MRC 2010

Sixth International Workshop on
Modeling and Reasoning in Context

Workshop at the
19th European Conference on Artificial Intelligence (ECAI 2010)
Lisbon, Portugal, 17 August 2010

Editors:
Jörg Cassens
Anders Kofod-Petersen
Marielba Zacarias
Rebekah K. Wegener

Preface

Ambient intelligent applications do not necessarily have the luxury afforded traditional software applications of "knowing" by design in which situations they are to function. Because of the ever-changing nature of the world with which these systems interact, they have to dynamically adapt their behaviour in run time. To do this, they must be able to interpret the environment in which they are situated and adapt their behaviour accordingly.

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 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. These factors may include not only physical characteristics of the task environment, but many other aspects including cognitive factors such as the knowledge states (of both the application and user) or emotions, and social factors such as networks, relations, roles, and hierarchies. This representation and reasoning problem presents research challenges to which methodologies derived from areas such as artificial intelligence, knowledge management, human-computer interaction, semiotics and psychology can contribute solutions.

The Modelling and Reasoning in Context (MRC) workshop series aims to provide a continuing forum for scientists and practitioners exploring modelling and reasoning issues and approaches for systems using context in a broad range of tasks, to share their problems and techniques across different research and application areas. The series aim to facilitate both the presentation of state-of-the-art research as well as interactions through formal and informal discussions.

This year's workshop, the sixth in the MRC series, is being held in conjunction with the 19^{th} European Conference on Artificial Intelligence (ECAI 2010) in Lisbon, Portugal. We received eight submissions for the workshop, each of which has been reviewed by at least three programme committee members. The committee decided to accept the following six papers for presentation and inclusion in the proceedings.

Models of context are an important issue, both when conveying information about situations and when systems are to reason about the current state of the world. The context of a developer can be viewed as a rich and complex network of elements across different dimensions that are not limited to the work developed in an IDE. Bruno Antunes, Francisco Correia and Paulo Gomes propose the definition of a software developer context model that takes all the dimensions that characterise the work environment of the developer into account.

One of the basic problems in knowledge representation and knowledge engineering is the impossibility of writing globally true statements about realistic problem domains. Multi-context systems (MCS) are a formalisation of simultaneous reasoning in multiple contexts. Tarek R. Besold and Stefan Mandl propose a multi-context system framework implementation, lining out the main functions and working principles.

Detecting context features is an essential aspect of context reasoning but has mostly focused on the quality of the detection rather than the benefits of context-based applications for the user. Robert Lokaiczyk, Andreas Faatz, Benedikt Schmidt, and Matthäus Martynus present the results of a case study focusing on usability issues which evaluates algorithms to detect three subtypes of user goals in knowledge work.

Movement recognition constitutes a central task in home-based assisted living environments and in many application domains where activity recognition is crucial. Well established movement recognition methods lack advanced knowledge representation and reasoning capacities that may help understanding through contextualisation. Alessandra Mileo, Stefano Pinardi, and Roberto Bisiani investigate how a lexical approach to multi-sensor analysis can be combined with answer set programming to support movement recognition. A semantic notion of contextual coherence is formalised and qualitative optimisation criteria are introduced in the reasoning process.

Situation identification methodologies in pervasive computing aim to abstract low level context data into more meaningful high level contexts for use by context-aware application developers and users. However, it is not possible to apply a single technique to a wide range of applications. Olga Murdoch and Paddy Nixon propose a unifying framework for existing situation identification methods that will automatically select the best techniques for a given application.

Disappearing computer systems is a notion comprising computer systems, applications, and appliances related to ubiquitous, pervasive, or ambient computing which blend more or less seamlessly into the user's natural environment. Such systems lack certain cues regarding their inner workings that could cause defects and inaccuracies in the mental models built by their users. In order to identify the nature of these defects, understand their effects on human-computer interaction and develop means of avoiding them, Felix Schmitt, Jörg Cassens, Martin C. Kindsmüller, and Michael Herczeg describe an ambient, context-aware computing framework to generate and test hypotheses within an action research paradigm.

We would like to thank all authors who submitted papers for MRC 2010 for their interest in the series. Further on, we owe a big thank-you to all members of the programme committee for their work on tight schedules. We would further like to thank the organisers of the 19th European Conference on Artificial Intelligence (ECAI 2010) and in particular the workshop chair, Ulle Endriss, for their invaluable support. In addition, we would like to thank Andrei Voronkov for his fantastic EasyChair reviewing system.

June 2010

Jörg Cassens
Anders Kofod-Petersen
Marielba Zacarias
Rebekah K. Wegener

Organization

Programme Chairs

Jörg Cassens, *University of Lübeck, Germany*
Anders Kofod-Petersen, *SINTEF ICT, Trondheim, Norway*
Marielba Zacarias, *Algarve University, Portugal*
Rebekah K. Wegener, *Macquarie University, Sydney, Australia*

Programme Committee

Agnar Aamodt, *NTNU, Norway*
Patrick Brézillon, *University of Paris 6, France*
Henning Christiansen, *Roskilde University, Denmark*
Adrian Clear, *Orange Labs, France*
Lorcan Coyle, *University College Dublin, Ireland*
Babak Farshchian, *SINTEF ICT, Norway*
Chiara Ghidini, *FBK-irst, Italy*
Eyke Hüllermeier, *University of Marburg, Germany*
Martin Christof Kindsmüller, *University of Lübeck, Germany*
Boicho Kokinov, *New Bulgarian University, Bulgaria*
David Leake, *Indiana University, USA*
Ana G. Maguitman, *Universidad Nacional del Sur, Bahía Blanca, Argentina*
Sobah Abbas Petersen, *NTNU, Norway*
Enric Plaza, *Spanish Council for Scientific Research, Spain*
Thomas R. Roth-Berghofer, *DFKI GmbH, Germany*
Hedda Schmidtke, *Karlsruhe Institute of Technology (KIT), Germany*
Sven Schwarz, *DFKI GmbH, Germany*
Patrícia Tedesco, *University of Pernambuco, Brazil*
Santtu Toivonen, *Idean, Finland*
José Tribolet, *Technical University of Lisbon, Portugal*
Roy Turner, *University of Maine, USA*
Andreas Zimmermann, *Fraunhofer FIT, Germany*

Table of Contents

Towards a Software Developer Context Model

Bruno Antunes[⋆], Francisco Correia and Paulo Gomes

Knowledge and Intelligent Systems Laboratory
Cognitive and Media Systems Group
Centre for Informatics and Systems of the University of Coimbra
Coimbra, Portugal
{bema,fcorreia,pgomes}@dei.uc.pt
http://www.cisuc.uc.pt/

Abstract. The context of a developer can be viewed as a rich and complex network of elements across different dimensions that are not limited to the work developed on an IDE. We propose the definition of a software developer context model that takes into account all the dimensions that characterize the work environment of the developer. We are especially focused on the project dimension and present a definition of what the context model encompasses at this level. The experimental work done on the context capture perspective show that useful contextual information can be extracted from project management tools. The extraction, analysis and availability of these contextual information can be used to enrich the work environment of the developer with additional knowledge that relate with the tasks being developed.

Key words: Context, Software Development, Knowledge Management.

1 Introduction

The term context has an intuitive meaning for humans, but due to this intuitive connotation it remains vague and generalist. Furthermore, the interest in the many roles of context comes from different fields such as literature, philosophy, linguistics and computer science, with each field proposing its own view of context [1]. The term context typically refers to the set of circumstances and facts that surround the center of interest, providing additional information and increasing understanding.

The context-aware computing concept was first introduced by Schilit and Theimer [2], where they refer to context as *"location of use, the collection of nearby people and objects, as well as the changes to those objects over time"*. In a similar way, Brown et al. [3] define context as location, identities of the people around the user, the time of day, season, temperature, etc. In a more generic definition, Dey and Abowd [4] define context as *"any information that can be used to characterize the situation of an entity. An entity is a person,*

[⋆] Supported by FCT grant SFRH/BD/43336/2008.

place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves".

As stated by Mena et al. [5], it is clear the importance of context for modeling human activities (reasoning, perception, language comprehension, etc), which is true in social sciences as in computing. But a generic context definition is hard to achieve, as context assumes different definitions and characteristics according to the domain where it is used. Nevertheless, there are a set of invariant characteristics in the different definitions of context: context always relates to an entity; is used to solve a problem; depends on the domain of use and time; and is evolutionary because it relates to a dynamic process in a dynamic environment.

Especially in software development, the context of a developer can be viewed as a rich and complex network of elements across different dimensions that are not limited to the work developed on an IDE (Integrated Development Environment). Due to the difficulty on approaching such challenge, there is not a unique notion of what it really covers and how it can be truly exploited. One of the problems software developers face today is the increasing dimension of software systems. Software development projects have grown in complexity and size, as well as in the number of functionalities and technologies involved. During their work, software developers need to cope with a large amount of contextual information that is typically not captured and processed in order to enrich their work environment.

Our aim is the definition of a software developer context model that takes into account all the dimensions that characterize the work environment of the developer. We propose that these dimensions can be represented as a layered model with four main layers: personal, project, organization and domain. Also, we believe that a context model needs to be analyzed from different perspectives, starting with context capture, going through its modeling and representation and ending up with its use and application. This way, each layer of the proposed context model will be founded in a definition of what context capture, modeling, representation and application should be for that layer.

This work is especially focused on the project layer the software developer context model. We give a definition of what the context model encompasses at the project layer and present some experimental work on the context capture perspective.

The remaining of the paper starts with an overview of the software developer context model we envision. In section 3 we describe the current work and some preliminary experimentation is discussed in section 4. An overview of related work is given in section 5. Finally, section 6 concludes the paper and point out some directions for future work.

2 Software Developer Context Model

The software developer context model we propose take into account all the dimensions that comprise the software developer work environment. This way, we have identified four main dimensions: personal, project, organization and domain. As shown in figure 1, these dimensions form a layered model and will be described from four different perspectives: context capture, context modeling, context representation and context application. In the following sections the context model layers and perspectives are described in more detail.

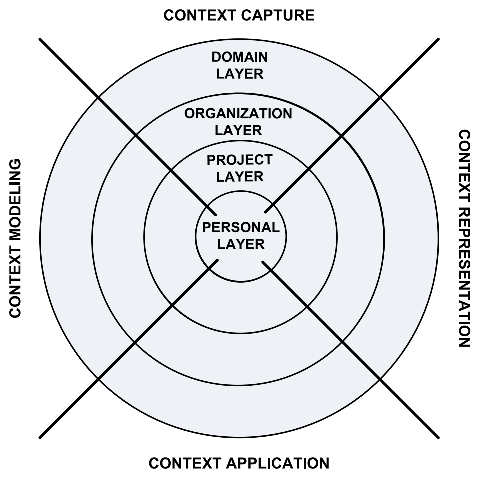


Fig. 1. The software developer context model layers and perspectives.

2.1 Context Model Layers

The developer context model we envision is based on a layered model made up of four layers: *personal layer*, *project layer*, *organization layer* and *domain layer*. Each one of these layers focus on different dimensions of the environment where a typical developer works. In the following sections we describe what is perceived as contextual information in each one of these dimensions.

Personal Layer. The personal layer represents the context of the work a developer has at hands at any point in time, which can be defined as a set of tasks. In order to accomplish these tasks, the developer has to deal with various kinds of resources at the same time, such as source code files, specification documents, bug reports, etc. These resources are typically dispersed through different places and systems, although being connected by a set of explicit and implicit relations that exist between them. At this level the context model represents the resources that are important for the tasks the developer is working on. For instance, when the developer is working on a specific task, there are a set of resources that are more relevant for that task than others and should be highlighted to the developer to facilitate her/his work.

Project Layer. The project layer focuses on the context of the project, or projects, in which the developer is somehow involved. A software development project is an aggregation of a team, a set of resources and a combination of explicit and implicit knowledge that keeps the project running. The team is responsible for accomplishing tasks, which end up consuming and producing resources. The relations that exist between people and resources are the glue that makes everything work. The project layer represents the people and resources, as well as their relations, of the software development projects where the developer is included. For instance, when fixing a specific bug it is important to know what other bugs might be related, which files are likely to be affected and which other developers working on the project may be of help to solve the bug.

