moment in a given scenarios). It's the case in Figure 6 for scenario "5" in a circle, it's a collision that can happen for two reasons: the driver can not avoid it or the driver thinks he can avoid it but realize that practically he can not. The two representations being complementary, we can model the driver in a very complete way.

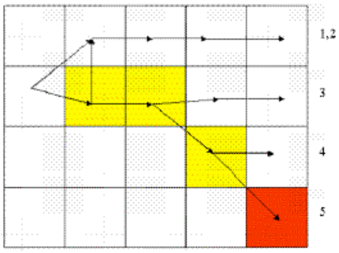


Fig. 4. The situation space

The fourth scenario supposes that, on the one hand, car-B's driver goes ahead to pass before car-A like in scenario 3, and, on the other hand, car-A's driver has a different interpretation of the situation, thinking that car-B's driver will wait before to move because car-B's driver has not the priority. Realizing that car-B's driver does not operate as expected, car-A's driver is surprised and has a short time only to react. Car-A's driver tries to break, but not enough quickly. To avoid the collision, and because there are no other vehicles than cars A and B in the area, car-A's driver decide to overtake car-B and to change lane. This decision avoids the collision of the two cars.

The fifth and last scenario is a variant of the fourth scenario. Car-A's driver tries to break, but not enough quickly. Car-A's driver has no time to change of lane (or can not do it) and a collision of the two cars thus happens.

The situation space. Figure 4 represents the previous behaviors in a situation space. Each cell is a driving situation and a path in that space corresponds to a scenario that can happen. White cells represent normal situation, grey cells pre-critical situations and the dark cell a critical situation (the collision). The distinction of the different types of situations is always relative to a given driver's viewpoint.

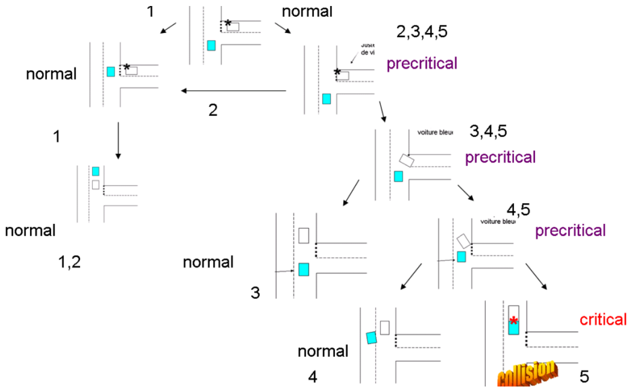
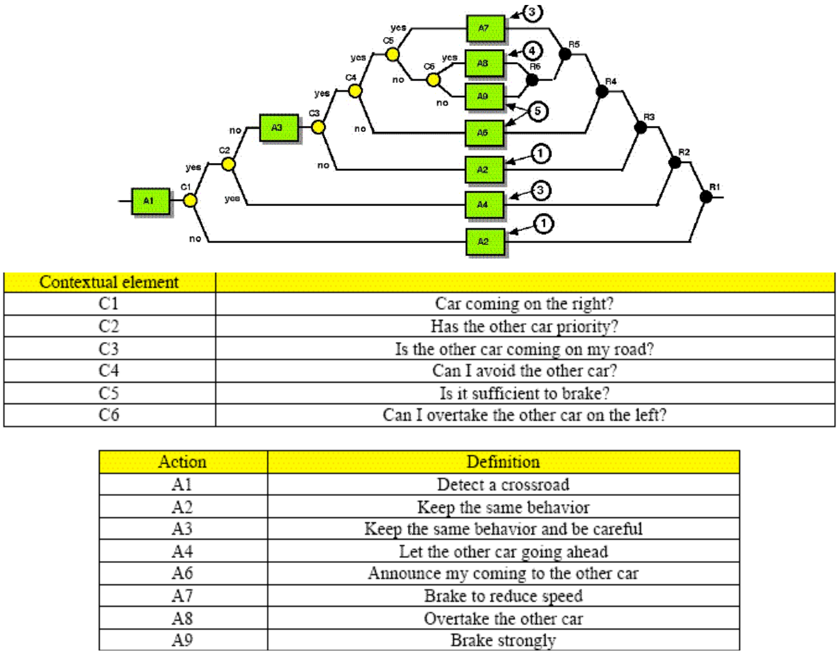


Fig. 5. The entire traffic situation

The behaviors space. Figure 6 represents some real behaviors of drivers in the contextual graphs. A path in the contextual graph corresponds to a driver's behavior.



Contextual element	
C1	Car coming on the right?
C2	Has the other car priority?
C3	Is the other car coming on my road?
C4	Can I avoid the other car?
C5	Is it sufficient to brake?
C6	Can I overtake the other car on the left?

Action	Definition
A1	Detect a crossroad
A2	Keep the same behavior
A3	Keep the same behavior and be careful
A4	Let the other car going ahead
A6	Announce my coming to the other car
A7	Brake to reduce speed
A8	Overtake the other car
A9	Brake strongly

Fig. 6. The behavior space

6 Conclusion and Perspectives

Driver modeling is an important domain that interests a number of administrations (for a uniform road security in European countries, for the police for interpreting correctly drivers' behaviors, for associations wishing to introduce some changes in the laws, etc.). Our contribution brings at least three new insights on this hot topic. First, we propose a "driver-based" classification of drivers and not an arbitrary classification. Second, we propose an open modeling in the sense that it is possible to incrementally acquire new behaviors of drivers. Third, we use good and bad practices for driver's self-learning, bad practices being mainly used by the system for identifying what is doing a given driver, and how to help him to correct his behavior.

Our goal is now to refine scenarios, unify the various representations, integrate the behaviors with the situation space, develop a tool for 3D simulation based on the situation space, confront the model to an about sixty drivers, and validate our model in real conditions.

Acknowledgments
The work presented in this paper is part of the ACC project that is supported by PREDIT and the French Minister of Transportation, concerning mainly the funding of a Ph.D. Thesis.

REFERENCES

[1] Abdi, H. (2003), Factor rotation. In M. Lewis-Beck, A. Bryman, T. Futing (Eds): Encyclopedia for research methods for the social sciences. Thousand Oaks (CA): Sage. pp. 792-795.

[2] Bailly, B., Bellet, T., Goupil, C., and Martin, R. (2003), Driver's Mental Representations: Experimental study and training perspectives, at the First International conference on driver behavior and training, Stratford-on-Avon, 2003.

[3] Bellet, T. and Tattegrain-Veste, H. (2003) COSMODRIVE: un modèle de simulation cognitive du conducteur automobile. In JC Spérandio et M. Wolf (eds), Formalismes de modélisation pour l'analyse du travail et l'ergonomie. Paris, Presses Universitaires de France, 77-110.

[4] Bellet, T., Tattegrain-Veste, H. and Bonnard, A. (2005) Risk of Collision and Behavior Adequacy Assessment for Managing Human-Machine Interaction 11th International Conference on Human-Computer Interaction (HCI), July 2005, Nevada USA.

[5] Brenac, T. (1997) L'analyse séquentielle de l'accident de la route: comment la mettre en pratique dans les diagnostics de sécurité routière, Outil et méthode, Rapport de recherche n°3, INRETS.

[6] Brézillon, P. (2005) Task-realization models in Contextual Graphs. In A. Dey, B. Kokinov, D. Leake, and R. Turner (Eds.), Modeling and Using Context, (CONTEXT-05), Springer Verlag, LNCS, 3554, pp. 55-68.

[7] Brézillon, P. and Pomerol, J.-Ch. (1999), Contextual knowledge sharing and cooperation in intelligent assistant systems. Le Travail Humain, 62, 3, 223-246.

[8] Brézillon, P., Brézillon, J. and Pomerol, J.-Ch. (2006) Decision making at a crossroad: a negotiation of contexts. Proceedings of the ICCAE Xith, Montereal, Canada.

[9] Chapelon, J. (2005) La sécurité routière en France : Bilan de l'année 2004. Rapport de l'Observatoire national interministériel de sécurité routière (ONSIR), mai 2005.

[10] CIECA (2002), Description et analyse des formations post permis des conducteurs d'automobiles et de motocyclettes (Commission Internationale des Examens de Conduite Automobile), Septembre.

[11] El Hadouaj, X . (2001) Conception de comportements de résolution de conflits et de coordination: Application à une simulation multi-agents du trafic routier, Doctoral Dissertation, Université Paris 6.

[12] Endsley, M.R. (1985) Toward a Theory of Situation Awareness in Complex Systems. Human Factors, 37, 32-64.

[13] GADGET (1999) Formation et évaluation du conducteur, obtention du permis de conduire. Vers une gestion théoriquement fondée du risque routier des jeunes conducteurs. Résultats du projet européen GADGET - Groupe de travail N°3, Stefan SIEGRIST (ed.) Berne 1999.

[14] Georgeon, O., Bellet, T., and Mille, A., Letisserand, D., Martin, R. (2005) Driver behavior modeling and cognitive engineering tools development in order to assess driver situation awareness, Proceedings of the International Workshop on Modeling Driver Behavior in Automotive Environments. 25 -27 May 2005, Joint Research Centre, Ispra, 236-241

[15] Gregersen, N.P. (1996) Young car driver: Why are they over represented in car accident? How can driver training improve their situation? VTI report 409A, Linkoping: Swedish National Road and Transport Institute.

[16] Joung, W. and Hesketh, B. (2006) Using "War Stories" to Train for Adaptive Performance: Is it Better to Learn from Error or Success?, Applied Psychology, Blackwell Publishing Ltd.

[17] Malaterre, G. (1987) Les activités sous contraintes de temps : le cas des manœuvres d'urgence en conduite automobile, Doctoral Thesis, Université Paris V.

[18] McKnight J.A. and Mc Knight S.A. (2003) Young novice drivers: Careless or clueless. Accident Analysis and Prevention, 35, 921-925.

[19] Michon, J.A. (1985) A critical view of driver behaviour models. What do we hnow, what should we do ? In L. Evans and R. Schwing (eds.), Human behaviour and traffic safety, New York, US.