Organization Layer. The organization layer takes into account the organization context to which the developer belongs. Similarly to a project, an organization is made up of people, resources and their relations, but in a much more complex network. While in a project the people and resources are necessarily connected due to the requisites of their work, in a software development organization these projects easily become separate islands. The knowledge and competences developed in each project may be of interest in other projects and valuable synergies can be created when this information is available. The organization layer represents the organizational context that surrounds a developer. For instance, when a developer needs to apply a specific technology with which no one of the project team is comfortable, there may be other colleagues, or even resources, in the same organization which can help.

Domain Layer. The domain layer takes into account the knowledge domain, or domains, in which the developer works. This layer goes beyond the project and organization levels and includes a set of knowledge sources that stand out of these spheres. Nowdays, a typical developer uses the Internet to search information and to keep informed of the advances in the technologies s/he works with. These actions are based on services and communities, such as source code repositories, development foruns, news websites, blogs, etc. These knowledge sources cannot be detached from the developer context and are integrated in the domain layer of our context model. For instance, due to the dynamic nature of the software development field, the developer must be able to gather knowledge from sources that go beyond the limits of the organization, either to follow the technological evolution or to search for help whenever needed.

2.2 Context Model Perspectives

In the following sections, we present a set of different perspectives that apply to each one of the context model dimensions discussed before: *context capture*, which represents the sources of context information and the way this information is gathered, in order to build the context model; *context modeling*, which represents the different dimensions, entities and aspects of the context information (conceptual model); *context representation*, which represents the technologies and data structures used to represent the context model (implementation); and *context application*, which represents how the context model is used and the objectives behind its use.

Context Capture. The context capture process is in the basis of any context-aware process. The quantity, quality and coverage of the context information available influences the definition of the context model at a first stage and the quality of the results in a second phase. Depending on the approach followed, this process can also be influenced by a predefined context model, which ultimately determines the context information needed. Also, the context gathering process is highly dependent on the domain and strongly limited by the environment where the user develops her/his activities.

In this work, we are concerned with the context information available in the work environment of a developer and how it can be retrieved. Most of the working time of a developer is spent in an IDE, this is the scenario where most of the action takes place, thus it is where a great amount of contextual information is available. But, the work of a developer is not limited to the use of an IDE, s/he typically needs to deal with information that is provided by different applications and systems, interact with other developers, search for information in sources inside and outside the organization scope, etc. While most of the approaches studied focus on the IDE as the only way to observe and get feedback from the developer, we believe that although the IDE represents the privileged way to get in touch with the developer, there are other paths that should not be ignored.

Context Modeling. The context model represents a definition of what context is in a specific domain. This definition must take into account the different dimensions on which context is present, as well as the various elements that pertain to each dimension. Again, the definition of a context model is highly dependent on the domain where it applies and may be very different from one application to another. It does not only depend on the characteristics of the domain, but also on the level of complexity needed in order to achieve the objectives of the application. Furthermore, it is somehow subjective, in the sense that it represents a specific perspective of what context is in a specific domain and may not be the only perspective available.

The definition of a context model for the software development domain is one of the main objectives of this work. Although some of the works studied already define some kind of context model, they commonly focus on what we consider to be a small portion of a bigger picture. This commonly happens because the objective was not to create a global model that could represent the context of the developer, but instead a context model that could aid a specific objective. Also, as discussed in the previous section, the working environment of the developer is commonly confined to the IDE, which we believe to be a reductive perspective on what the work of a developer usually comprises. We aim to create a unified context model that integrates all the dimensions of the context information provided by the working environment of a developer, as described before when discussing the layered architecture of the developer context model.

Context Representation. The context representation deals with the way context information is actually represented and depends on the complexity of the context model. The technology or data structure used must be expressive enough to represent the details of the context information and suitable to deal with the needs of the application regarding the use of that information.

The approaches studied present various ways of representing context information, ranging from simple lists to ontologies [6]. The simplest structures are easier to build and maintain, but provide limited reasoning over the context information, thus limiting the outcomes we may expect from such approaches. On the other hand, semantically rich structures, such as ontologies, provide a more powerful formalism to store and exploit context information, but are more difficult to build, maintain and validate. Ultimately, the complexity of the context model is what most determines the choice of what technology or structure to use.

Taking into account various issues, we propose to use one or more ontologies in order to represent the context model we envision. First, we think that the complexity of a global and unifying context model for software development demands the use of a complex structure, such as an ontology. Although some approaches already make use of ontologies to represent their context model, we believe that they are under-explored, especially in the software development domain. The ontologies can be represented and manipulated using the Semantic Web technologies. We believe that the use of the Semantic Web technologies is an advantage, first because of the standards accepted by the scientific community and second because they promote the reusability and knowledge sharing.

Context Application. The ultimate objective of capturing, modeling and representing context information is always the improvement of some process and the consequent improvement on the results we may expect from that process. We focus on software development, especially from the perspective of the developer.

By taking into account the context of the developer, we are able to filter out information that does not fit the actual interests of the user, to rank information in order to reflect the actual importance they may have to the developer and to elicit information that could not be highlighted otherwise. But, where we believe we can make the difference is presenting relevant information to the user that s/he never asked for, but which may be relevant for the activity s/he is performing.

The process of suggesting information to the user is of special importance when we get out of the scope of the IDE and take into account information that is not present in such environment, but is very important for the activities of the developer. The objective is always to present the most relevant information to the developer, either when s/he explicitly demands for that information or when the system considers that information to be relevant to the user.

3 Current Work

We are currently working on the project layer of our developer context model, and we will discuss our work at this level from the different perspectives we have presented before.

Concerning context capture, the main sources of contextual information that feed up the developer context model at the project level are project management tools. These tools store a big amount of explicit and implicit information about the resources produced during a software development project, how the people involved relate with these resources and how the resources relate to each other. We are focusing on two types of tools: *Version Control Systems*[1] (VCS) and *Issue Tracking Systems*[2] (ITS). As shown in figure 2, the former deals with resources and their changes, the second deals with tasks. These systems store valuable information about how developers, tasks and resources relate and how these relations evolve over time. We are especially interested in revision logs and tasks. Briefly described, a revision log tell us that a set of changes were applied by a developer to a set of resources in the repository. A task commonly represents a problem report and the respective fix.

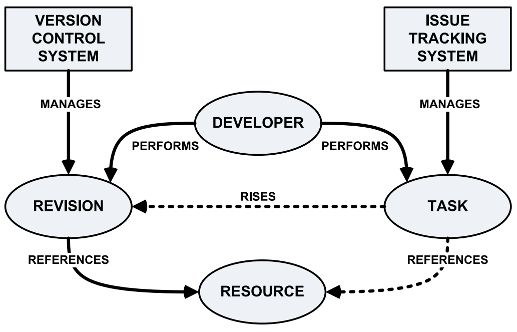


Fig. 2. The project layer relevant entities and their roles.

The network of developers, resources, tasks and their relations will be used to build our context model at the project level. This way, the context model of the developer, from a project point of view, will be modeled as a set of implicit and explicit relations, as shown in figure 3. The lines with a filled pattern represent the explicit relations and those with a dotted pattern represent the implicit ones. The developers are explicitly related with revisions, as they are the ones who commit the revisions, and with tasks, as each task is assigned to a developer. The relation between tasks and resources is not explicit, but we believe it can be identified by analyzing the information that describe tasks and revisions. The

[1] `http://en.wikipedia.org/wiki/Version_control_system`
[2] `http://en.wikipedia.org/wiki/Issue_tracking_system`

proximity between developers can be inferred by analyzing the resources were they share work. Also, the resources can be implicitly related by analyzing how often they are changed together.

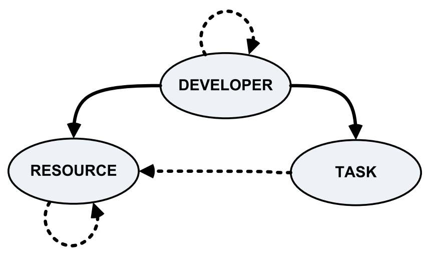


Fig. 3. The elements that compose the context model in the project layer.

In order to extract relations from the information stored in project management tools, that information is previously imported and stored locally for analysis. The prototype developed uses a database to store both the imported data and extracted relations. In the next phase, we intend to represent these relations and connection elements in an ontology, which will gradually evolve to a global developer context model ontology.

The context information extracted at the project level will be used to inform the developer about the network that links people, resources and tasks on the project s/he works. This information can be prepared to facilitate consulting and presented to the developer easily accessible in her/his working environment. This way the developer can easily gather information about what resources are likely to be implicated in the fulfillment of a task, what other tasks affected these resources in the past, and what other developers may be of help if extra information is needed.

4 Experimental Work

The experimental work conducted so far is focused on the project layer of the developer context model, which has been discussed in the previous section. We have elected VCS and ITS as two central project management tools, from the developer point of view, and started using the information provided by these tools in order to build our context model.

We have implemented connectors that allowed us to collect all the desired information from the *Subversion*[3] and *Bugzilla*[4] systems, as they are among the most popular in use. Through the collected information, we could already perceive a set of explicit relations: which resources are created/changed by which

[3] `http://subversion.tigris.org/`
[4] `http://www.bugzilla.org/`

developers, which tasks have been assigned to which developers, which resources are likely to be related because they were changed together, etc. There is also a set of implicit relations that would be valuable if disclosed: which revisions are associated with a task and which resources are associated with a task.

4.1 Relation Extraction

Our approach to extract implicit relations between revision logs and tasks relies on the analysis of the text provided by revision messages, task summaries and task comments. It is common to find references to task identifiers in revision messages, denoting that the revision was made in the context of a specific task. Also, task summaries and comments commonly reference specific resources, either because a problem has been identified in a specific class or because a error stack trace is attached to the task summary to help developers identify the source of the problem. Taking this into account, we have defined three algorithms to find resource/task and task/revision relations:

- *Resource/Task (I).* For each resource referenced in a revision, the respective name was searched in all task summaries. The search was performed using the file name and, in case it was a source code file, the full qualified name (package included) and the class name separately.
- *Resource/Task (II).* For each resource referenced in a revision, the respective name was searched in task comments. This search was performed as described for the previous relation.
- *Task/Revision.* For each task, the respective identification number was searched in revision messages. The search was performed using common patterns such as "<id>", "bug <id>"and "#<id>".

4.2 Preliminary Results

To validate the relation extraction algorithms, these were tested against two open-source projects from the Eclipse[5] foundation:

- *gEclipse*[6]. The gEclipse framework allows users and developers to access Computing Grids and Cloud Computing resources in a unified way, independently of the Grid middleware or Cloud Computing provider. Our analysis was performed over the work of 19 developers in a time window of approximately 3 years and 3 months, which included 3279 revisions and 1508 tasks with 7765 comments.
- *Subversive*[7]. The Subversive project supports the development of an Eclipse plug-in for Subversion integration. Our analysis was performed over the work of 10 developers in a time window of approximately 3 years and 2 months, which included 1120 revisions and 1013 tasks with 3252 comments.

[5] `http://www.eclipse.org/`
[6] `http://www.eclipse.org/geclipse/`
[7] `http://www.eclipse.org/subversive/`

Table 1. Number of extracted relations.

	Resource/Task (I)	**Resource/Task (II)**	**Task/Revision**
gEclipse	2527	31671	629
Subversive	208	9076	773

By applying the relation extraction algorithms to the information related with these two projects, we have gathered the results represented in table 1. The table shows the number of distinct relations extracted using each one of the algorithms in the two projects analyzed.

The results show that a large amount of implicit relations can be extracted from the analysis of the information associated to tasks and revisions. These relations complement the context model we are building by connecting tasks with related resources. With a more detailed analysis we have identified some problems with the algorithms creating some relations that do not correspond to an effective correlation between the two entities analyzed. These problems are mainly related with string matching inconsistencies and can be corrected with minor improvements in the way expressions are searched in the text.

5 Related Work

The EPOS (Evolving Personal to Organizational Knowledge Spaces) project, presented by Schwarz [7], aims to build organizational structured knowledge from information and structures owned by the elements of the organization. The world of a knowledge worker is defined as containing document-like objects, objects used for classification and applications. This work environment is taken into account when modeling the user context, which comprises information about various aspects, including currently or recently read documents, relevant topics, relevant persons, relevant projects, etc. The context model is formalized using RDFS [8], and each user's context is an up-to-date instance of this context model. The context information is gathered and modeled through user observation and/or user feedback. The gathering and elicitation of contextual information is done in two ways: context-polling, by requesting a snapshot of the user's current context; and context-listening, by subscribing the Context Service to be informed of every change in the user's context. Being a developer a knowledge worker, much of the concepts referenced in this work apply to the software development domain, but the specificities of this domain demand for a context model adapted to the reality of the work environment of a software developer.

Modularity is in the basis of the development of complex software systems and largely used to support a programmer's tasks, but not always help the programmer finding the desired information or delimit the point of interest for a specific task. Based on this, Kersten and Murphy [9] have been working on a model for representing tasks and their context. The task context is derived from an interaction history that comprises a sequence of interaction events representing operations performed on a software program's artifact. They have developed Mylar, now called Mylyn[8], which uses the information in a task context either to help focus the information displayed in the IDE, or to automate the retrieval of relevant information for completing a task. The focus of this work is the task and the knowledge elements present in the IDE that are more relevant for the fulfillment of that task. Our approach aims to define a context model that goes beyond the IDE and explores the knowledge provided by the different systems that support the software development process.

In the same line of task management and recovery, Parnin and Gorg [10] propose an approach for capturing the context relevant for a task from a programmer's interactions with an IDE, which is then used to aid the programmer recovering the mental state associated with a task and to facilitate the exploration of source code using recommendation systems. Their approach is focused on analyzing the interactions of the programmer with the source code, in order to create techniques for supporting recovery and exploration. Again, this approach is largely restricted to the IDE and the developer interaction with it.

With the belief that customized information retrieval facilities can be used to support the reuse of software components, Henrich and Morgenroth [11] propose a framework that enables the search for potentially useful artifacts during software development. Their approach exploits both the relationships between the artifacts and the working context of the developer. The context information is used to refine the search for similar artifacts, as well as to trigger the search process itself. The context information model is represented with RDF [12] statements and covers several dimensions: the user context, the working context, and the interaction context. While the focus here is in software reuse, our approach focuses on the information captured during the process development.

6 Conclusions

We have presented our approach of a software developer context model. Our context model is based on a layered structure, taking into account four main dimensions of the work environment of a software developer: personal, project, organization and domain. We proposed that each layer should be defined taking into account context capture, modeling, representation and application and explained what should be take into account in each one of these perspectives.

The current work is focused on the project layer of the software developer context model. We have discussed this layer in more detail and presented preliminary experimentation on the context capture perspective. The results show that it is possible to relate tasks and revisions/resources using simple relation extraction algorithms. The relations extracted help us defining the network that exist between developers, tasks and resources. This network represents the context model of the developer at the project layer and can be used to unveil useful information to the developer.

[8] `http://www.eclipse.org/mylyn/`

As future work we plan to integrate the contextual information in an IDE, for instance using an Eclipse plug-in, and present this information to the developer in a way that should not be intrusive but useful for the tasks that s/he is performing. Once this information is integrated in the work environment of the developer, it is crucial to understand what her/his personal context comprises, so that the information provided can be adequate to her/his information needs. The remaining layers of the context model will be addressed iteratively, as an extent to the work already developed. Finally, we pretend to test our approach with developers working in real world projects.

References

1. Mostefaoui, G.K., Pasquier-Rocha, J., Brezillon, P.: Context-aware computing: A guide for the pervasive computing community. In: Proceedings of the IEEE/ACS International Conference on Pervasive Services, ICPS 2004. (2004) 39-48
2. Schilit, B., Theimer, M.: Disseminating active map information to mobile hosts. IEEE Network (1994) 22-32
3. Brown, P.J., Bovey, J.D., Chen, X.: Context-aware applications: From the laboratory to the marketplace. Personal Communications, IEEE 4 (1997) 58-64
4. Dey, A.K., Abowd, G.D.: Towards a better understanding of context and context-awareness. In: CHI 2000 Workshop on the What, Who, Where, When, and How of Context-Awareness, The Hague, The Netherlands (2000)
5. Mena, T.B., Saoud, N.B.B., Ahmed, M.B., Pavard, B.: Towards a methodology for context sensitive systems development. In Kokinov, B.N., Richardson, D.C., Roth-Berghofer, T., Vieu, L., eds.: Modeling and Using Context, 6th International and Interdisciplinary Conference, CONTEXT 2007, Roskilde, Denmark, August 20-24, 2007, Proceedings. Volume 4635 of Lecture Notes in Computer Science., Springer (2007) 56-68
6. Zuniga, G.L.: Ontology: Its transformation from philosophy to information systems. In: Proceedings of the International Conference on Formal Ontology in Information Systems, ACM Press (2001) 187-197
7. Schwarz, S.: A context model for personal knowledge management applications. In: Modeling and Retrieval of Context, 2nd International Workshop (MRC 2005), Edinburgh, UK (2005)
8. Brickley, D., Guha, R.V.: Rdf vocabulary description language 1.0: Rdf schema (2004) Published: W3C Recommendation.
9. Kersten, M., Murphy, G.C.: Using task context to improve programmer productivity. In: Proceedings of the 14th ACM SIGSOFT International Symposium on Foundations of Software Engineering, Portland, Oregon, USA, ACM (2006) 1-11
10. Parnin, C., Gorg, C.: Building usage contexts during program comprehension. In: Proceedings of the 14th IEEE International Conference on Program Comprehension (ICPC'06). (2006) 13-22
11. Henrich, A., Morgenroth, K.: Supporting collaborative software development by context-aware information retrieval facilities. In: DEXA Workshops, IEEE Computer Society (2003) 249-253
12. Miller, E., Manola, F.: Rdf primer (2004) Published: W3C Recommendation.

Towards an Implementation of a Multi-Context System Framework

Tarek R. Besold and Stefan Mandl

University of Erlangen-Nuremberg,
Chair of Computer Science 8: Artificial Intelligence,
D-91058 Erlangen, Germany
sntabeso@i8.informatik.uni-erlangen.de
stefan.mandl@informatik.uni-erlangen.de

Abstract. This system description provides (after a limited formal introduction to the topic) an overview of our proof-of-concept multi-context system framework implementation, lining out the main functionalities and working principles. Moreover, two basic techniques for reducing computational complexity when searching for equilibria of a multi-context system are proposed.

1 Introduction

One of the basic problems in knowledge representation and knowledge engineering is the impossibility of writing globally true statements about realistic problem domains. Multi-context systems (MCS) are a formalization of simultaneous reasoning in multiple contexts. Different contexts are inter-linked by bridge rules which allow for a partial mapping between formulas/concepts/information in different contexts. Recently there have been a number of investigations of MCS reasoning (for instance, see [1] or [2]), with [3] and [4] being two of the latest contributions. In the former one, the authors describe reasoning in multiple contexts that may use different logics locally. In the latter one, the concept of MCS reasoning is extended to also integrate sub-symbolic contexts, and an algorithm for finding the equilibria of an MCS is presented.

In this paper, we want to describe how we designed and implemented a proof-of-concept MCS framework software, based on results from [3] and [4].

2 (Very) Crisp Introduction to Multi-Context Systems

As an introduction to the topic, we restate the key definitions given in [3]: First, the concept of *logic*[1] is defined in terms of input-output conditions.

[1] As in the following, no information containing satisfiability or inference rules within the corresponding logics will be given, also the denomination "pre-logic" would be justifiable. For the sake of consistency we will keep the original naming calling it a "logic".

Definition 1. *A logic $L = (\mathbf{KB}_L, \mathbf{BS}_L, \mathbf{ACC}_L)$ is composed of the following components:*

1. $\mathbf{KB}_L$ *is the set of well-formed knowledge bases of L. We assume each element of $\mathbf{KB}_L$ is a set.*
2. $\mathbf{BS}_L$ *is the set of possible belief sets,*
3. $\mathbf{ACC}_L : \mathbf{KB}_L \mapsto 2^{\mathbf{BS}_L}$ *is a function describing the "semantics" of the logic by assigning to each element of $\mathbf{KB}_L$ a set of acceptable sets of beliefs.*

Given several logics, bridge rules are used to translate between the logics.

Definition 2. *Let $L = \{L_1, \dots, L_n\}$ be a set of logics. An L_k -bridge rule over $L, 1 \leq k \leq n$, containing m conditions, is of the form*

$$s \leftarrow (r_1 : p_1), \dots, (r_j : p_j), \mathbf{not}(r_{j+1} : p_{j+1}), \dots, \mathbf{not}(r_m : p_m) \qquad (1)$$

where $j \leq m$, $1 \leq r_k \leq n$, p_k is an element of some belief set of L_{r_k} and s is a syntactically valid element of a knowledge base from $\mathbf{KB}_k$,[2] *and for each $kb \in \mathbf{KB}_k : kb \cup \{s\} \in \mathbf{KB}_k$.*

A configuration of logics and bridge rules comprises a multi-context system.

Definition 3. *A multi-context system $M = (C_1, \dots, C_n)$ consists of a collection of contexts $C_i = (L_i, kb_i, br_i)$, where $L_i = (\mathbf{KB}_i, \mathbf{BS}_i, \mathbf{ACC}_i)$ is a logic, kb_i a knowledge base (an element of $\mathbf{KB}_i$), and br_i is a set of L_i-bridge rules over $\{L_1, \dots, L_n\}$.*

A belief state is the combination of the belief sets of all contexts of the MCS.

Definition 4. *Let $M = (C_1, \dots, C_n)$ be a MCS. A belief state is a sequence $S = (S_1, \dots, S_n)$ such that each S_i is an element of $\mathbf{BS}_i$.*

Now we can clarify the applicability of a bridge rule in the context of a belief state: We say a bridge rule r of form (1) is applicable in a belief state $S = (S_1, \dots, S_n)$ iff for $1 \leq i \leq j : p_i \in Th(S_{r_i})$ and for $j+1 \leq k \leq m : p_k \notin Th(S_{r_k})$.[3]

A belief state is an equilibrium if the consequences of all applicable bridge rules are given or derivable, hence each context has an acceptable belief set given the belief sets of the other contexts.

Definition 5. *A belief state $S = (S_1, \dots, S_n)$ of M is an equilibrium iff, for $1 \leq i \leq n$, the following condition holds:*

$$S_i \in \mathbf{ACC}_i(kb_i \cup \{head(r) | r \in br_i \text{ applicable in } S\}).^4$$

[2] In contrast to [3] where a similar constraint concerning the nature of s is imposed only implicitly.

[3] $Th(\cdot)$ indicates the deductive closure, i. e. $Th(\Theta) = \{\varphi \in \mathbb{L} \mid \Theta \models \varphi\}$, where Θ is a set of formulae, φ is a formula, and $\mathbb{L}$ is a logic.

[4] Please note that, if r is a bridge rule of the form $a \leftarrow \dots$, then $head(r) = a$

Equilibria are a fundamental concept of multi-context systems as they make the introduction of a notion of semantics for MCS possible, and under certain circumstances exhibit properties analogue to those of fixpoints as means of semantics: In [3], a restricted class of MCS is introduced, for which the authors present a reduction similar to the Gelfond/Lifschitz reduction for logic programs. For this class of "reducible MCS", the notion of "grounded equilibria" is given, following ideas and terminology from logic programming. Brewka and Eiter finally define a well-founded semantics for reducible MCS by constructing an operator involving the concepts of reducibility and grounded equilibria, and using a fixpoint argument in the style of the Knaster-Tarski theorem.

For the implementation of the proof-of-concept MCS framework, we used a generalized account of the just outlined theory, which also allows for the incorporation of sub-symbolic contexts of reasoning. We will now briefly restate the relevant concepts (introduced in [4], where also a more thorough discussion of the now following account may be found), which may be understood as direct generalizations of the above stated ideas.