[20] Ochanine, V.A. (1977) Concept of operative image in engineering and general psychology, in B.F. Lomov, V.F. Rubakhin, and V.F. Venda (Eds). Engineering Psychology. Science Publisher : Moscow.

[21] Oliver, N. (2000) Towards perceptual intelligence: Statistical modeling of human individual and interactive behaviors. Doctoral Thesis, Program in Media Arts and Sciences, Massachusetts Institute of Technology, Cambridge, MA.

[22] Poitrenaud, S., Richard, J.-F. and Tijus, C. (2005) Properties, categories and categorization. Thinking and Reasoning, 11, 151-208.

[23] Richard, J.-F. (1990) Les activités mentales. Comprendre, raisonner, trouver des solutions. Paris, Armand Colin.

[24] Van Elslande, P. (2001) Dynamique des connaissances, catégorisation et attentes dans une conduite humaine située, Thèse de doctorat en psychologie cognitive, Université Paris V.

Reverse-engineering of Individual and Inter-Personal Work Practices: A Context-based Approach

Marielba Zacarias [1,2], H.Sofia Pinto [3,4] José Tribolet [1,3]

1 Center for Organizational Engineering, INESC, Lisboa, Portugal.
2 Universidade do Algarve, ADEEC-FCT, Faro, Portugal.
3 Department of Information Systems and Computer Science, IST/UTL,Lisbon, Portugal.
4 ALGOS, INESC-ID, Lisboa, Portugal
mzacaria@ualg.pt, sofia@algos.inesc-id.pt, jose.tribolet@inov.pt

Abstract. Current enterprise modeling approaches allow representing organization's *design*. This paper describes context-based approach for organizational analysis, to discover and model aspects of the organization's *implementation*. The proposed approach aims at allowing the elaboration of graphical representations of actual work practices within given execution contexts. In particular, it addresses the discovery and analysis of personal and inter-personal contexts from action repositories. The approach is illustrated with sample graphics from case studies. Results on the automatic discovery of personal contexts are also reported.

1 Introduction and Motivation

Enterprise modeling is an overlapping activity of the Information Systems (IS) and Artificial Intelligence (AI) fields. Whereas IS models are commonly referred as Enterprise Architectures, in AI are better known as Enterprise Ontologies. Within both fields, enterprise models aim at; (1) supporting the development of business applications, (2) facilitating shared understandings among organizational members and (3) facilitating interoperability among systems [1]. Since the goal is to facilitate the communication among human and automated agents, they provide languages with syntax and semantics that seek to reduce ambiguities. Enterprise modeling frameworks provide semi-formal and graphical means to represent organization's structures and processes i.e., aspects of the organization's *design*. The hypothesis driving the present research is that enterprise modeling can be valuable tools in facilitating shared understandings of the *actual implementation of organizations*, particularly of the *specific subjects* that fulfill tasks and the *specific ways* of performing these tasks. This kind of information allows uncovering individual and collective *work practices*.

The importance of discovering work practices to improve user support has been acknowledged in [2,3]. From our point of view, analyzing work practices is also important to (1) discover problems not detected by generic tasks models and (2) assessing the alignment between design and execution. These issues entail tracing the actual relationship of workers with organizational tasks, resources and other workers. Thus, a better knowledge of organization's implementation issues is useful not only for IS developers but also for organization analysts and managers. Due to their focus on organization's design, current modeling approaches provide process-centered, role-based models that are not able to capture work practices. Moreover, these approaches regard organizations as static, mechanistic and deterministic systems that not reflect the nature of human behavior. We need a semi-formal modeling framework that captures the *complexity*, *situated* and *dynamic* behavior of people at work.

In this paper, we describe an approach to discover and model individual and interpersonal work practices. The proposed approach is based on a conceptual framework that regards organizations as complex, adaptive systems that result from the interaction among its agents [4]. This framework defines an architecture and ontology of organizations agents consistent with that view. The concept of context is an essential component of this *"architected"* ontology [5]. Drawing on this ontology, we propose an approach to discover and depict context-based representations of work practices from repositories of executed actions. More specifically, we capture subject actions in terms of <subject, verb, object> triples, that identify the human actor, action type and resources used. We use contexts to group together action streams and associated resources of each individual. These groupings are thereafter regarded as "units" to identify individual and interpersonal work patterns. The representations obtained are intended mainly for organizational analysis ends. Hence, rather than supporting an engineering process, we aim at facilitating a "reverse-engineering" of work practices. The remaining of this paper is structured as follows; Sections 2 and 3 summarize related work on enterprise and context modeling. Section 4 summarizes the underlying model of the framework. Section 5 describes the acquisition approach proposed and illustrates it with examples from case studies. Section 6 briefly summarizes results on automatic discovery of personal contexts and Section 7 gives our conclusions and future directions.

2 Enterprise modeling

Enterprise modeling approaches coming from IS/AI fields models are commonly referred as Enterprise Architectures or Enterprise Ontologies. One distinctive feature of Enterprise Architectures is enabling to model organizations from different perspectives or viewpoints and to provide means to assess the alignment between them. Enterprise Architectures are process-driven and regard agents as simple resources of business process. In AI, two well known enterprise ontologies are the Enterprise Ontology (EO) proposed by Uschold [2] and the ontologies developed within the TOVE project [12]. Several Multi-Agent Systems (MAS) development frameworks have proposed meta-models comprising several social and organizational concepts, encompassing single-agent, two-agent, group and organizational level concepts [11]. AI Enterprise ontologies and MAS meta-models provide an organizational perspective with richer sets of agent-related concepts. However these models do not fully reflect the complexity and autonomy of organizational agents.

3 Context notions and modeling approaches

Despite efforts to enlarge a shared understanding of this notion [6], the definition of context remains dependent on its application area. In a pioneer work within the AI field, McCarthy [14] introduces contexts as abstract mathematical entities to allow axioms valid within limited contexts to be expanded to transcend its original limitations. He argues that the formulas *ist(c,p)* (i.e., a proposition p is. true (ist) in a given context c) are always considered as themselves asserted within a context, that produce assertions like *ist(c',ist(c,p))*. Hence, this regress is infinite. In AI and other fields of computer sciences, context is viewed as a collection of things (sentences, propositions, assumptions, properties, procedures, rules, facts, concepts, constraints, sentences, etc) associated to some specific situation (environment, domain, task, agents, interactions, conversations, etc). In problem solving, Pomerol and Brézillon define context as the implicit constrains of each step of a problem and link context to the notion of knowledge [6]. Context surrounds a focus (e.g. a task at hand) [16]. For a given focus, context is the sum of three types of knowledge; external knowledge, contextual knowledge and proceduralized context. External knowledge represents related knowledge not relevant for a particular problem step. Contextual knowledge is the corpus of knowledge directly relevant for a problem step. Proceduralized context is the part of contextual knowledge that is invoked, assembled, organized, structured and situated according the given focus and shared among the actors involved in the decision involved. The authors represent proceduralized contexts through contextual graphs. Contextual graphs are composed by two essential concepts; (1) actions and (2) contextual nodes. Actions are elementary acts composing a task. Contextual nodes are conditions surrounding task execution that may alter the course of actions taken (e.g. location, motivation, user or time-related factors).

In cognitive sciences, B. Kokinov [8] developed a dynamic theory of context that defines it as the set of all entities that influence human (or system's) behavior on a particular occasion. The main principles of the dynamic theory of context are: (1) context is a state of the mind, (2) context has no clear-cut boundaries, (3) context consists of all associatively relevant elements and (4) context is dynamic. Sociological approaches typically regard context as networks of interacting entities (people, actors/agents and artifacts). Whereas some focus on the network elements, others focus on its emergent properties. In the latter case, context is regarded as sets of rules and resources that support and regulate interactions among agents [10]. Activity Theory [9] and Actor-Network Theory [7] have been used in modeling social contexts.

4 Conceptual Framework

The approach described in section 4 is based on a conceptual framework, which regards organizations as complex and adaptive systems that emerge from successive interactions among activity and resource-related agents [4, 5]. The framework combines five essential concepts; activities, resources, agents, roles and contexts. Activities define what organizations do. Activities use resources (inputs) and produce resources (outputs). Resources are the *things, persons* or *information* required for the realization of activities. Activities are composed of tasks, which have associated *procedures* (steps to execute them). Agents are special resources with acting, coordination/management and change/learn capabilities. Roles define the observable behavior of agents within particular interaction contexts. Based on their capabilities, each agent can play a set of activity or resource-related roles. The definition of **activities** and agent **roles** is part of the organization's *design*. Contexts emerge from *execution*. At execution time, agents perform actions that change the state of given resources. Action streams create and update action contexts, originating a network of actions and resources. Interactions are pairs of *adjacent, communicative* actions exchanged between two given agents. *Interaction* contexts emerge from interactions among agents. Contexts may contain one or more *action streams* and their associated resources. Context boundaries are defined by given topic(s), agent(s) and time-intervals. Since activities are abstractions, the relationship between activities and contexts depend on the activity definition. While a single activity may involve several contexts, a context may be related to one or several activities.

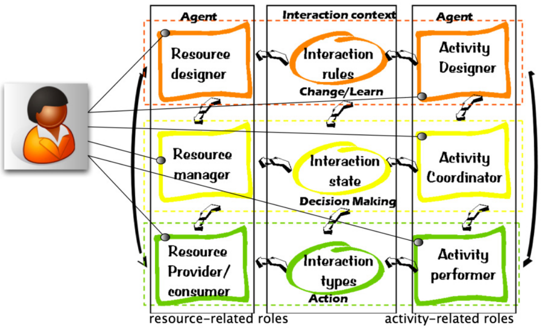


Fig. 1. Architecture and Ontology of Organizational Agents

The framework integrates agent and enterprise architectures. It is divided in three layers; action, decision-making and change/learn. The ***Action Layer*** captures action and interaction patterns between activity performer and resource provider/consumer roles, situated within specific interaction contexts. The ***Decision-making Layer*** captures how resource manager and activity coordinator agents *activate* resource provider/consumer or activity performer roles and their corresponding contexts. This layer offers a state-based view of agents and contexts. The ***Change/Learn Layer*** aims at capturing the (re)design of interaction and activation patterns of resource managers, producer/consumer, as well as activity coordinator and performer roles. In this framework, the nature of context varies according the agent layer. In the action layer, contexts are regarded as networks of agents, actions and resources. At decision-making layer, they are regarded as states of affairs. At the design layer, contexts define the set of unobservable rules governing agent behavior (for details, see [5]).