Definition 6. *A* reasoner *is a 5-tuple* $R = (\mathbf{Inp}_R, \mathbf{Res}_R, \mathbf{ACC}_R, \mathbf{Cond}_R, \mathbf{Upd}_R)$ *where*

1. $\mathbf{Inp}_R$ *is the set of possible* inputs *to the reasoner.*
2. $\mathbf{Res}_R$ *is the set of possible* results *of the reasoner*
3. $\mathbf{ACC}_R : \mathbf{Inp}_R \mapsto 2^{\mathbf{Res}_R}$ *defines the actual reasoning, assigning each input a set of results in a decidable manner.*
4. $\mathbf{Cond}_R$ *is a set of decidable conditions on inputs and results:*

$$cond_R : \mathbf{Inp}_R \times \mathbf{Res}_R \mapsto \{0, 1\}.$$

5. $\mathbf{Upd}_R$ *is a set of update functions for inputs:*

$$upd_R : \mathbf{Inp}_R \mapsto \mathbf{Inp}_R \,.$$

Please note that in contrast to Definition 1 we do not make any assumption about the form of the input to the reasoner.

The following definitions adapt the basic concepts of multi-context reasoning given in Definitions 2 to 5 for the use with reasoners as of Definition 6.

Definition 7. *Let* $R = \{R_1, \ldots, R_n\}$ *be a set of reasoners. An* R_k *-bridge rule over* $R, 1 \leq k \leq n$*, containing* m *conditions, is of the form*

$$\begin{aligned} u \leftarrow (r_1 : c_1), \ldots, (r_j : c_j), \\ \mathbf{not}(r_{j+1} : c_{j+1}), \ldots, \mathbf{not}(r_m : c_m) \end{aligned} \tag{2}$$

where $j \leq m$*,* $1 \leq r_k \leq n$ *and* c_k *is a condition of inputs and results of some* R_{r_k} *and* u *is an element of the* $\mathbf{Upd}_k$*.*

Analogously to Definition 3 we now define a generalized MCS.

Definition 8. *A generalized multi-context system* $M = (C_1, \ldots, C_n)$ *consists of a collection of contexts* $C_i = (R_i, inp_i, br_i)$*, where* $R_i = (\mathbf{Inp}_i, \mathbf{Res}_i, \mathbf{ACC}_i, \mathbf{Cond}_i, \mathbf{Upd}_i)$ *is a reasoner,* inp_i *an input (an element of* $\mathbf{Inp}_i$ *), and* br_i *is a set of* R_i*-bridge rules over* $\{R_1, \ldots, R_n\}$ *as of equation (2).*

Belief states also have to be adapted.

Definition 9. *Let* $M = (C_1, \ldots, C_n)$ *be a generalized* MCS*. A generalized belief state is a sequence* $S = (S_1, \ldots, S_n)$ *such that each* S_i *is of the form* (inp_i, res_i) *with* $inp_i \in \mathbf{Inp}_i$ *and* $res_i \in \mathbf{Res}_i$*.*

We say a bridge rule r of form (2) is applicable in a generalized belief state $S = (S_1, \ldots, S_n)$ iff for $1 \leq i \leq j : c_i(inp_i, res_i) = 1$ in S_i and for $j+1 \leq k \leq m : c_k(inp_k, res_k) = 0$ in S_k. Now we prepare for the concept of equilibrium in the generalized setting.

Definition 10. *The set of (context local) update functions with respect to a corresponding element* S_i *of a belief state* S *is given by*

$$\mathbf{US}_i(MCS, S) = \{head(r) | r \in br_i \text{ applicable in } S\},$$

where br_i *denotes the set of bridge rules of* S_i*'s corresponding context* C_i*.*

In general, a set of update functions may yield different results when the functions are applied multiple times or in different orders. We do not allow such sets of update functions.

Definition 11. *An applicable set of update functions* $\mathbf{US}_i(MCS, S)$ *is stationary for an input* inp_i *iff the following two conditions hold:*

- $\forall u \in \mathbf{US}_i(MCS, S) : u(inp_i) = u^m(inp_i)$ *for* $m \geq 1$ *(idempotency)*
- $\forall u, u' \in \mathbf{US}_i(MCS, S) : u(u'(inp_i)) = u'(u(inp_i))$ *(commutativity)*

Definition 12. *The update of a belief state element* S_i *of a belief state* S*, with respect to a set of update functions* $\mathbf{US}_i$ *with* k *elements, is given by*

$$u_1(u_2(\ldots u_k(inp_i)\ldots))$$

if $\mathbf{US}_i$ *is stationary for* inp_i*, and undefined otherwise.*

Please note that stationarity is only required for the set of update functions that is actually applied to belief state elements at a time. We now can give the definition of the generalized concept of equilibrium.

Definition 13. *A generalized belief state* $S = ((inp_1, res_1), \ldots, (inp_n, res_n))$ *of* M *is an equilibrium iff, for* $1 \leq i \leq n$*, the following condition holds:*

$$update(inp_i, \mathbf{US}_i) = inp_i \text{ and } res_i \in ACC(inp_i)$$

where $update(inp_i, \mathbf{US}_i)$ *denotes the update of* S_i *with respect to* US_i*, which in turn has to be stationary for the corresponding* inp_i*.*

The remainder of this section describes a procedure to compute all equilibria of a finite MCS, based on complete enumeration, which served as basis for our proof-of-concept implementation.

Definition 14. *An MCS* $M = (C_1, \ldots, C_n)$ *is said to be finite, iff for* $1 \leq i \leq n$, *following condition holds:* $|ACC(inp_i)| < \infty$ *and* $|br_i| < \infty$.

For the implementation, we consider finite MCS only.

Definition 15. *Let* $\mathbf{Br}$ *be a set of* n *bridge rules of an MCS. A bridge rule model is an assignment* $\mathbf{Br} \mapsto \{0,1\}^n$ *that represents for each bridge rule in* $\mathbf{Br}$ *whether it is active or not.*

Proposition 1. *For each equilibrium there is exactly one bridge rule model.*

For a given bridge rule model and an MCS we first apply all the bridge rules activated in the bridge rule model yielding $inp'_1 ... inp'_n$. Then we compute the set of results for each context i given inp'_i by applying $ACC(inp'_i)$, yielding a set of results res_i^j for each i, being of finite cardinality as MCS was said to be finite. Thus, testing whether (inp_i, res_i^j) is an equilibrium for all j, we obtain the set of equilibria for the given bridge rule model. Iterating the procedure over the (finite) set of all bridge rule models and joining the resulting sets of equilibria finally yields the set of all equilibria.

Definition 16. *Given an MCS with a (global) set of bridge rules* $br = \bigcup_i br_i$. *A set of bridge rules* $br_j \subseteq br$ *is called update-monotonic iff for all belief states S, S' the following condition holds:*

$$S' = update(MCS, S) \Rightarrow VC(MCS, S) \subseteq VC(MCS, S')$$

where

$$VC(MCS, S) = \bigcup_i \{cond_i \in R_i | cond_i(inp_i, res_i) = 1\}$$

and $update(MCS, S)$ *is the (global) update over all* $S_i \in S$.

As bridge rules in the update-monotonic subset of bridge rules of the MCS are guaranteed to remain active after any update, the update-monotonic bridge rules that are initially active in the MCS when searching for equilibria have to be active in any equilibrium. Hence, when iterating over all bridge rule models, only those bridge rule models that comply with the initially active update-monotonic bridge rules have to be considered.

Proposition 2. *For finite MCS, the above sketched algorithm (for a pseudo code representation see Algorithm 1) is both complete and correct.*

3 Refinements of the Basic Algorithm

In the following, we present two refinements – mainly used for reducing computational complexity – we added to the MCS algorithm from [4].

```
**Input**: $MCS = (C_1, \ldots, C_n)$ (a finite multi-context system)
**Output**: answer
answer $\leftarrow \{\}$
$br \leftarrow \bigcup_{1 \le i \le n} br_i$
$ums \leftarrow$ UpdateMonotonicBridgeRules(MCS,br)
$cand \leftarrow \{brm \in 2^{br} | ums \subseteq brm\}$
**for** $brm \in cand$ **do**
    $S \leftarrow ((inp_1, res_1), \ldots, (inp_n, res_n))$
    **for** $b \in brm$ **do**
        $S \leftarrow$Apply$(head(b), S)$

    **for** $1 \le i \le n$ **do**
        $S[i] \leftarrow$(inp($S[i]$), **ACC**$_i$(inp($S[i]$)))
    **for** $1 \le i_1 \le |$res$(S_1)|, \ldots, 1 \le i_n \le |$res$(S_n)|$ **do**
        $S^* \leftarrow$ ((inp($S_1$),res($S_1$)[$i_1$]),..., (inp($S_1$),res($S_1$)[$i_n$]))
        **if** Equilibrium($S^*$) **then**
            answer$\leftarrow$ answer$\cup\{S^*\}$
```

Algorithm 1: Algorithm for computing all equilibria of an MCS

3.1 The Notion of Conflicting Bridge Rules

The first refinement comprises a method for pruning out some bridge rule models (BRMs) already before they are falsely considered as possible equilibria.

We exploit the (within every context locally valid) demand for consistency in order to reduce the equilibria search space via "conflicting bridge rules":

Definition 17. *Two bridge rules $br_i, br_j \in br$ (where br is a set of bridge rules of an MCS) are called i-conflicting if applying br_i would directly inhibit the application of br_j due to a condition $c_{jk} \in body(br_j)$ which contains (where applicable) "$\neg head(br_i)$" or "not $head(br_i)$" (or any other statement contradicting $head(br_i)$).*

This allows for an extension of the algorithm: For a given set of bridge rules br, we can initially compute, for every $br_i \in br$, the corresponding set of i-conflicting bridge rules in br. As every active bridge rule $br_i \in br$ may directly inactivate all i-conflicting bridge rules, we may further reduce the number of BRMs tested on being an equilibrium by setting all i-conflicting bridge rules to 0, when setting br_i to 1.

The additional effort spent on initially computing the conflicting bridge rules within the set br in most cases pays off, as in total $\sum_{m=1}^{C} 2^{n-(1+m)}$ (n the total number of bridge rules of the MCS, C the total number of i-conflicting bridge rules for all $br_i \in br$) BRMs are pruned out. Hence, at least $(\sum_{m=1}^{C} 2^{n-(1+m)}) \cdot N$ (N the number of contexts of the MCS) reasoner calls less have to be performed. An upper boundary for the extra costs of computing the conflicting bridge rules within br is given by $n \cdot l$ (n the total number of bridge rules of the MCS, l the maximum of the numbers of conditions in the body of a bridge rule from br) comparisons between bridge rule heads and bridge rule conditions from bridge rules within the set br.[5]

3.2 Constraints on Bridge Rules

Moreover, the wish to reduce some complexity by making knowledge of the combinatorical structure of bridge rules (that is already implicitly contained in the MCS) explicit may arise. In order to do so, constraints on bridge rules or groups of bridge rules may be used, resulting in a formalism to impose constraints on the BRMs to be considered.

The constraint formalism shall offer possibilities to group bridge rules into classes and impose cardinality restrictions on these classes up to uniqueness of a rule as a representant of a certain rule class. It may be used to model interconnectedness and hierarchy between the classes of bridge rules by establishing a concept of subclasses and superclasses, as well as introducing complementary classes. Moreover, features have been added to provide some kind of rudimentary inference mechanism amongst bridge rules/bridge rule classes.[6]

1. $X.maxCard(numb)$: $class_1.maxCard(a)$, with $0 \leq a \leq max$, where max is the total number of bridge rules in the model, expresses that only BRMs in which at most a bridge rules from the class $class_1$ are active shall be considered.
2. $X.minCard(numb)$: Analogously.
3. $X.unique$: $class_1.unique$ expresses that only BRMs in which exactly one bridge rule from the class $class_1$ is active shall be considered.
4. $X.bundled$: $class_1.bundled$ expresses that only BRMs in which all bridge rules from the class $class_1$ are active, or BRMs in which all bridge rules from the class $class_1$ are inactivated, shall be considered.
5. $X.superClass(Y)$: $class_1.superClass(class_2)$ expresses that $class_1$ has $class_2$ as a superclass, i.e. $class_1 \subset class_2$.
6. $X.subClass(Y)$: Analogously.
7. $X.complementary(Y)$: $class_1.complementary(class_2)$ expresses that only BRMs in which bridge rules from $class_1$ or bridge rules from $class_2$, but not bridge rules from both classes at a time, are active shall be considered.
8. $X.oneOfImplicatesExactlyOne(Y)$: $class_1.oneOfImplicatesExactlyOne(class_2)$ expresses that only BRMs in which, whenever a bridge rule of $class_1$ is active, exactly one bridge rule out of $class_2$ is active, shall be considered.
9. $X.oneOfImplicatesSome(Y)$: Analogously.
10. $X.oneOfImplicatesAll(Y)$: Analogously.
11. $X.entireClassImplicatesExactlyOne(Y)$: $class_1.entireClassImplicatesOne(class_2)$ expresses that only BRMs in which, whenever the bridge rules of $class_1$ are all active, exactly one bridge rule out of $class_2$ is active, shall be considered.

[5] As in general the comparisons between heads and conditions can be performed by magnitudes more time-efficient as a reasoner call (e.g. in many logical contexts the comparison may be performed by a simple syntax based matching between heads and conditions), the overall computational efficiency of the MCS will be improved.

[6] In the following using the implemented constraint language, where X and Y are variables which can be replaced by concrete classes, and $numb$ is a numeric integer variable.

12. $X.entireClassImplicatesSome(Y)$: Analogously.
13. $X.entireClassImplicatesAll(Y)$: Analogously.

The individual bridge rules are explicitly distributed over the different classes by labeling them with the class name, e. g. directly in the bridge rule base (by this establishing some kind of $instanceOf$ relation between the individual bridge rules and the classes they belong to: the class being the concept, the bridge rule the instance). Every unlabeled bridge rule is implicitly added to a generic class of bridge rules ($class_{generic}$) which is the superclass of all other classes. On $class_{generic}$, no constraints in form of properties as listed above may be imposed.

4 A Proof-of-concept MCS Framework Implementation

The proof-of-concept implementation is based on the principles of the aforementioned algorithm for finding the equilibria of an MCS presented in [4]. After some steps of reading the MCS specification, getting input and creating the local representations of the knowledge bases and the sets of bridge rules of the MCS, we generate all possible BRMs. Then for each BRM we perform the updates it indicates and check whether the result is an equilibrium or not. When doing so we also perform a test if the conditions of all applied bridge rules are really fulfilled, and the bridge rules have been applied justifiedly. According to the result of this tests, we add the BRM to the list of equilibria BRMs, or we directly proceed with the next bridge rule model.

Inter alia for the implementation of the constraint mechanism for bridge rules and bridge rule classes, we make use of the lparse/smodels combination (see [5] and [6]) as implementation of the stable model semantics for logic programmes. Both are at some points called as external components.

4.1 Infrastructure & Ex Ante Setup

As programming language for the first implementation[7] of the MCS framework we used Scala, a general purpose programming language.

In our MCS framework, an MCS is unequivocally defined by an MCS configuration file. In the file all contexts are represented. Having read the configuration file and built an internal representation of each context, the MCS framework sets up a list of initial knowledge or input bases, as well as a list of bridge rule bases, each associated with a context.

Moreover, if indicated in the invocation, the bridge rule constraint configuration file is processed. This file contains information concerning the grouping of bridge rules into different classes, the properties of individual bridge rule classes and the relations between different classes (see Section 3.2). For doing so, the different bridge rule classes explicitely have to be declared as such in the file, afterwards their properties and the relations between the classes may be named.

[7] In the meantime, a second implementation using Clojure has been built.

If this functionality shall be used, in the bridge rule bases every bridge rule individually has to be labelled with the name of the class it belongs to. Unlabelled bridge rules by default will be treated as belonging to $class_{generic}$.

After performing these steps, the BRMs which afterwards will be tested for representing an equilibrium are generated. This may be done by creating a bitstring representing a BRM for each possible combination of activation/inactivation of bridge rules. Thereby, in total 2^n (n the total number of bridge rules in the MCS) bitstrings are generated.

Due to performance considerations, we implemented another mechanism, exploiting – whenever indicated and possible – constraints on bridge rules and conflicting bridge rules. In this scenario, we are using the lparse/smodels answer set solver to generate the BRM bitstrings. This offers the possibility to directly include the information given in the constraints on the bridge rules into the model generation: smodels is used for generating a complete list of models of bridge rule combinations complying with the constraints stated via the constraints on bridge rules formalism, which then are transformed into BRM bitstrings. By this, no a priori contradicting bridge rule combinations are considered possible BRMs. At least for logical contexts we may also use the notion of conflicting bridge rules, further sparsening the field of BRMs. The bridge rule head bases and the bridge rule body bases are scanned for conflicting structures, adding information of any conflict found to the input for the answer set solver.

To start the equilibria mechanism, the MCS's knowledge bases, the bridge rule bases and the BRMs are handed over to the "*findEquilibria*" subroutine.

"*findEquilibria*" returns pairs of equilibria and corresponding BRMs. The output may be written out directly, or postprocessing steps may be performed.

4.2 The Equilibria Mechanism: "*findEquilibria*"

Given the MCS's knowledge bases, the bridge rules, a list of BRMs and the contexts "*findEquilibria*" lists the equilibria amongst the BRMs and the corresponding belief states of the respective contexts.

Given a BRM, the heads and bodies of the corresponding bridge rules, the contexts and the knowledge bases, "*findEquilibria*" performs all the updates which are indicated by the heads of bridge rules set to 1 in the given BRM. This is done by a subroutine called "*createReasonerInput*". Then it invokes the corresponding reasoners (indicated by the information stored in the contexts), thereby performing a global reasoner call over all the (possibly updated) knowledge bases. If the reasoner calls yield consistent results for all knowledge bases, for every inactivated bridge rule "*findEquilibria*" has to test whether the bridge rule is correctly set to 0 in the BRM. To do so, it has to find in every inactivated bridge rule's body at least one condition that is not fulfilled. The same has to be done for all the bridge rules set to 1: All the conditions in every bridge rule body have to be fulfilled. If this tests may successfully be completed, the BRM represents an equilibrium for the given MCS, and the BRM, as well as the the belief state are linked and jointly added to the list of equilibria. Then, "*findEquilibria*" proceeds with the next BRM (if there is any).

As long as we are dealing with logical contexts for which we may use the lparse/smodels answer set solver as a reasoner, we may make the test for founded activation of the 1-bridge rules superfluous, by modifying the "*createReasoner-Input*" mechanism: For every knowledge base element a, demanded in a condition in the body of an activated bridge rule, we integrate a constraint ":- *not a*" into the answer set solver input (which in its result inhibits the generation of a model in that a is not present), and for every knowledge base element b, listed in an inhibitive condition, we add ":- b" (preventing the occurrence of models in which b is present). Thus, we assure that if a model is returned, for every 1-bridge rule the conjunction of the conditions in its body holds.

For the inactivated bridge rules this technique may not be applied: For a bridge rule set to 0 it is sufficient that only one of its conditions fails. Thus, doing as we did before with the 1-bridge rules would be by far too restrictive.

But nevertheless, as long as we are dealing with contexts allowing for smodels as a reasoner, and yielding unique reasoning results (i.e. only one model is returned when smodels is called as a reasoner), we may exploit some functionality of smodels when testing for founded inactivation of a 0-bridge rule: The following test will be based on comparing the total number of bridge rules in the MCS with the number of correctly activated/inactivated bridge rules. If both values are equal, the BRM represents an equilibrium of the MCS.

Given the valid results of the reasoner calls, we cycle through the representation of the contexts in the BRM. If a context's BRM part doesn't contain a 0, we may directly add the number of bridge rules of this context to the number of foundedly activated/inactivated bridge rules (founded activation has already been assured by the use of the smodels-optimization when creating the reasoner input for the context) and proceed with the next context's representation. Otherwise, we take the context's model, which was previously returned as result by the smodels call. For every inactivated bridge rule, we cycle through its body, and add every condition to a list corresponding to the conditions target context (e.g. for "$2:\ not\ q$" "*not q*" would be added to C_2's list). Having completely traversed the bridge rule's body, we add to every context's reasoning result a constraint, containing the conjunction of the elements of the context's list. E.g. for the list "$\{p,\ not\ q,\ r,\ s\}$" the constraint "$:-\ p,\ not\ q,\ r,\ s.$" would be added to the model previously returned by the smodels call. Having done so for all the contexts of the MCS, we perform another global reasoner call, invoking smodels for every single context, handing over the modified reasoning results of the earlier calls. If there is at least one context – with modified reasoning result as input – for which the second reasoner call returns a valid model (i.e. the conjunction constraint is not applied), we may augment the number of foundedly activated/inactivated bridge rules by one and proceed with the next bridge rule. If the models of all contexts with modified input are invalid, the 0-bridge rule has been inactivated unfoundedly, and the BRM does not represent an equilibrium.

Unfortunately, we have to perform this process for every single bridge rule, and cannot include all constraints from bridge rules into one global reasoner call, as bridge rules which "overlap" in their conditions (e.g. one containing "2 : *not s*", the other containing "2 : *s*') may otherwise cause wrong results.

4.3 Identifying Self-sustaining Equilibria

For MCS consisting only of logical contexts using smodels as a reasoner, we added "$findSstEquilibria''$, used to list the self-sustaining[8] (s-st.) equilibria.

The "$findSstEquilibria$" function is given a BRM, the knowledge bases, the bridge rules and the contexts. The "*createReasonerInput*" subroutine is called with empty knowledge bases (again performing the smodels-optimization already mentioned), and the result is handed over to the contexts' reasoners. If a reasoner returns an invalid (e.g. inconsistent) reasoning result, the BRM does not represent an s-st. equilibrium of the given MCS.

In order to eliminate all equilibria other than the s-st. ones, additional tests are performed on the reasoning results: For every bridge rule set to 1 in the BRM, a test is performed whether one of its conditions contains a "*not*"-statement. If this test yields a positive result, the found equilibrium is not an s-st. equilibrium, as its status as equilibrium is based on the absence of the knowledge base element contained in the "*not*"-condition. Then, a test for unfoundedly inactivated 0-bridge rules in the BRM has to be performed analogously as in "$findEquilibria$".

Finally, we add the content of the knowledge bases initially handed over when invoking "$findSstEquilibria$" to the belief state obtained as equilibrium, and perform another reasoner call for every context, assuring the consistency of the expanded belief state. If all reasoner calls yield a valid result, the equilibrium found is a s-st. equilibrium of the MCS. The BRM and the expanded belief state are linked and jointly added to the list of s-st. equilibria. Then, "$findSstEquilibria$" proceeds with the next BRM (if there is any).

4.4 Comments Concerning Algorithmic Complexity

As a measure for the complexity of the "$findEquilibria$" routine we may use the number of reasoner calls performed. Given an MCS with m contexts and n BRMs. In the worst case, at least $m \cdot n$ reasoner calls have to be performed. Therefore, in order to identify all equilibria of an MCS, in the worst case at least $m \cdot 2^k$ reasoner calls (k the number of bridge rules in the MCS) are needed.[9]

In real world applications, when all equilibria of an MCS ought to be found, the dominating element in this estimation is the number of bridge rules k. For the naive generate and test approach, in the most general case the number of $m \cdot 2^k$ reasoner calls is a hard lower boundary. All BRMs have to be tested, and in every test for every context a reasoner call has to be performed.

[8] We call an equilibrium self-sustaining iff the application of all bridge rules may only be based on the application of other bridge rules, and thus be independent of the concrete knowledge bases of the MCS.

[9] Always assuming that per context per BRM one reasoner call is in fact needed and is sufficient. Here, sufficiency may be assured for the different types of contexts of finite generalized MCS (as admissible for the algorithm presented in [4]).

For the "*findSstEquilibria*" mechanism, in the most general case $m \cdot (2^k - 1)$ is a hard lower boundary for the number of reasoner calls (m the number of contexts of the MCS, k the number of bridge rules). In the average case, the number of reasoner calls will be higher: For every BRM which in fact represents a s-st. equilibrium, m^2 reasoner calls have to be performed.

5 Conclusion

When applying the MCS framework to different testing examples, it has shown to be applicable and appropriate (see [7] and [8] for examples). For the future, more sophisticated techniques for reducing computational complexity will be needed. Nevertheless, the approach to implementing an MCS has proven to be effective, and an important step towards the implementation of an MCS framework has been made.