This research employs the present framework to capture individual and inter-personal behaviors at action and decision-making layers. From the action layer perspective, personal contexts are networks of actions and resources (information items, individual skills, tools and other subjects) created by action threads of an individual. Within their personal contexts, individuals see themselves as activity performers. Other agents are regarded as resource providers or consumers. Figure 2 illustrates an example personal context of an individual of our case study (Alexandre. Figure shows how this personal context, identified as the data collection for mail application context, is related to two formal tasks of Alexandre (data collection and analysis).

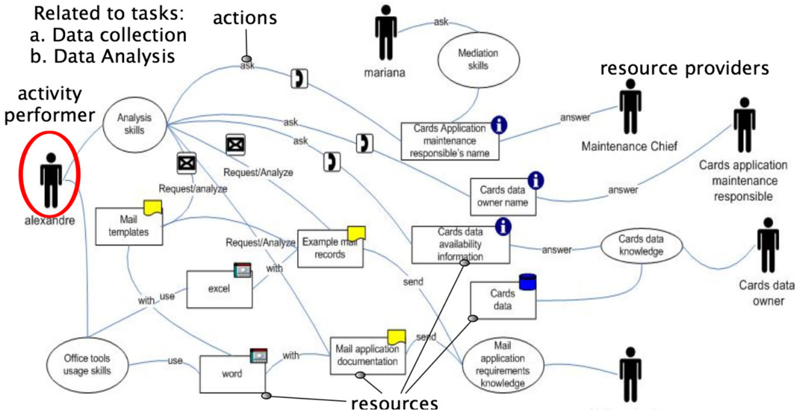


Fig. 2. A example personal context: (data collection for mail application)

Due to their multitasking abilities, people handle several, *unrelated* contexts and switch among them. Furthermore, interactions between two individuals create inter-personal contexts and the same two agents may share several inter-personal contexts. Individual and inter-personal work practices reflect not only action layer behaviors i.e. how individuals perform activities, which resources they use (tools, information or human). They also reflect behaviors of the decision-making layer such as how do they coordinate their own work and the work of others, as well as the dynamics of inter-personal networks created by action and resource flows. These practices can only be discovered analyzing the corresponding execution contexts.

5 Model Acquisition Approach

The conceptual framework described in section 3 suggests first, that agent observable behavior (action/interaction patterns) may be captured from its actions. It also suggests a separate modeling of the different complexity levels of agent behavior Third, agent behavior cannot be dissociated from their contexts of execution. Consequently, we define a bottom-up and context-based approach where we collect actions of a group of subjects, identify and analyze *action-layer* behavior i.e. typical actions and resources of personal and inter-personal contexts and infer *decision-making layer* behavior i.e., find personal and inter-personal context activation patterns. *Design* or *change/learn-layer* behavior is acknowledged suggesting a cyclic or periodic usage of the approach.

Case Studies. This approach is being developed iteratively, refining it successively from lessons learned from case studies in real organizational settings. Presently, it has been tested it in two case studies. The first case involved a software development team of a commercial bank. The main motivation of this case study was to (1) discover multitasking behavior of the team members and (2) discover team interaction patterns. The team was integrated by 4 programmers *(Gonçalo, Carla, Catarina, Alexandre*) and the project leader *(Mariana).* During the observation, the team performed tasks on the following applications; (1) *Suppliers,* (2) *Claims,* (3) *Clients' Mail Correspondence (called Mail application),* (4) *Evictions* and (5) *Marketing Campaigns*. In this case, a three-week observation was conducted, where over 650 actions were collected. A second case study was conducted within a purchasing center team of a furniture retail store. This case was motivated by the need of further improving a set of performance metrics. Five members, all performing similar activities, integrated the team. In this case, a three-week observation period was conducted, where 711 actions were collected.

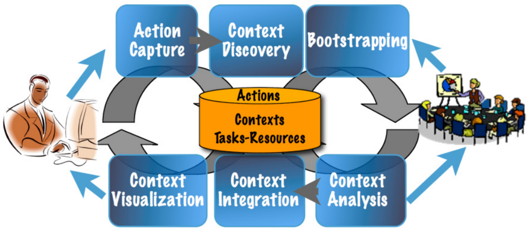


Fig. 3. A Context-Based Approach

Approach Activities. Our approach encompasses six activities; (1) bootstrapping, (2) action capture, (3) context discovery, (4) context visualization (5) context-based analysis and (6) context integration. Figure 3 depicts an overview of the approach activities. In this section we describe and illustrate them it with examples from our first case study.

1. Bootstrapping: Our basic building blocks are agent, action and resource types. Agent types are *individuals or teams*. The following action set is an example set of action types identified in one of our case studies: *accept, analyze, answer, ask assist, calculate, discuss, elaborate, evaluate, find, help, inform, install, modify, print, program, promise, propose, reject, remind, request, research, send, solve, supervise, test, update*. Resource types include *formal information items* such as documents, *informal information items* such as suggestions, ideas, facts, etc. Another type of resource are the tools employed in performing each action, which in the organizational environments we have addressed, are mainly composed of *software tools*. The basic set of action and resource types is defined after a brief observation period. The basic set is discussed and validated by the observed subjects. The collection of actions is started with this basic set, but it can be extended through the acquisition process.

2. Capturing and Structuring Actions: Traditional modeling approaches describe tasks, activities or processes with predicates (e.g. **sell** car, **buy** book, **fill** form). These descriptions lack the subject. We register actions in chronological order, and described through what we have defined as *organizational sentences* [5]. Organizational sentences (depicted in figure 4) are triples subject-**verb**-object, where the subject identifies agents, the verb identifies the action type and the object identifies the resources used or produced by the subject performing the action. *(e.g.* Gonçalo **solve** problem in Suppliers Application). Communicative actions are further structured using speech theory

[13]. Speech acts are composed of a propositional content and the intention of the sender on that proposition. Communicative actions implicate the execution of another action (which can also be a communicative action). In other words, communications actions have always embedded another action. In our approach, communicative actions took the form subject-verb-action. For example, Mariana **request** (Gonçalo **solve** problem in Suppliers Application).

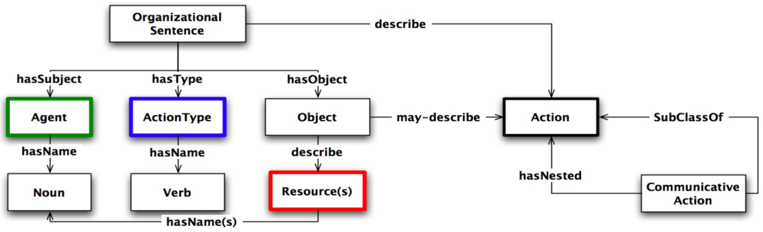


Fig. 4. The structure of Actions

The object of organizational sentences may include several noun(s) (or noun phrases) including not only informational resources but also of auxiliary tools used in performing the action. All the collected actions fitted within the present structure. Most complex actions found where two-level communicative actions, i.e. communicative actions embedding another communicative action e.g. *"Alexandre request Mariana to ask Maintenance Chief who is responsible for the cards application"*, which has the following structure: Alexandre **request** (Mariana **ask** (Maintenance Chief **answer** who is responsible for the cards application).

Table 1. Characteristics of Carla's Common Services Application context

			Object Keywords	
Context	Action	informational resources or implicated tasks	tools	human resources
c1	program	common services application	visual studio dotnet, sqlserver, msdn, google	
c1	discuss	technological issues, common services application		pedro, mariana

3-4. Context Discovery and Visualization. We discover personal contexts accordingly to our definition; grouping together action threads with similar resource sets of single individuals. Each grouping is shown to their owners, who validate and label them. Table 1 depicts the most representative actions and resources of the context c1 of Carla (Common Services Application Programming), as well as its associated set of informational resources, tools and human resources. Subjects validate (and maybe regroup) and label groupings using the information provided in this table. Table 2 depicts some labeled personal contexts of Alexandre, Carla, Mariana and Gonçalo.

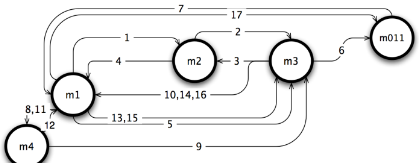


Fig. 5. Context Switches of Mariana during first observation day

5. Context-based Analysis: Identifying, characterizing and labeling contexts allow using them as unit of analysis. The identification of personal contexts allows a variety of context-based depictions Due to space limitations, we here only show a small sample of graphics from both case studies. Figure 5 illustrates the context switches of Mariana, the team leader of our first case study during the first observation day. Each circle represents a personal context of Mariana (see table 2), and numbered arrows represent context switch and its order of occurrence. These graphics were used in measuring daily work fragmentation, and in finding context activation rules.

Table 2. Some Personal Contexts

Person Name	Context ID	Context Name
Alexandre	*a1*	Data Collection for Mail Application
	a2	Mail Application Programming
	a3	Evictions Web Service Problem
	a5	Carla's Support (Web Serv & Mail App)
Carla	*c1*	Common Services Application Programming
	c2	Programming support (Mail & Suppliers App)
	c3	Team Meetings
mariana	*m1*	Project Management
	m011	Cards Information Collection
	m3	Integration Tests
	m4	Claims Application User Support
	m6	Evictions Web Service Problem
	m8	Suppliers Application Programming
goncalo	*g1*	Suppliers Application Programming
	g2	Discussions/Collaboration with Catarina
	g3	Development and User Support

Once owners have labeled their personal contexts, each action can be tagged with its corresponding context. Grouping together tagged interaction threads between two given individuals and personal contexts, allows identifying inter-personal contexts. Figure 8 depicts two inter-personal contexts; the web service problem (a3-m6) and data collection for mail application-cards information collection (a1-m011) shared by two subjects from our case study (Mariana and Alexandre). Since personal contexts reflect a personal view of an interaction context, they do not necessarily have the same label. Linking together several context pairs allows finding inter-personal, context-based networks. Each personal context is related to specific action types and resources. Thus, it is possible to build resource and action flows from these networks. Figure 9 depicts part of the resulting network from linking several context pairs.

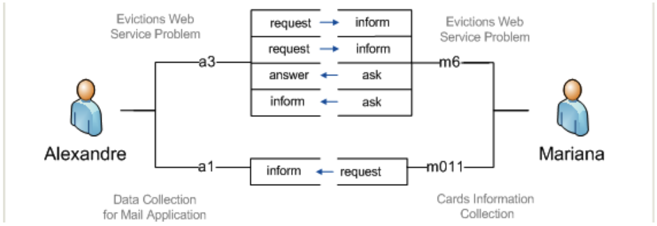


Fig. 6. Inter-personal Contexts

"Decompiling" tasks entails identifying the possible action flows from executed action threads. In this work, tasks are partially decompiled identifying recurrent action-resource sequences. These sequences are found grouping together similar action-resource threads within personal and inter-personal contexts. In our first case study, we identified several (request/inform-publication of software) sequences among developers, the team leader and the publication team. The numbered blue lines depicted in figure 9 illustrate these recurrent sequences and their frequency of occurrence within the action log. These sequences partially revealed the software publication practice implemented by the team. When considering inter-personal interactions as a contextual factor, the potential of flow variability is very high. Hence, we do not seek a complete and deterministic and specification of action flows. Rather, we aim at expressing them probabilistically. We are currently exploring the usage of *sequence clustering* algorithms to this end. Preliminary results are reported in [15].

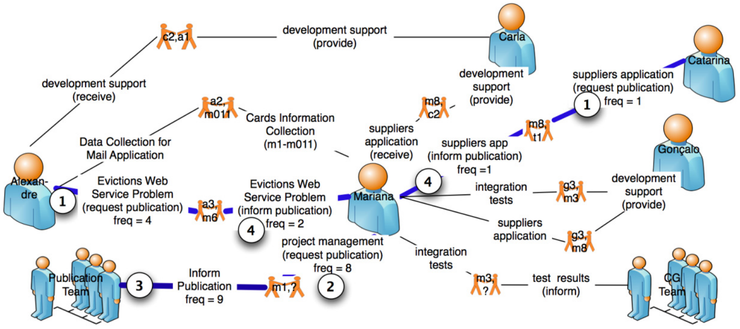


Fig. 7.A Context-based interaction network

6. Context Integration: This activity "decontextualizes" the graphics and representations from the context-based analysis activity. It is a *human* process where context-based representations are discussed and compared with current task models in order to decide their re(design). Figure 10 depicts the final form of the *official* (shared) software publication procedure after it was discussed among the team members.

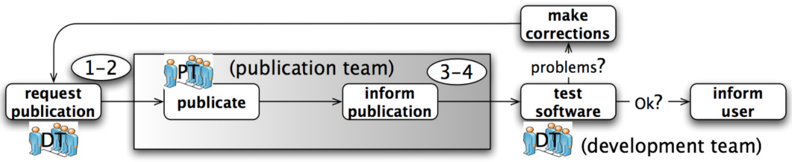


Fig. 8. The *official* publication procedure

5 Automatic Discovery of Personal Contexts

Since we are in the process of validating our approach, we have used relatively small data sets. However, a wider application will require processing of very high data volumes. Hence, it is necessary to devise automated mechanisms to support all its activities. One critical issue is the identification of personal contexts; we are currently researching on its discovery through automated means. This section provides a brief summary of the results obtained from applying a probabilistic clustering algorithm to discover personal action contexts using data coming from our first case study.

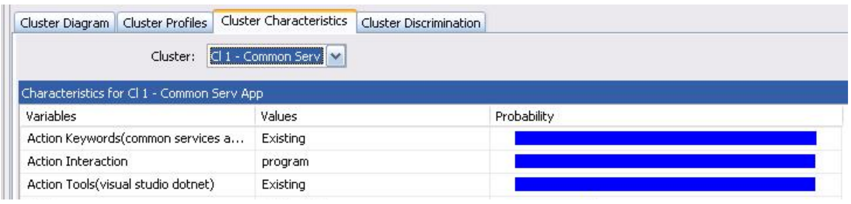
Clustering Procedure and Results. We selected a probabilistic clustering approach to discover personal contexts due to (1) the nature of the attributes (textual data) and (2) it allowed overlapping clusters. We used the Microsoft EM **Clustering algorithm** ®. Actions were stored in a MS Sql Server® data-base using the structure described in section 5.2, with their date and sequence of occurrence. Since no pre-defined structured was defined for object descriptions, before applying the algorithm, we used Sql Server text mining services to extract most recurrent noun phrases within action objects. Extracted noun phrases were analyzed and identifying nested actions, informational items, tools or people were selected as keywords. Table 3 depicts some noun phrases and their frequency.

Actions were clustered according to action type and object keywords. Since our goal was to discover personal action contexts, the clustering process was performed separately for each individual.

Table 3.Object Keywords

Keyword Te rm	frequency
suppliers application	192
Claims application	105
Team meeting	58
evictions web service	42
Mail application	31

Cluster Evaluation. We evaluated each cluster qualitative and quantitatively. The qualitative evaluation involved the analysis of the cluster diagrams, as well as cluster characteristics and profile produced by the algorithm. The algorithm discovered three clusters for the subject Carla. Figure 12 illustrates the most important characteristics of cluster 1. As depicted in figure 12, cluster 1 is characterized by the action *program*, the object keyword the *common services application* and the *tool visual studio* .NET. These characteristics suggested correspondence with Carla's personal context c1; the *Common Services Application Programming* (table 2). This procedure was applied for all clusters of all case study individuals.



Cluster: Cl 1 - Common Serv

Characteristics for Cl 1 - Common Serv App

Variables	Values	Probability
Action Keywords(common services a...	Existing	
Action Interaction	program	
Action Tools(visual studio dotnet)	Existing	

Fig. 9. Main Characteristics of the Cluster 1 of Carla

As a result, a correspondence matrix relating manual and automatic clusters was built. The qualitative evaluation allowed mapping manual and automated clusters but it does not provide a quantitative measure of its accuracy. In order to obtain an accuracy measure, a comparison matrix was built. The matrix rows identified the subjects' manual contexts and columns identified the clusters produced by the algorithm for each subject. The action-cluster distribution of the algorithm was similar to the context-cluster correspondence found by the qualitative analysis (matrices are not shown here due to space reasons). Accuracy was estimated adding the number of correctly grouped actions and dividing them by total number of actions. Accuracy estimates for each cluster, individual and overall accuracy were calculated. At a cluster level of each subject, programming and team meetings contexts exhibit more accuracy. This is consistent with the qualitative evaluation since those contexts were identified most easily than others. At an individual level, accuracy ranged from 0.89 (Carla) to 0.56 (Mariana). The overall accuracy estimate (0.71) indicates that over 70% of the total actions were correctly grouped. Table depicts the values corresponding to the subject Carla.

Table 4. Success rate of clusters found for the subject Carla

Context Description	Cluster 1	Cluster 2	Cluster 3	*Total Context*	**Success Rate**
Common Services Application Programming	***20***	1		*21*	**0.95**
Development Support		***8***	2	*10*	**0.80**
Team Meetings		1	***11***	*12*	**0.92**
Total Cluster	*20*	*10*	*13*	*43*	***0.89***

6 Conclusions and Future Work

In this work, we describe a context-based approach to discover and model personal and inter-personal work practices from action repositories. The present approach facilitates the depiction of a variety of representations to facilitate organizational analysis. The main aspects of our approach were illustrated through examples from two case studies.

In these cases, we gathered empirical evidence of the usefulness of semi-formal, graphical depictions in developing shared understandings of actual work practices. Using contexts in analyzing execution had two main benefits. First, activities are abstract concepts; associating actions to their corresponding activities requires prior knowledge of the activity definition. Our definition of context allows grouping actions without this prior knowledge. Second, it allows a situated modeling approach, appropriate in capturing the behavior specific individuals, the usage of specific resources and time-related circumstances, which is essential in capturing work practices. Moreover, it allows situating interactions among individuals at work. The present approach can be used first, to assess the alignment between tasks models and actual execution and to correct deviations. Second, to discover innovations and update current tasks models. Third, to uncover problems related to how tasks are implemented by people. Presently, the applicability of our approach is restricted to case studies conducted within limited time intervals and organizational settings. A wider and longer application entails the development of automated means to support the approach. In this paper, we briefly discuss some results of the application of clustering techniques in discovering personal action contexts. Results show that clustering produces acceptable groupings. Nonetheless, more testes need to be conducted to further improve current success rate. Another issue that must be addressed is devising ways of minimizing action entry effort. Developing automated means of extracting actions embedded in logs of collaborative tools such as e-mail applications are highly desirable. We are currently researching semantic technologies and text mining techniques to address this issue. Finally, further case studies should be conducted in order to continue refining the proposed approach.