References

1. Roelofsen, F., Serafini, L.: Minimal and Absent Information in Contexts. In: International Joint Conference on Artificial Intelligence. Volume 19., Lawrence Erlbaum Associates LTD (2005) 558
2. Brewka, G., Roelofsen, F., Serafini, L.: Contextual Default Reasoning. Proc. IJCAI-07, Hyderabad, India (2007)
3. Brewka, G., Eiter, T.: Equilibria in Heterogeneous Nonmonotonic Multi-Context Systems. In: Proceedings of the National Conference on Artificial Intelligence. Volume 22., Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999 (2007) 385-390
4. Besold, T.R., Mandl, S.: Integrating Logical and Sub-Symbolic Contexts of Reasoning. In Filipa, J., Fred, A., Sharp, B., eds.: Proceedings of ICAART 2010 - Second International Conference on Agents and Artifcial Intelligence, INSTICC Press (2010) . For a full version of the paper see also http://www8.informatik.uni-erlangen.de/inf8/Publications/bridging_mcs_original.pdf.
5. Syrjnen, T.: Lparse 1.0 User's Manual. (2000)
6. Simons, P., Niemela, I., Soininen, T.: Extending and implementing the stable model semantics. Artif. Intell. 138(1-2) (2002) 181-234
7. Besold, T.R., Schiemann, B.: A Multi-Context System Computing Modalities. In: Proc. 23rd Int.Workshop on Description Logics (DL2010),Waterloo, Canada. Number 573 in CEUR Workshop Proceedings (2010)
8. Besold, T.R.: Theory and Implementation of Multi-Context Systems Containing Logical and Sub-Symbolic Contexts of Reasoning. Master's thesis, Department of Mathematics & Department of Computer Science 8: Artificial Intelligence, FAU Erlangen-Nuremberg (2009) . The full thesis is available under http://www.opus.ub.uni-erlangen.de/opus/volltexte/2010/1587/.

Useability versus Adaptation – Approaches to Higher Level Context Detection

Robert Lokaiczyk[1], Andreas Faatz[2], Benedikt Schmidt[2], and Matthäus Martynus[3]

[1] TU Darmstadt,
Rundeturmstr. 10
D-64283 Darmstadt, Germany**
[2] SAP Research,
Bleichstr. 8,
D-64285 Darmstadt, Germany
[3] SAP AG
Dietmar-Hopp-Allee 16
D-69190 Walldorf
Germany

Abstract. While the research in detection of context features - even cognitive ones like the user's goal - steps forward continuously, it is often focused on the quality of detection. But since context often finds use in context adaptive systems that proactively adapt to the user's need, research should always consider the resulting benefit of context for the user in any context-based application. This paper presents the results of a case study which a) evaluates algorithms to detect three subtypes of user goals in knowledge work and b) focuses on useability issues, that have a direct impact for a measurable improvement of the work results. Our work includes a quantitative evaluation like task completion time improvement and qualitative aspects (e.g. intrusiveness of the system). Finally, we draw conclusions from our measurements, especially on how modeling of an adaptive approach at the workplace might take place.

1 Introduction

While low level context features like location or temperature can be determined easily by appropriate sensors, higher level context is not an explicit state that could be measured by sensors. In personal information management systems or context-aware e-learning scenarios, such cognitive context features include the user's goal, which is in best case a conscious state of mind expressible by the user. Neither is it feasible to measure the user's goal with a physical apparatus, nor is it adequate to continuously ask the user for his/her goal. Actively interrupting the user during his/her work reduces the useability of a contextualized information or e-learning system. In most cases frequent pop-ups will immediately lead the user to a refusal of the overall software system, no matter how helpful the

** research conducted while the author was with SAP Research Darmstadt recommended resources or actions based on the context are. Useability overrules context-aware adaptation here.

Consequently, research should not only focus on unintrusive context detection with high precision and accuracy but also on adequate mechanisms in order to use this context information for applications in provable helpful way. Useability of context-aware applications might also suffer in many ways while reconfiguration of GUI elements or ever changing lists of recommendations might distract the user from the working task and actually lead to longer task completion times (efficiency) or even lower task completion ratios (effectiveness). As a conclusion context based systems should not only be benchmarked by the quality of their context detection mechanisms but primarily by the benefit that context information is gives the user.

In order to analyze effects of the design of context adaptive systems to useability and work results in a corporate knowledge work scenario we set up a user study with the goal to measure the interrelation between acceptance (i.e. usability) and efficiency (i.e. ability to speed up knowledge work) of a context detection environment. Therefore we sketch the connection between context and user goal in Section 2. define a taxonomy of user goals for knowledge work in Section 3 and introduce up-to-date approaches for unintrusive, probabilistic context detection in Section 4. We instantiate these approaches in a corporate user study that involves 12 typical tasks of knowledge work (see Section 5). The resulting improvement of the user's knowledge work is measured qualitatively and quantitatively in Section 5, before we draw conclusions more on a systems modeling level.

2 Connection between User Goal and Context

Context adaptive systems generate user support with respect to a user goal identified by the system. This demands an understanding of the connection between goal and context. To describe this connection, we use an extended k-system control-circuit model as presented by [1] (see Figure 1), originally used to describe system-world interaction. In our adaptation it shows the dominance of the user goal on user context mediated by action in and perception of the real world. A user might have multiple goals concurrently which have different relevancies. We consider the goal with the highest relevance as trigger for the organization of user-world interaction in a situation. Organization means that the goal leads to a planning process of the user, how to achieve a goal. The resulting plan as behavior in rehearsal guides user perception of and user action in the real world, as described in adaptive resonance theory [2]. Thus, user context is dependent on a goal and the resulting plan. A respective context term has been introduced by [3][4] and slightly modified by [5]. They distinguish the following categories of the real world for an individual:

– Intrinsic Context: Those elements of the physical world which are consciously perceived and considered by an individual as related to a goal

[4] referred to in [4]

– Extrinsic Context: Those elements of the physical world which are consciously perceived and not considered as related to a goal
– Unperceived things: Aspects of the real world which are not consciously perceived by an individual

Context is changed by actions and consumed by perception. That can result in adaptation of the plan and can again have an effect on the goal, which closes the modeled control-circuit. A context adaptive system interacts with the control circuit. It is an element of the real world which detects user actions based on sensors and deduces user goals based on collected sensor data. In a desktop environment this means that the actions of the user, like opening applications, using application-specific functionalities, etc. are indicators which are used to identify a user goal. Once, the system identifies a user goal three different kinds of support can be given:

– support the perception of the intrinsic context (e.g. highlight specific elements of the context to support user orientation)
– support actions on elements of the intrinsic context (e.g. automation of time consuming and stereotype activities)
– extend the intrinsic context by adding elements to the conscious and goal related user perception (e.g. recommendation lists)

To identify goals and decide on reasonable support a thorough understanding of user goals and their effects on the interaction with the real world is necessary. In the following we describe the understanding of user goals we followed to realize a context adaptive system.

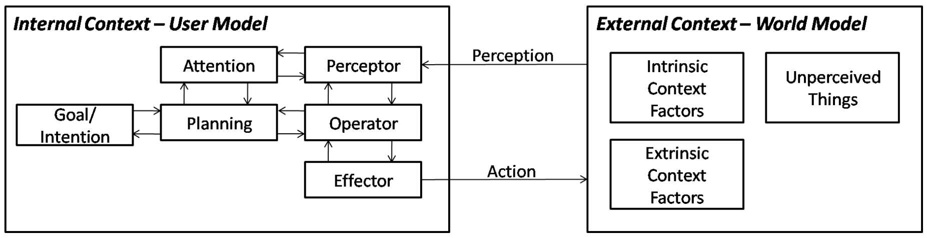

Internal Context – User Model
Attention
Perceptor
Goal/ Intention
Planning
Operator
Effector
Perception
Action
External Context – World Model
Intrinsic Context Factors
Unperceived Things
Extrinsic Context Factors

Fig. 1. Connection between User Goal and Context

3 Modeling User Goals

In accordance to Broder [6], who defined a taxonomy of user goals for web retrieval, we sub-divide the concept of user goal into three different subtypes including:

1. Informational Goal

2. Transactional Goal
3. Navigational Goal

While all subcategories are intentions the user might have during his/her work, the characteristics and the methodology to detect these goals differ.

In certain work situations the knowledge worker needs factual knowledge in order to fulfill his/her working task. An *informational goal* addresses background knowledge that is necessary to understand concepts with regard to the working task. If for example the knowledge worker needs to model a software with UML, the understanding of the concept UML class diagram is crucial to complete the task. The search for proper knowledge resources, that explain a class diagram, embodies an informational goal.

The user goes after a *transactional goal* if s/he is trying to accomplish a working task in a work process by performing certain transactions. Those transactions might involve user interaction with the system that result in a defined work result (e.g. a text document or a spreadsheet). The detection of a transactional goal consist of the analysis of the user interaction with the system in order to reason about the anticipated work task. A work task might be writing a letter, creating a balance sheet or compiling a presentation.

A *navigational goal* however is not characterized with regard to content aspects, but does target on a navigation path to a state of location in the work environment. The user's intent here is to find a particular document, directory or file, s/he already used. This involves also web resources like URLs the users has visited sometimes in the past. The resource might be of interest for him/her as a template or as an example. The navigational goal represents a description of the location of the object of interest via a path or a URL.

Since the three types of user goals cover different aspects of knowledge work, the algorithms to anticipate them vary too. The next section describes probabilistic approaches to reason for the supposed goal of the user by analyzing context features of the work environment (for a detailed enumeration of our context features see [7]). All approaches target to manage to get along with low explicit user input (like it would be the case with online learning or feedback mechanisms), in order not to spoil the useability. We strongly believe the user is not willing to accept additional effort to support a system that is meant to support him/her without an overall benefit.

4 Approaches to User Goal Detection

4.1 Informational Goal

In order to estimate the user's need for information in a certain work situation, the system needs to have means of interpretation of the topic the user is currently dealing with. The scientific field of topic detection offers a number of options given a textual corpus that can be analyzed. In distinction to topic detection in computational linguistics, where the input consist solely of one text document or fragment, we define the combination of all textual context features of the work environment as input for the topic detection. This involves not only documents currently opened in different applications, but also the content of websites displayed in the browser, window titles and file names. Recent methods for topic detection that deliver good results include e.g. LSA[5] [8]. But since we extract topical information only for the purpose to identify relevant words or concepts the user might have a question or informational need for, we focus on simple keyword-based approaches here. The extracted keywords relevant to the user's current working task represent potential informational goals the user needs information on. Based on the list of keywords a knowledge management system or work-place embedded help system might offer resources that explain the concept or define the keyword. In a generic scenario the learning resources can be derived simply by offering the corresponding Wikipedia page to the keyword, in specific corporate scenarios a corporate knowledge repository, reflecting professional needs is recommended. The relevance of a keywords with regard to the working task can be estimated by term relevancy measures as in [9] or given by a static list of relevant terms defined by a domain expert. In a similar fashion to the APOSDLE approach [10] we applied string matching on a list of keywords characteristic for a particular task. But in contrast to the APOSDLE approach the tagged learning material was automatically drawn from Wikipedia, i.e. referring to pages with the respective tags.

4.2 Transactional Goal

Task detection as a research category, which is used for the identification of a transactional goal, has already been described in [11], [12] and [7]. We also proceed with a machine learning approach that uses user interaction with the system and the work environment itself as an indication for a particular work task, since after a short training phase which can also be outsourced, the system works autonomously. Therefore we operate on slices of the event stream captured by desktop sensors. Our context model here is a holistic one with regard to the number of features that can be captured on the computer desktop itself, not in the physical environment around it. We apply a hybrid voting approach between the decision tree ID3 [13], Naïve Bayes [14], Euclidean distance [15], Irep [16] and SMO(128) [17] algorithms in this experiment. This outperformed the single algorithms named on our data set.

4.3 Navigational Goal

The identification of the navigational goal is the task of identifying the next document to be opened by the user. If we anticipate the document right and provide a short link the user saves time for navigation and search. We formulate this problem scientifically as sequential prediction and leverage from recent research in clickstream analysis (see [9]). Without any background models we create a navigational graph in-time that consists of documents accessed as nodes

[5] Latent Semantic Analysis

and transitions between two documents as vertices in the graph. A sample graph from our user study is shown in Figure 2. The detection of the matching document is then based on a sequence of documents in the navigation path, which was recently used by the user. On this graph we can use partitioning algorithms and propose all navigation objects in the actual partition that have not been accessed in the actual session. Phase 1 of our user study showed the most effective results for navigational goal detection, which are based on the paradigm of spreading activation [18] on such a navigational graph.