References

1. Uschold M.. Building Ontologies: Towards a Unified Methodology, Paper presented at the16th Annual Conf. of the British Computer Society Specialist Group on Expert Systems, Cambridge (1996)
2. Alan J. Dix: Managing the Ecology of Interaction. Proceedings of the International Workshop on Task Models and Diagrams TAMODIA 2002, pp. 1-9, Bucharest (2001)
3. Brézillon, P., Using Context for Supporting Users Efficiently, Proceedings of the 36th Hawaii International Conference on System Sciences (HICSS'03), Hawaii (2003)
4. Zacarias M., Magahães R., Caetano A., Tribolet J. Towards Organizational Self-Awareness: An Initial Architecture and Ontology. In Ritten, P. (Ed), Ontologies for Business Interactions (In Press)
5. Zacarias M., Pinto H. Sofia, Tribolet, J., Integrating Engineering, Cognitive and Sociological Approaches for a Comprehensive Modeling of Organizational Agents and their Contexts, CONTEXT 2007, LCNS (LNAI) vol. 4635, pp. 517- 530, Springer, Heidelberg (In Press)
6. Brazire M., Brezillon P., Understanding Context before Using It, In Dey A., Kokinov B., Leake D., Turner R. (Eds.), CONTEXT 2005, LCNS (LNAI) vol. 3554, pp. 29-40, Springer, Heidelberg (2005)
7. Latour B. Reassembling the Social : An Introduction to Actor-Network, Oxford University Press, Oxford (2005)
8. Kokinov B., A dynamic approach to context modeling. In Proceedings of the IJCAI Workshop on Modeling Context in Knowledge Representation and Reasoning, London (1995)

9. Engeström Y., Miettinen R. and Punamäki R.L. Perspectives on Activity Theory, Cambride University Press, Cambridge (2005)
10. Giddens A., The Constitution of Society, University of California Press, California (1984)
11. Ferber ,J., Gutknecht, O. & Fabien, M. From agents to organizations: An organizational view of multi–agent systems. In Odell J., Giorgini P. & Muller J. (Eds.), AOSE 2003. LCNS, vol. 2935, pp. 214–230, Springer, Heidelberg (2003)
12. Fox, M. S., Barbuceanu, M., Gruninger, M. & and Lin, J. An Organization Ontology for Enterprise Modelling. In M. Prietula, K. Carley & L. Gasser (Eds), Simulating Organizations: Computational Models of Institutions and Groups, pp. 131-152, AAAI/MIT Press California,(1995)
13. Searle, J. Austin on locutionary and illocutionary acts. The Philosophical Review, 77, 405–42 (1978)
14. McCarthy, J., Notes on Formalizing Context, International Joint Conferences on Artificial Intelligence (IJCAI-93), France (1993)
15. Brezillon P., Task-Realization in Contextual Graphs, In Dey A., Kokinov B., Leake D., Turner R. (Eds.), Modeling and Using Context, LCNS (LNAI) vol. 3554, pp. 55-68, Springer, Heidelberg (2005)
16. Ferreira D., Zacarias M., Malheiros M., Ferreira P. Approaching Process Mining with Sequence Clustering. Accepted for the 5th International Conference on Business Process Management (BPM '07), Australia (2007)

Modeling Context in Software Reuse

Eduardo Cruz[1], Vaninha Vieira[1], Eduardo S. de Almeida[1],
Sílvio R. L. Meira[1], Ana Carolina Salgado[1], Patrick Brézillon[2]

Informatics Center – Federal University of Pernambuco (UFPE) - Brazil
Laboratoire d'Informatique de Paris 6 – Université Paris 6 (LIP6) - France
{ecrs,vvs,esa2,srlm,acs}cin.ufpe.br,
{patrick.brezillon}@lip6.fr

Abstract. Software reuse research is focused in building new software based on software artifacts previously made, in order to achieve software quality and productivity. In this field, software reuse repositories are used to store reusable software artifacts for later search and retrieval. Unfortunately, it is not common to use context neither to enrich the semantics of the software artifacts stored in the repository nor to improve the possibilities of assembly between assets for further retrieval. Assembling can be seen as a processing of contextual knowledge, which can be composed by contextual elements, that is need for a given focus. In this work we propose the improvement of an existing software reuse repository inserting context information in it.

1 Introduction

Software reuse is not a new issue in software engineering, in 1968 Douglas McIlroy discussed in your work intituled "Mass Produced Software Components" [1] saying that "*the software industry is weakly founded and one aspect of this weakness is the absence of a software component sub-industry*". By that time the term software component was very related to source code and executable software. And now the term software asset has been adopted to express the idea that any artifact of the software development life cycle like: use case diagrams, test cases or software requirements documentation can be reused.

Since there, software reuse research has been trying to achieve high productivity and quality, focusing on areas like: component based development, component certification, software product lines, component search and retrieval and asset repositories. In 2004 Almeida et. al proposed the RiSE[1] Framework [2] to achieve a systematic software reuse adoption. This framework is composed of non-technical aspects like adoption process in an enterprise and technical ones like component certification process, asset repository system, tools to help the user in a software reuse environment and best practices.

Here we focus in asset repositories as the basis of the framework where our future research will evolve. Asset repositories are intended to store information

[1] http://www.rise.com.br

and artifacts of the software development life cycle that can be used during application development. In order to retrieve the information of these repositories the system needs to receive the users queries and match it with the stored artifacts to find some information that should be the expected result. However, not every information that is stored is relevant for the search. This same concept can be seen out of the software engineer frontiers. In artificial intelligence, Brézillon and Pomerol [3] propose a definition of context and divided it in *external knowledge*(EK) and *contextual knowledge*(CK). CK is the part of the context that is relevant for the current focus while EK is the knowledge that is not relevant for the current focus. In a common search, the system may use the content of the artifact to compare it with similar artifacts. In contextual search, we may use contextual knowledge to assemble artifacts that are only related by its context because CK acts as a filter that defines, at a given time, what knowledge pieces must be taken into account (explicit knowledge) from those that are not necessary or already shared (implicit knowledge). Thus, a contextual system can help the user to assemble information between assets.

It is important to understand that Brézillon and Pomerol speak of knowledge because they consider context in reference to the knowledge of human actors, and here we are speaking of information exchanged between actors and contextual knowledge of the actor must be considered.

In this paper, we present the modeling of contextual information to improve the user software reuse experience in matching unlike software assets of different levels of abstraction. Furthermore, we define what is relevant or not to search and retrieve for the user based on the task at hand and on common context of different roles.

Hereafter the paper is organized in the following way. We start with conceptual definitions of software reuse (Section 2) to define the scope and domain we are working on. Section 3 presents previous works of the authors in the fields of software reuse. We then present the system in Section 4, related work is detailed in Section 5 and our conclusions and future work are given in Section 6.

2 Background

2.1 Software Reuse

Software reuse development can basically be divided in two different approaches. Development with reuse and development for reuse. The first means that one will develop its software based on artifacts previously used in the development of another software. The latter means the development of processes, tools, techniques and reusable artifacts for later development of software with reuse.

The expected benefits of software reuse are higher productivity, because the artifacts will not be built again but reused from previous projects. Higher quality, since reusable artifacts might have been previously tested and/or inspected.

Our approach is basically development for reuse. Where the software reuse repository will be used to store information and artifacts. These artifacts will be later used to develop new software, however, as the amount of information in the repository grows, it is important to present only relevant information for each kind of user. Thus the application will need to support the creation of different roles based on company hierarchies and/or on software engineering roles to present different products to different users with different views related to which information is relevant to each one.

2.2 Context Definition

Context is a widely used concept for different areas such as psychology, linguistic or artificial intelligence, but our focus here is based and motivated by a human-centered approach described by Brézillon in [4]. This approach helped us on the focus and scope of the context modeling and also in the definition of the contextual elements. In this case, based on the context related to the user role.

Brézillon and Pomerol [5] defines context as the collection of relevant conditions and surrounding influences that makes a situation unique and comprehensible. On the other hand they also present the problem in which numerous interacting factors that people do not even pay attention to on a conscious level, and many of which are outside the ability of machine input devices to capture.

In order to computationally treat context, it is important to make this distinction between contextual data, contextual information and contextual knowledge. Therefore, we use the term contextual element (CE) to refer to pieces of data, information or knowledge that can be used to define the context. Contextual data is the basic, atomic part of the context that can be acquired directly through virtual or physical sensors, such as location coordinates, people identification or weather temperature. Contextual information is the CE that can be derived from several contextual data through association. While the information is something that once inferred can be easily instantiated and shared between human and software agents, the contextual knowledge is personal and it is inside people's head as mental schemas that help them to interpret external events.

The focus specifies what must be contextual knowledge and external knowledge, i.e. a focus in the user roles will drive the contextual knowledge about a software development project, so the implementation elements like source code programming language might be contextual knowledge for a software developer but financial information about this project might be external knowledge for him and contextual knowledge for a project manager. Although, the focus is not static, it is interlocked with its context and evolves along the execution of a series of actions resulting from the decision making process that it follows.

In summary the contextual knowledge itself has a sub-set that it is proceduralized for addressing specifically the current focus. We call it the proceduralized context. The proceduralized context is a sub-set of contextual knowledge that is invoked, assembled, organized, structured and situated according to the given focus and is common to the various people involved in decision making.

To sum up, contextual knowledge is all the contextual information that is related to a defined task, even if the information is relevant or not to the focus. External knowledge is only the information that is not the relevant for the task or the individual. And the proceduralized context is a part of the contextual knowledge which is invoked, structured and situated according to a given focus. So, we could say that the contextual knowledge is useful to identify the activity whereas the proceduralized context is relevant to characterize the task at hand.

We cannot speak of context out of its context. Context surrounds a focus (e.g. the decision making process or the task at hand like assembling two different software artifacts) and gives meaning to items related to the focus. Thus, context guides the focus of attention, i.e. the subset of common ground that is pertinent to the current task. Indeed, context acts more on the relationships between the items in the focus than on items themselves, modifying their extension and surface. Moreover, the focus allows identifying the relevant elements to consider in the context. To help in this identification, we defined that the focus of this work would be to model contextual elements for software reuse based on the user role instead of in the artifacts. So we could have a focus of the tasks that would be executed and which contextual elements would be relevant for each role.

3 Previous Work

In this section we give an introduction to previous works developed by the authors related to software reuse and context. These previous works complement each other and are joined in the next section where we present the system.

3.1 Software Reuse

The RiSE framework [2] main objective is a systematic software reuse adoption. It has been validated in many areas sucha as software component certification, software reuse process, software reuse metrics, domain analysis, software architecture evaluation and software component search and retrieval [6].

As stated in [7], an efficient search engine should consider among other requirements the active search or proactive one. In this context, a first proactive search approach was proposed in [8] but this work did not focused on an important requirement like context information.

In this paper we consider the inclusion of context information in our search engine called B.A.R.T. (Basic Asset Retrieval Tool) project to reduce the problem identified by Frakes [9], that he identified as one of the main problem of software reuse is the "*no attempt to use*" where the user not even tries to reuse, because he is not aware of the possibility of something reusable is available for him.

B.A.R.T Project The challenge for researchers developing programming tools and environments for high-performance computing is to enable application programmers to more easily develop software systems that exploit contemporary architectures, while scaling up through the physical aspects of the problem, including problem size, data set size and complexity, the coupling of component solutions, and the complexity of numerical calculations [10].

Through the years, a vast collection of tools have been prototyped. Some of these have been developed for integrated environments, some can cooperate loosely with some others and many are freestanding [10]. Each tool or environment is still highly specific to particular context.

According to the idea that reuse can be performed in a systematic way [11], supported by an environment to aid in the software development process activities, we constructed *B.A.R.T* (*Basic Asset Retrieval Tool*). The main idea of *B.A.R.T* project is that the environment evolves in a incremental and systematic way [11], across the whole software development life cycle phases, through the integration of different techniques and methods that act to improve the effectiveness and the results of the environment. As a result, we expect to progress towards the adoption of a systematic software reuse plan. And to support the evolution of this system we aim to adopt a context management solution to go a step further.

3.2 Context Management

Context management involves the definition of models and systems to assist the acquisition, manipulation and maintenance of a shared repository of contextual elements (CE), thus enabling the usage of these elements by different context-sensitive systems. The main idea is to reduce the complexity of building context-sensitive systems, by transferring tasks related to CE manipulation to an intermediate layer. In this light, it includes the definition of a representation model to describe and share CE sets, an infrastructure to detect, update and query CE sets, mechanisms to reason, infer and process new CE sets from existing ones, and mechanisms to identify the ICE in a focus [12].

The context management process comprises the following steps: (i) acquisition of the CEs associated to a situation from virtual or physical sensors, user interfaces (e.g. forms), persistent databases, etc. (ii) to process the acquired CE through reasoning and associations the system must use knowledge bases, and inference engines. (iii) The interpreted context is used to infer information and to trigger services that must be provided and executed.

CEManTIKA Project Vieira et al. [12] presented a context management system, named CEManTIKA (Contextual Elements Management Through Incremental Knowledge Acquisition), which proposes the incremental acquisition of contextual elements according to the usage of the context-sensitive system. CEManTIKA addresses two main issues: (1) define and manage as much contextual elements as possible in the application domain; (2) identify how to use these contextual elements to assist a specific situation distinguishing the set of relevant contextual elements.

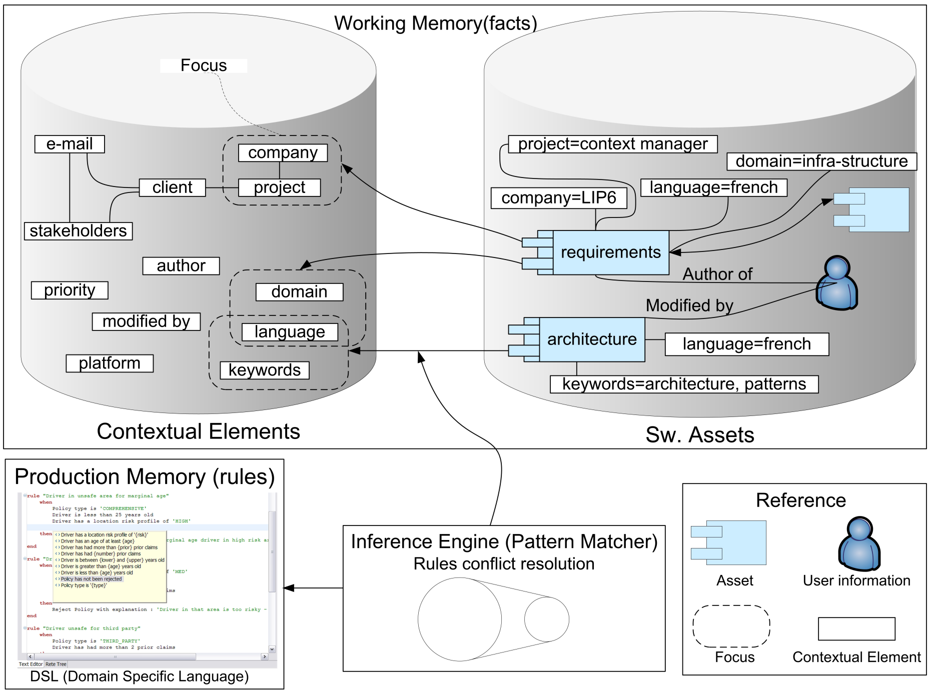
Context is a dynamic construction that evolves with the focus. As the focus changes, the set of contextual elements that must be considered changes accordingly. So, CEManTIKA manages the different focus in the domain and, for a given focus, identify which CE Sets must be considered and instantiated to support the task at hand (the ICE Set). A Proceduralized Context Base (PCB) maintains historical cases of the ICE Set built and their respective focus. The historical ICE Sets stored in the PCB aids the identification of the relevant CE in other focus.

In this joint we use CEManTIKA as an intermediate layer to support the inclusion of the context-sensitive features into B.A.R.T. The first step in this process is to identify the contextual elements involved in the software reuse domain, building a contextual elements base (CEB) that is used as the input for CEManTIKA. In a given focus CEManTIKA uses the CE stored in the CEB to build the corresponding PC that will support B.A.R.T. in the assets search and retrieval.

4 Description of The System

4.1 Archictecture

In this section, we present the proposed architecture to join the benefits the requirements and functions of both the asset manager and the context manager.


Working Memory(facts)
Focus
e-mail
company
client
project
stakeholders
author
priority
domain
modified by
language
platform
keywords
Contextual Elements
project=context manager
domain=infra-structure
company=LIP6
language=french
requirements
Author of
Modified by
architecture
language=french
keywords=architecture, patterns
Sw. Assets
Production Memory (rules)
Driver has a location risk profile of '{risk}'
Driver has an age of at least {age}
Driver has had more than {prior} prior claims
Driver has had {number} prior claims
Driver is between {lower} and {upper} years old
Driver is greater than {age} years old
Driver is less than {age} years old
Policy has not been rejected
Policy type is '{type}'
Text Editor
Rete Tree
DSL (Domain Specific Language)
Inference Engine (Pattern Matcher)
Rules conflict resolution
Reference
Asset
User information
Focus
Contextual Element

```
rule "Driver in unsafe area for marginal age"
    when
        Policy type is 'COMPREHENSIVE'
        Driver is less than 25 years old
        Driver has a location risk profile of 'HIGH'

    then
                                   rginal age driver in high risk a
end

rule "Dr
    when
                                   f 'MED'


                                   ims

    then
        Reject Policy with explanation : 'Driver in that area is too risky -
end

rule "Driver unsafe for third party"
    when
        Policy type is 'THIRD_PARTY'
        Driver has had more than 2 prior claims
```

Fig. 1. Proposed Architecture

In this architecture we defined the *Working Memory* as the source of information. This will be the source of the proceduralized context.

In this source we include the contextual elements and asset elements as described in FIGURE 1. We also have the contextual elements disposed and stored together so they can be assembled and linked and are available to be instantiated as proceduralized context. On the other hand, we have *Rules* that will treat the contextual and reusable information. These rules will be described in a Domain Specific Language developed to use the users vocabulary to make easier to learn. Finally the inference engine which makes the pattern matching between the rules defined and the information in the short term memory and is also responsible for resolving possible conflicts between predefined rules.

4.2 User Role Based Context

The context manager as a tool alone is not enough to resolve the problem of contextual asset management. We need to model the contextual information to make it usable and adaptable to be useful in different aspects of the problem.

In our approach, we centered the model in the concept of the role because the notion of role is attached to each enterprise it's culture and model of work. Each enterprise is organized in terms of role. Usually a set of tasks and responsibilities are attached to each role. One or more roles are allocated to a person (an employee, an actor). This is a very important issue because the adaption of the system to the same person acting with different roles is crucial to the different tasks a user has to accomplish.

Role, task and actor are associated with each context. The item organization gives a dependency graph on the different contexts. Each of these contexts are like a filter on the domain knowledge (the contextual elements). In short, to extract the necessary contextual elements we used the perspective of the user identifying the main roles that would interact with the system in order to extract the relevant information for each one which are describe below.

4.3 Identified Roles

The identified roles were extracted from common software development life cycle roles like Rational Unified Process. We have identified: **Project Manager**, **Software Architect**, **Software Tester**, **Software Developer**, **Configuration Manager** and **Software Quality Engineer**. Our intention is not to define all possible roles but to list some common ones that make possible to model the context to help in software reuse tasks.

Note that an actor can have multiple roles associated to him and this is more common in smaller projects.

Project manager - The project manager is the role responsible for managing the project, here we use the definition of the Project Management Group (PMI), where the responsibilities of the project manager are basically related to control the time, cost and scope of the project. Many artifacts need to be controlled and revised by the project manager. Useful reusable assets for this role would be project plans templates or project plans from similar projects where cost and time metrics could be compared.

Software Architect - This technical role is responsible for the high level structure, technology, modeling and implementation of the system. For this role architectural patterns, software requirements with functional and non-functional requirements and project test plans, and also code.

Configuration Manager - The configuration manager (CM) is the role related to the control of changes in the system. For the CM, all artifacts are relevant because he needs to track changes in any trackable artifact of the system. But for reuse purposes he would be concerned with a software configuration management plan and the project plan that also can include software configuration management information that can be reused between projects.

Software Quality Engineer - Software Quality Assurance is an issue that should concern every person in the project team, but the role that is specific related to it is the Software Quality Engineer. The actor having this role is worried about quality related activities being performed effectively. And possible reusable relevant artifacts for him are the Software Quality Plan and the project management.

Software Developer - The Software Developer is responsible for the implementation of the system. Low level technical documents, source code, components and use case documents can be reusable by software developers.

Test Engineer Test Engineers are focused in the software correctness, completeness, security and quality. And to reduce they're work and rework artifacts from previous projects like test cases and test plans can be reused by them in new projects.