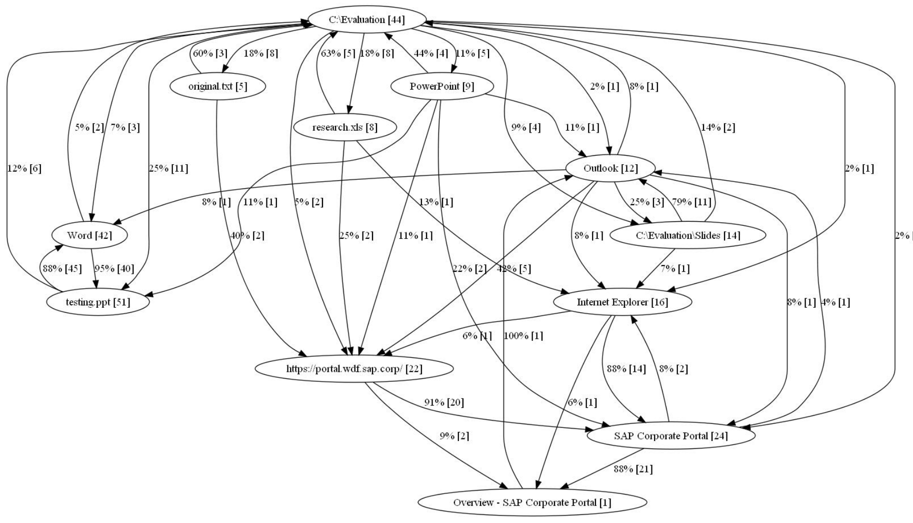


Fig. 2. Navigational graph enabling algorithms without task or domain model

5 Design of the User Study

The following paragraph describes the general setup of our in-house context detection study in the corporate environment of a large software company with dominant characteristics of knowledge work in daily business. Our general research question addresses not only the accuracy of context detection itself but also the interrelation between acceptance (i.e. usability) and efficiency (i.e. ability to speed up knowledge work) of the described context detection environment. The experiment consisted of two phases:

In phase 1 we collected training and evaluation data. We used these data to improve our system and find the best machine learning algorithms and parameters for the recommendation algorithms. In phase 2 we tested the improved system.

Our report will sketch these phases, discuss the measurements and draw conclusions for the introduction of context detection to an organization - in general, along task models and along domain ontologies characterizing the organization.

Phase 1

We defined 12 tasks, which have been performed by 20 participants from the SAP Research CEC Darmstadt (mainly postdoc-level researchers and PhD candidates) during the first phase without any contextualized support.

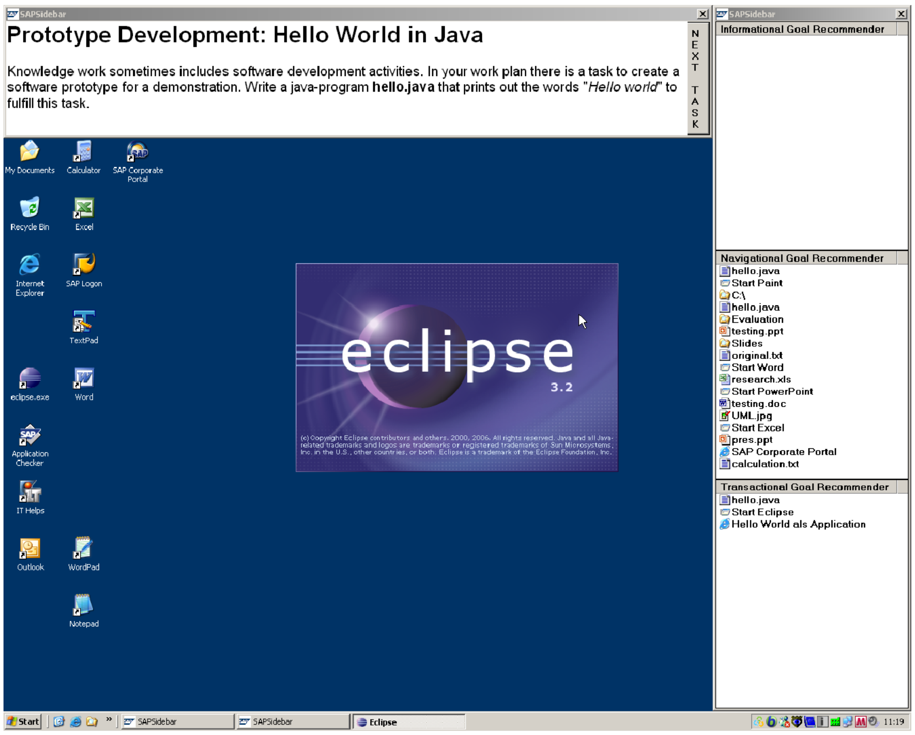
The tasks are condensed versions of typical work a researcher has to carry out in the context of the industrial research projects, transfer projects and program activities at SAP Research in Darmstadt (comments in italics):

1. create a presentation on Generics in Java *(i.e. preparing technical slides)*
2. leave request *(interacting with a typical SAP tool)*
3. update the SRN *(the SAP Research Knowledge Representation Tool)* page of your project
4. distribute presentation slides *(find the right people and their full names to send slides to)*
5. visualization of quantitative research results *(MS Excel-style)*
6. translation of executive summary *(a typical task as SAP is a bilingual company)*
7. code development: Hello World class in Java *(very simplified programming task)*
8. create a handout *(for a presentation)*
9. create a UML-diagram *(very simplified)*
10. budget calculation *(no tool pre-nominated)*
11. software update *(non-automatic software update of one tool)*
12. inventory update *(modeled as an interruption of another task)*

To avoid correlations between the tasks the participants got the tasks in a random order. During the data collection phase these tasks were conducted in a restricted time and with some hints on supporting material (e.g. the presentation slides). We collected the data in a database and labeled it according to the task in which it was collected.

Phase 2

In this phase we let 15 participants from the SAP Research CEC Darmstadt (a true subset of the phase 1 participants) perform the tasks a second time in random order. This time they were supported by our context detection system (see Figure 3). This happened weeks after the first phase to blur the participants memory on how the tasks are performed - a quite realistic condition with support from our system. In comparison to recent approaches (e.g. APOSDLE P3) we used a very simplified user interface (see the screenshot below) only showing documents stemming from task detection, topic detection and clickstream analysis to collect information about how our system helps them to fulfill the task faster and easier. By design, the users in phase 2 had as easy access to the tools for the tasks (left side, for instance MS Excel) as in phase 1 to principally enable work without context detected support.


Prototype Development: Hello World in Java

Knowledge work sometimes includes software development activities. In your work plan there is a task to create a software prototype for a demonstration. Write a java-program **hello.java** that prints out the words "*Hello world*" to fulfill this task.

Informational Goal Recommender

Navigational Goal Recommender
- hello.java
- Start Paint
- C:\
- hello.java
- Evaluation
- testing.ppt
- Slides
- original.txt
- Start Word
- research.xls
- Start PowerPoint
- testing.doc
- UML.jpg
- Start Excel
- pres.ppt
- SAP Corporate Portal
- calculation.txt

Transactional Goal Recommender
- hello.java
- Start Eclipse
- Hello World als Application

Fig. 3. Screenshot of the simplified UI - the right hand sidebar showing three potential recommended list of documents from topic detection (with a topic as informational goal), purely unsupervised mechanisms based on the users clickstream (a document or URL as a navigational goal) and task detection (with a task as transactional goal).

5.1 Evaluation and Discussion

We used a connection of the users actions to a system clock to measure the amount of time needed per task in phase 1 and phase 2. Figure 4 shows a detailed time analysis in the form of and the relative amount of time gained (green) or lost (red) with/without context adaptive support.

The average user became 30% faster on an average task. Intuitively, this number is more than the speed-up expected without tool support by a pure learning effect with scrambled, once interrupted short tasks resembling a heavy workload. But the proof via a control group that tool support is the reason for speed-up is still missing.

The two most effectively tasks speeded up were routine tasks (Update SRN and Software Update) with relatively low involvement of personal creativity. However, despite the fact that these were routine tasks, their repetitivety is low enough to still offer enormous room for automated support as in our study, e.g. by just presenting the right entry point to the system to be updated. Tasks like Visualization Results and the ones on Prototype Development or Translation

	Generics	Update SRN	Distribute Slides	Visualization Results	Leave Request	Translation	Prototype Development	Inventory Update	Handout	UML	Budget Calculation	Software Update	Complete
User 1	-21,05%	-70,31%	36,42%	59,05%	-61,63%	33,33%	-36,76%	-0,85%	-31,76%	-39,92%	-32,75%	-69,38%	-29,96%
User 2	13,59%	-60,63%	-8,89%	158,20%	24,38%	32,52%	28,13%	-4,19%	105,93%	22,20%	33,93%	-39,89%	14,97%
User 3	-55,59%	-62,33%	17,65%	-20,14%	32,45%	-1,74%	-25,12%	-24,14%	-58,91%	5,31%	-10,97%	-7,33%	-18,91%
User 4	-50,00%	-51,56%	-53,40%	41,22%	-28,30%	-31,00%	-18,57%	-51,75%	-26,24%	-49,53%	-35,41%	-51,56%	-33,69%
User 5	-53,55%	-55,81%	-35,03%	-35,27%	-39,01%	-54,22%	-58,88%	-56,33%	-31,30%	-50,72%	66,67%	-55,17%	-42,78%
User 6	-61,19%	-74,88%	-11,97%	-59,88%	-29,05%	-23,87%	-24,51%	-22,97%	-17,65%	-33,05%	-2,67%	-26,83%	-32,13%
User 7	-50,48%	-93,52%	-63,70%	-10,33%	45,71%	-25,65%	4,22%	-57,58%	-53,72%	-36,73%	23,60%	11,22%	-31,92%
User 8	5,83%	-84,77%	-29,53%	-78,69%	-47,00%	-10,39%	-55,81%	2,16%	-10,42%	-29,54%	-35,14%	-48,55%	-41,62%
User 9	-66,88%	-33,11%	-6,02%	-9,50%	4,33%	-0,48%	39,08%	-6,15%	-79,15%	-31,79%	-4,03%	-53,47%	-25,14%
User 10	-30,60%	-55,26%	-62,41%	-34,72%	-10,90%	-47,62%	12,39%	-46,19%	3,83%	-59,27%	-35,78%	-54,82%	-32,17%
User 11	-5,29%	-81,25%	-16,99%	-40,67%	-34,84%	-42,79%	33,97%	-5,11%	-38,39%	32,76%	-42,66%	-45,00%	-29,77%
User 12	-55,65%	-87,85%	-51,92%	-49,76%	-57,36%	-29,05%	-65,16%	-59,71%	-75,80%	-54,53%	-59,42%	-76,53%	-58,82%
User 13	-67,00%	-82,46%	-71,63%	-3,88%	43,42%	-9,74%	13,22%	-41,52%	-14,38%	-38,44%	-62,46%	2,58%	-31,98%
User 14	-26,97%	-47,92%	-55,06%	-3,61%	24,22%	29,72%	-36,50%	-2,82%	-10,26%	-30,43%	-19,18%	-62,17%	-20,00%
User 15	-37,77%	-45,13%	-7,14%	-29,46%	-37,33%	-62,14%	-54,94%	-60,16%	-30,70%	-6,83%	-22,44%	-60,66%	-36,32%
Average	-37,51%	-65,79%	-27,97%	-7,83%	-11,39%	-16,21%	-16,35%	-29,15%	-24,60%	-26,70%	-15,91%	-42,50%	-30,02%

Fig. 4. Speed measurements, relative speed-up (green) or slow-down (red) of users

involve more creativity, but still show the supportability by context detection in our lab setting. There was one user (User 2), who was very much distracted by the recommendation system - at least in a positive sense, as s/he liked the system very much and started playing with it. This attitude seems to be an outlier in comparison to the other fourteen users. Another outlier, the leave request with relatively many users taking more time to perform with context detection support, lacks explanation so far. None of the 15 users found particular bad suggestions by the system regarding this task - and even 43% of the users found particularly good recommendations in that task.