4.4 Examples

The roles listed above give us the focus we need to model the context in a human centered approach. To do this, we need to think about the possible tasks at hand.

Using context to refine the search - For example, a new project manager is assigned for a project and the company has already developed other system on the same domain. In this context the project manager can have access to the previous project plans where he can see information about team sizes, costs and time to deliver versions for the client. It is expected that he will reuse this knowledge to compose his project plan with more quality than he would do with no knowledge of previous projects.

To make the relationship between the assets the user would need in this case, we need the user role to reduce the search scope and we also need other contextual elements like domain of application, project manager associated, company and sometimes even software development process can be used to assemble different project plans and retrieve then to the user even if he does not make a search, only by matching the contextual elements of his artifact with similar ones. All these examples of contextual elements can be used to match spefic contextual to refine the search and reduce the recall of assets to get a better precision.

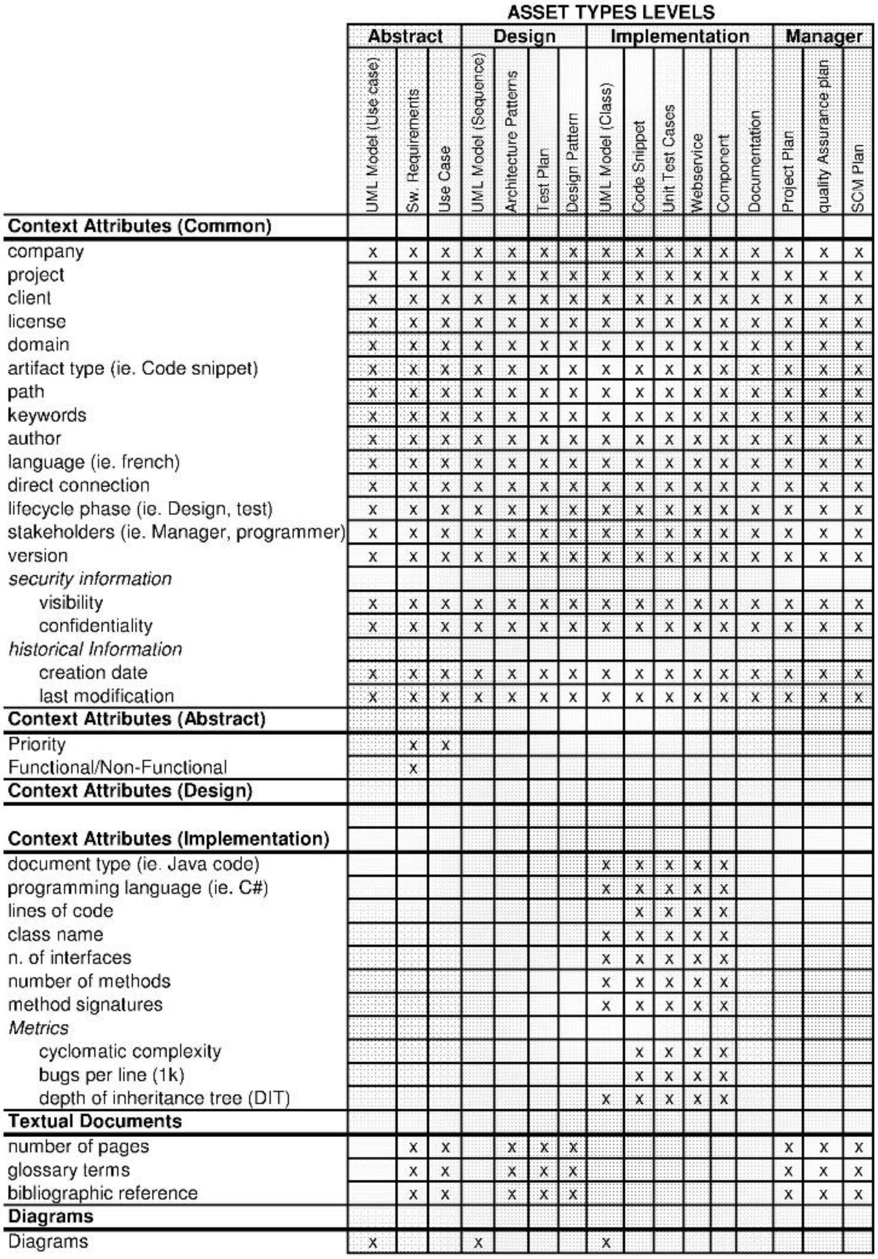
Using context to relax the search - In a different situation a software architect who is designing the software architecture of his project identifies bottleneck points in the system. To help him in this task, a search can be made using his role as one of the contextual elements and bring to him software architecture plans of similar projects or even architectural patterns that where previously described with contextual elements like domain, kind of problem is solves or even listed as a common pattern used in the enterprise. These patterns can be related to documented non-functional software requirements or use case documentation. Based on the new architecture patterns selected use case documentation modification can be proposed, based on the documentation retrieved. In this case we use the contextual elements not to refine the search and reduce the recall but to improve it and retrieve more assets because we retrieve not only the the ones that matched the query, but those which are related to the same context.

These are examples of situations where the contextual information might be used to assemble specific contextual elements which might represent a specific contextual attribute of the asset, the task or the user. In simpler terms it might also be used to assemble a composition of contextual elements that represent a given situation or context of a specific user. For example, if we use a composition of contextual elements where the role of user is test engineer, the domain of application is games, and

4.5 Contextual Elements

Contextual elements rely on the domain knowledge, and the domain knowledge rely on (for partionning purpose) on tasks and the notion of role that control tasks. Here we list contextual elements for our domain of knowledge. They are organized and selected according to the interests of the defined roles. Depending on the focus these contextual elements will be part of the external knowledge or if they are relevant for that focus and are instantiated for a given artifact they will be part of the proceduralized context. This information is essential for the system. How we model it, stored it and relate the contextual elements with the software assets. In our approach, we related the possible contextual elements with the user role, and these roles during the software development life cycle works with different kinds of assets in different levels of abstraction. Therefore, we grouped and defined these abstraction levels in FIGURE 2 as: abstract level, design level, implementation level and management level.

The first category called *management level* are assets that are not only technical but for management purposes they need to store information about different points of view like time constraints and cost of the team in a month, they are not directly related to reusable technical artifacts the can be reused to make useful the knowledge of previous projects and also save time and improve quality of new projects. Information about time constraints and experiences with software development processes and how they where mitigated in can be extremely useful. Some assets that we can include here are project plans, quality assurance plans, software configuration management plan or even cost analysis of the project.



	ASSET TYPES LEVELS															
	Abstract			Design				Implementation						Manager		
	UML Model (Use case)	Sw. Requirements	Use Case	UML Model (Sequence)	Architecture Patterns	Test Plan	Design Pattern	UML Model (Class)	Code Snippet	Unit Test Cases	Webservice	Component	Documentation	Project Plan	quality Assurance plan	SCM Plan
Context Attributes (Common)																
company	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
project	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
client	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
license	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
domain	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
artifact type (ie. Code snippet)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
path	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
keywords	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
author	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
language (ie. french)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
direct connection	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
lifecycle phase (ie. Design, test)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
stakeholders (ie. Manager, programmer)	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
version	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
security information																
visibility	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
confidentiality	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
historical Information																
creation date	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
last modification	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x	x
Context Attributes (Abstract)																
Priority		x	x													
Functional/Non-Functional		x														
Context Attributes (Design)																
Context Attributes (Implementation)																
document type (ie. Java code)								x	x	x	x	x				
programming language (ie. C#)								x	x	x	x	x				
lines of code									x	x	x	x				
class name								x	x	x	x	x				
n. of interfaces								x	x	x	x	x				
number of methods								x	x	x	x	x				
method signatures								x	x	x	x	x				
Metrics																
cyclomatic complexity									x	x	x	x				
bugs per line (1k)									x	x	x	x				
depth of inheritance tree (DIT)								x	x	x	x	x				
Textual Documents																
number of pages		x	x		x	x	x							x	x	x
glossary terms		x	x		x	x	x							x	x	x
bibliographic reference		x	x		x	x	x							x	x	x
Diagrams																
Diagrams	x			x				x								

Fig. 2. Contextual Elements Matrix

For us, *abstraction level* assets are those created in the beginning of the software development to describe the system in a high level and which will be used later to refine other system description. In this group we include software requirements documents with it's functional and non-functional requirements. Use case documents and UML use case diagrams.

Design level assets usually receive information from high abstraction level assets, refine their information and will be used later in the implementation level. We include here UML diagrams like deployment and sequence, documentation about architecture patterns, design patterns or database diagrams. Project plans that describe how the system will be tested in a high level of abstraction are included here. Nevertheless, detailed information like implementation test cases are described are in the implementation level assets.

Finally, *implementation level* assets are those with more technical details or the executable artifacts itself. They are UML class diagrams, executable code like object oriented classes or even parts of then we call *code snippets*, documented test cases or implemented unit tests, components, or documentation about APIs.

Have made the main asset type levels we also included subcategories to identify the relationship between all these contextual elements and how we could assemble them. Starting with common attributes we can assemble any asset by then no matter how it is structures and their abstraction level. For example, a simple information like what is the project related to that asset can be used to define a context of the whole project and a user with permission access to it can list can track different abstraction levels assets based on its project.

On the other hand, we have specific contextual elements in each abstraction level, this is useful when you want for example work with a user role like software developer and in the implementation level of abstraction, in this context of use, the task at hand might be retrieve assets with a contextual element like programming language, which is specific of this kind of abstraction level and won't be found in assets like the project plan. With this concepts in mind we identify which contextual element is related to each abstraction level. Where the abstraction level is one component of the user task. But can have contextual elements that are common to any level or specific ones. Depending on the need of these elements this can be used to increase the recall of assets when we bring assets related to each other based not on specific information of search but in implicit or explicit contextual information, but in the other hand, we can also use these contextual elements to restrict and decrease the search to specific abstraction levels or roles.

5 Related Work

In this section we present some related work related to formal specification, organizational learning and we show similarities and differences from our approach.

5.1 1996 - KACTUS

KACTUS [13] stands for modelling Knowledge About Complex Technical systems for multiple USe. It is an European ESPRIT-iii project aiming at the development of a methodology for the reuse of knowledge about technical systems during their life-cycle. This implies using the same knowledge base for design, diagnosis, operation, maintenance, redesign, instruction, et cetera. Reuse is be achieved by giving these knowledge bases an explicit structure (often called an ontology). Our approach does not use an ontology based solution in order to reduce the effort to introduce the solution and also reduce the need of an expert who would model the ontology.

5.2 1997 - CBR

In *Managing Software Engineering Experience for Comprehensive Reuse* [14] the authors introduce a tool architecture supporting continuous learning and reuse of of experience from the software engineering domain. Such retrieval is realized using context-sensitive queries and similarity functions based on case-based reasoning technology. While their approach is focused in cases and learning experience, ours has the same objective of help the user learn and finish the task at hand, but we are in artifact retrieval.

5.3 1999 - Formal Deduction Based

In 1999 Baar [15] used an approach called deduction-based software component retrieval. This approach uses formal specifications as component indexes and as queries, builds proof tasks from these, and checks the validity of the tasks using an automated theorem provers (ATP). A component is retrieved if the prover succeeds on the associated task-retrieval becomes a deductive problem. The problem with this approach is that using a formal method would increase the effort to model the problems and contexts and tasks at hand. Out approach gives the flexibility to use an expert to define and specify which contextual elements would compose the focus of an specific context making the process a lot more flexible.

5.4 2001 - CodeBroker

In 2001 Ye, implements the *CodeBroker [16]* project. CodeBroker is a proactive search tool wich context-aware browsing. Unfortunatelly, Ye proposes the architecture of the tool but does not define how the information must be modeled, which contextual elements must be used and which roles would be interacting with the system. Also the Codebroker is only focused in source code retrieval.

5.5 2005 - Strathcona

Holmes presents *Strathcona [17]* which is not a proactive search engine, but used the concept of structural context, based on java source code relations and dependencies to define relationship between the user activity and the source code stored in a repository. But Strathcona uses only the structural information of the source code to make the search and retrieval. Meaning that different actors, with different roles, executing different tasks, are considered the same way.

6 Conclusion and Future Work

Here we presented the evolution and merge of two fields, software reuse and context management. Software reuse is the domain we are working on as a problem to solve and context management a complementary field that we use to improve the solutions in a human centered approach. As described, it is not common to use context neither to enrich the semantics of the software artifacts stored in the repository nor to improve the possibilities of assembly between assets for further retrieval. In brief, we described how to use context to improve the semantics of the software assets stored in the repository in a manner that these context-enriched assets have more semantic information and we can use this information to related each other and propose useful artifacts for the user´s task at hand. On the other hand, we use the same approach to reduce the effort the user would need to search for a specific asset, because we can use not the explicit and conscious information he uses for a search query, but also the context of the user role, their task at hand and interacting factors that people do not even pay attention to on a conscious level as described by Brézillon and Pomerol [5].

We believe that this approach is very relevant and useful for the user and as future work we intend to make a formal collaboration between the Laboratoire d'informatique de Paris 6 (France) and the Informatics Center from Federal University of Pernambuco (Brazil) in order to go on with further research in this field. The project scope will be to continue the study of the application of context to asset management, definition of the necessary services the context-aware repository will need to have, implement the solution, validate in industrial environment and generate the evaluation reports.

As a result, we expect to improve the expertise in software reuse, context and software development quality and productivity of the partners. Exchange knowledge between institutions. Develop a product that will help the asset management of the institutions. As a parameter we expect also this project to be as relevant as the DEC VAX project caller R1/XCON (eXpert CONfigurer) with has been used with success in the eighties to save effort and money from DEC, using production rules to reduce hardware assemble errors of their sales orders. In our case, the results are expected in the software level, reducing the effort and improving the quality to produce, adapt or maintain software.

We also expect for an specification to develop an Asset Configurator at the software level for an optimal configuration based on technical internal software viewpoint and also from the user viewpoint adapted to his specific needs. Furthermore, commercial partnership with industry is also expected to result in sales of a final product.

References

1. McIlroy, M.D.: Mass produced software components. In: NATO Software Engineering Conference, Garmisch,Germany (1968) 138–155
2. de Almeida, E.S., Alvaro, A., Lucrédio, D., Garcia, V.C., de Lemos Meira, S.R.: Rise project: Towards a robust framework for software reuse. In Zhang, D.,

Grégoire, É., DeGroot, D., eds.: IRI, IEEE Systems, Man, and Cybernetics Society (2004) 48–53

3. P., B., J, P.: Contextual knowledge sharing and cooperation in intelligent assistant systems. Le Travail Humain **62**(3) (1999) 223–246
4. Brézillon, P.: Focusing on context in human-centered computing. IEEE Intelligent Systems **18**(3) (2003) 62–66
5. Brézillon, P., Pomerol, J.C.: Some comments about knowledge and context (2001)
6. Garcia, V.C., Lucrédio, D., Durão, F.A., Santos, E.C.R., de Almeida, E.S., de Mattos Fortes, R.P., de Lemos Meira, S.R.: From specification to experimentation: A software component search engine architecture. In Gorton, I., Heineman, G.T., Crnkovic, I., Schmidt, H.W., Stafford, J.A., Szyperski, C.A., Wallnau, K.C., eds.: CBSE. Volume 4063 of Lecture Notes in Computer Science., Springer (2006) 82–97
7. Lucrédio, D., do Prado, A.F., de Almeida, E.S.: A survey on software components search and retrieval. In: EUROMICRO, IEEE Computer Society (2004) 152–159
8. Mascena, J.C.C.P., Garcia, V.C., de Almeida, E.S., Meira, S.R.L.: Admire: Asset development metrics-based integrated reuse environment. In: XX Brazilian Symposium on Software Engineering. (2006)
9. Frakes, W.B., Fox, C.J.: Quality improvement using a software reuse failure modes model. IEEE Trans. Software Eng. **22**(4) (1996) 274–279
10. Harrison, W., Ossher, H., Tarr, P.: Software engineering tools and environments: A roadmap. In: 22nd International Conference on on Software Engineering, Future of Software Engineering Track, Limerick Ireland, ACM Press (2000) 261–277
11. Frakes, W., Prieto-Diaz, R., Fox, C.: DARE: Domain analysis and reuse environment. Annals of software engineering (1998)
12. Vieira, V., Tedesco, P., Salgado, A.C., Brézillon, P.: Investigating the specifics of contextual elements management: The cemantika approach. In: Context'07 (waiting for approval). (2007)
13. Schreiber, A.T.: The kactus booklet version 1.0. esprit project 8145. september, 1996. Technical report (1996)
14. Althoff, K., Birk, A., Tautz, C.: The experience factory approach: Realizing learning from experience in software development organizations (1997)
15. Thomas Baar, Bernd Fischer, D.F.: Integrating deduction techniques in a software reuse application. Volume 5., J. UCS (1999) 52–72
16. Ye, Y., Fischer, G.: Context-aware browsing of large component repositories. In: ASE, IEEE Computer Society (2001) 99–106
17. Holmes, R., Murphy, G.C.: Using structural context to recommend source code examples. In Roman, G.C., Griswold, W.G., Nuseibeh, B., eds.: ICSE, ACM (2005) 117–125

Author Index

RECENT RESEARCH REPORTS

#116 Marco Baroni, Alessandro Lenci, and Magnus Sahlgren, editors. *Proceedings of the 2007 Workshop on Contextual Information in Semantic Space Models: Beyond Words and Documents*, Roskilde, Denmark, August 2007.

#115 Paolo Bouquet, Jérôme Euzenat, Chiara Ghidini, Deborah L. McGuinness, Valeria de Paiva, Luciano Serafini, Pavel Shvaiko, and Holger Wache, editors. *Proceedings of the 2007 workshop on Contexts and Ontologies Representation and Reasoning (C&O:RR-2007)*, Roskilde, Denmark, August 2007.

#114 Bich-Liên Doan, Joemon Jose, and Massimo Melucci, editors. *Proceedings of the 2nd International Workshop on Context-Based Information Retrieval*, Roskilde, Denmark, August 2007.

#113 Henning Christiansen and Jørgen Villadsen, editors. *Proceedings of the 4th International Workshop on Constraints and Language Processing (CSLP 2007)*, Roskilde, Denmark, August 2007.

#112 Anders Kofod-Petersen, Jörg Cassens, David B. Leake, and Stefan Schulz, editors. *Proceedings of the 4th International Workshop on Modeling and Reasoning in Context (MRC 2007) with Special Session on the Role of Contextualization in Human Tasks (CHUT)*, Roskilde, Denmark, August 2007.

#111 Ioannis Hatzilygeroudis, Alvaro Ortigosa, and Maria D. Rodriguez-Moreno, editors. *Proceedings of the 2007 workshop on REpresentation models and Techniques for Improving e-Learning: Bringing Context into the Web-based Education (ReTIeL'07)*, Roskilde, Denmark, August 2007.

#110 Markus Rohde. *Integrated Organization and Technology Development (OTD) and the Impact of Socio-Cultural Concepts — A CSCW Perspective*. PhD thesis, Roskilde University, Roskilde, Denmark, 2007.

#109 Keld Helsgaun. An effective implementation of *k*-opt moves for the Lin-Kernighan TSP heuristic. 2006, Roskilde University, Roskilde, Denmark.

#108 Pernille Bjørn. *Virtual Project Teams — Distant Collaborative Practice and Groupware Adaptation*. PhD thesis, Roskilde University, Roskilde, Denmark, 2006.

#107 Henrik Bulskov Styltsvig. *Ontology-based Information Retrieval*. PhD thesis, Roskilde University, Roskilde, Denmark, 2006.

#106 Rasmus Knappe. *Measures of Semantic Similarity and Relatedness for Use in Ontology-based information Retrieval*. PhD thesis, Roskilde University, Roskilde, Denmark, 2006.

#105 Davide Martinenghi. *Advanced Techniques for Efficient Data Integrity Checking*. PhD thesis, Roskilde University, Roskilde, Denmark, 2005.