Figure 5 shows the overall satisfaction with the combination of tools as determined by a questionnaire.

A surprising result shown at the top of Figure 5 is the overall opinion about the intrusiveness of the system. 57% of the users did not feel intrusiveness, a reason to follow to the presentation of documents and not tasks and topics, such as in APOSDLE P3 [19] and similar systems as the Microsoft Office Assistant. As the design of the UI came nearly by chance and just piggy-bagged of the scientific approach to make the three different flavors of context detection (navigational, transactional, informational) comparable for the lab study, we see a very interesting anchor for further UI design around the smoothly morphing list of document hits updated by the context detection regularly. Further research in that direction drawing a distinction from the current metaphors "pop-up of suggestions" or "click to get a context-based suggestion" seems to be very valuable. Although the system was not felt to be very intrusive (or at least by a minority of the users perceived as such), a strong majority (93%) of the users were aware of the fact, that different kinds of context detection were designed to support them. The recommendation of documents from the spreading activation algorithm turned out to be most valuable from the users perspective. We also

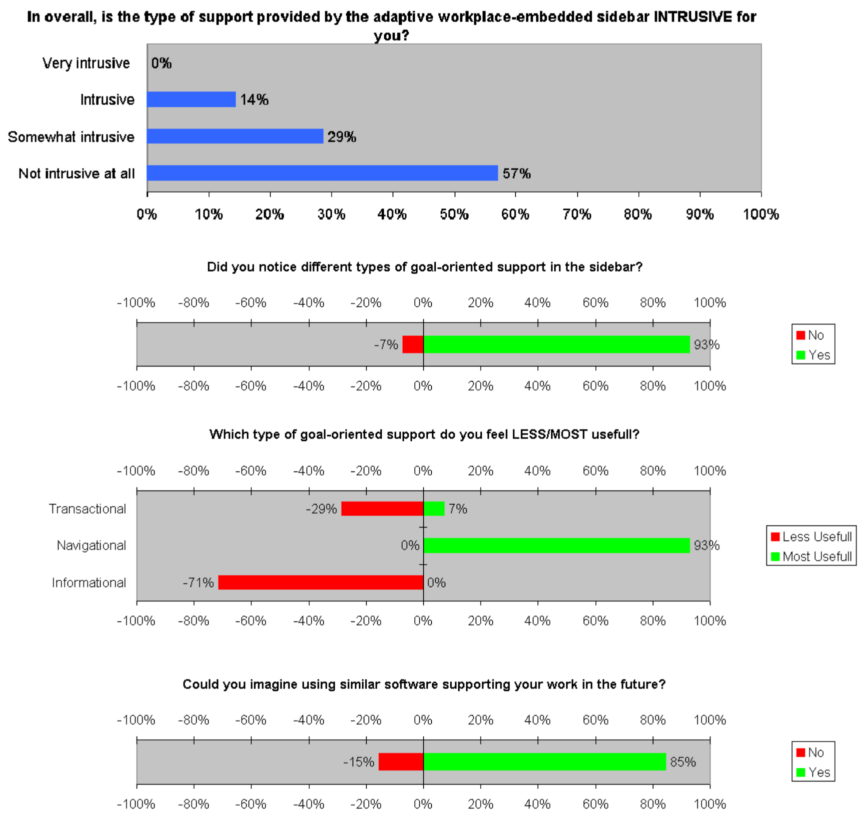


Fig. 5. Results of the questionnaires

related this question to the click behavior, which showed that these documents from the navigational mechanism were also selected most often - in contrast to a mediate selection (and a medium judgement in the questionnaires) of documents from the task detection and almost no response to the topic detection. This is a clear distinction from the results in the summative evaluation of APOSDLE, where the topic detection was favored above the task detection. The outcome depends on the quality of the models and annotation. In this experiment, the linking to Wikipedia was perceived as too general even for "non-office" tasks, where a source of encyclopedic knowledge might be expected as useful. The other way around, navigational mechanisms do not depend on models, supported the users in our study - but have the conceptual drawback of only referring to documents, which were already at least touched by the users (i.e. triggering the question, how new knowledge can be transported to the individual user, a question of true knowledge transfer in the organization).

Finally, the system was perceived as helpful, which we conclude from the combination of the two facts that it caused speed-up and that most of the users claimed, that they would use it again. The questions which remain open are less on the algorithmic side, but more on the organizational side and UI side:

– How to effectively and efficiently connect models to the context detection and how to complement this with the more personal navigational mechanisms?
– Which UI-metaphor is best suited for presenting the continuous flow of context-dependent suggestions?
– Does the possible extension of defining and adopting holistic enterprise models compensate the heavy modeling effort?

6 Summary and Outlook

We identified the best context detection algorithms in a specific empirically accessible workplace environment. Our study included task detection, topic detection and model-free, purely statistically determined navigational context detection. Contextualized support seems to be helpful along with the right metaphor for the UI but - when going beyond the helpful reconstruction of personal document streams and piles - strongly dependent on the investment in the modeling of tasks and topics (i.e. in transactional and informational goals). Thus the main follow-up of our study is, that we are now in the position to suggest a way of context detection intertwined with modeling, which slightly deviates from the strategy taken in e.g. APOSDLE integrated modeling [20]. The key is human activity monitoring: We suggest to apply context detection already in the very early phases of task and topic modeling by navigational mechanisms individually and to complement this by task models and ontologies, which in some sense (e.g. by pooling search terms passed to search engines) are seeded decentrally in the organization. The purpose is to focus modeling on situations, where we can automatically determine, that navigational context detection passes back values (e.g. documents), which are from the personal history, but do not reflect the user's goal. Such a gap analysis should determine the shape of the domain and task model. This differs from a modeling strategy of interviewing experts and opens the door to a more decentralized and continuous way of relating detection mechanisms and modeling. However, the knowledge engineer and the domain expert are still be needed to work on the seed we mentioned as a result of human activity monitoring and the later phases of validation. In our upcoming work, we will focus on the category of transactional goals. We want to use user action data of multiple users to identify reoccuring subcategories of the transactional goal. For these subcategories we want to provide identification mechanisms and realize specific, proactive user support. This will help to get a better understanding of the difference and similarity of transactional goals of multiple users.

References

1. Stachowiak, H.: Allgemeine Modelltheorie. Springer-Verlag (1973)

2. Grossberg, S.: The link between brain learning, attention, and consciousness. Causality, Meaningful Complexity and Embodied Cognition (1999) 3–45
3. Hewitt, K.: Context effects in memory: A review. Unpublished manuscript. Cambridge University, Psychological Laboratory. (1977)
4. Tiberghien, G.: Il CONTEXT AND COGNITION: INTRODUCTION. Revue bimestrielle publiée avec le concours des Universités de Provenœ et Aix-Marseille II **6**(2) (1986)
5. Öztürk, P., Aamodt, A.: A context model for knowledge-intensive case-based reasoning. International Journal of Human Computer Studies **48** (1998) 331–356
6. Broder, A.: A taxonomy of web search. SIGIR Forum **36**(2) (2002) 3–10
7. Lokaiczyk, R., Faatz, A., Beckhaus, A., Görtz, M.: Enhancing just-in-time e-learning through machine learning on desktop context sensors. In Kokinov, B.N., Richardson, D.C., Roth-Berghofer, T., Vieu, L., eds.: CONTEXT. Volume 4635 of Lecture Notes in Computer Science., Springer (August 2007) 330–341
8. Landauer, T.K., Dumais, S.T.: A solution to Plato's problem: The latent semantic analysis theory of the acquisition, induction, and representation of knowledge. Psychological Review **104** (1997) 211–240
9. Lokaiczyk, R.: Verfahren kontextbasierter Nutzerzielanalyse. PhD thesis, Universität Leipzig (2009)
10. APOSDLE consortium: Deliverable d6.3 - use cases & application requirements 2 (second prototype). Technical report (2008)
11. Shen, J., Li, L., Dietterich, T.G.: Real-time detection of task switches of desktop users. In Veloso, M.M., ed.: IJCAI. (2007) 2868–2873
12. Granitzer, M., Kröll, M., Seifert, C., Rath, A.S., Weber, N., Dietzel, O., Lindstaedt, S.N.: Analysis of machine learning techniques for context extraction. In Pichappan, P., Abraham, A., eds.: ICDIM, IEEE (2008) 233–240
13. Quinlan, J.R.: Induction of Decision Trees. Machine Learning **1**(1) (1986) 81–106
14. Dumais, S.T., Platt, J.C., Hecherman, D., Sahami, M.: Inductive Learning Algorithms and Representations for Text Categorization. In Gardarin, G., French, J.C., Pissinou, N., Makki, K., Bouganim, L., eds.: Proceedings of the 1998 ACM CIKM International Conference on Information and Knowledge Management, Bethesda, Maryland, USA, November 3-7, 1998, ACM (1998) 148–155
15. Agrawal, R., Faloutsos, C., Swami, A.N.: Efficient similarity search in sequence databases. In: FODO '93: Proceedings of the 4th International Conference on Foundations of Data Organization and Algorithms, London, UK, Springer-Verlag (1993) 69–84
16. Fürnkranz, J., Widmer, G.: Incremental Reduced Error Pruning. In: Proceedings the Eleventh International Conference on Machine Learning, New Brunswick, NJ (1994) 70–77
17. Platt, J.: Fast Training of Support Vector Machines using Sequential Minimal Optimization. MIT Press (1998)
18. Anderson, J.R.: A spreading activation theory of memory. In Collins, A., Smith, E.E., eds.: Readings in Cognitive Science: A Perspective from Psychology and Artificial Intelligence. Kaufmann, San Mateo, CA (1988) 137–154
19. Lindstaedt, S.N., Ley, T., Mayer, H.: Integrating Working and Learning with APOSDLE. In: Proceedings of the 11th Business Meeting of Forum Neue Medien, 10-11 November 2005, Vienna, Verlag Forum Neue Medien (2005)
20. Rospocher, M., Ghidini, C., Serafini, L., Kump, B., Pammer, V., Lindstaedt, S., Faatz, A., Ley, T.: Collaborative enterprise integrated modelling. In: Proceedings of the 5th Workshop on Semantic Web Applications and Perspectives (SWAP2008), Rome, Italy, December. (2008) 15–17

Movement Recognition using Context: a Lexical Approach Based on Coherence

Alessandra Mileo[1], Stefano Pinardi[1], and Roberto Bisiani[1]

Department of Informatics, Systems and Communication, University of Milan-Bicocca, viale Sarca 336/14, I–20126 Milan

Abstract. Movement recognition constitutes a central task in home-based assisted living environments and in many application domains where activity recognition is crucial. Solutions in these application areas often rely on an heterogeneous collection of body-sensors whose diversity and lack of precision has to be compensated by advanced techniques for feature extraction and analysis. Although there are well established quantitative methods in machine learning for robotics and neighboring fields for addressing these problems, they lack advanced knowledge representation and reasoning capacities that may help understanding through contextualization.

Such capabilities are not only useful in dealing with lacking and imprecise information, but moreover they allow for a better inclusion of semantic information and more general domain-related knowledge.

We address this problem and investigate how a lexical approach to multi-sensor analysis can be combined with answer set programming to support movement recognition. A semantic notion of contextual coherence is formalized and qualitative optimization criteria are introduced in the reasoning process. We report upon a first experimental evaluation of the lexical approach to multi-sensor analysis and discuss the potentials of knowledge-based contextualization of movements in reducing the error rate.

1 Introduction and Motivations

Movement recognition is an important aspect of situation assessment both in Ambient Intelligence and Healthcare applications [1]. It is also important in order to forecast critical situations like a fall or a stroke, to understand emotional patterns from position of the body [2, 3], to track activities [4, 5]. It is important also for gait and posture analysis, human computer interaction, and in motion recognition and capture [6, 7, 4, 8].

It is possible to use video cameras for movement classification, but with a video camera you have to segment body from the background, identify body parts, solve luminance and hidden parts problems, and target the monitored person when more people are present in the area. Video movement analysis is a useful technique in hospital but it can hardly be used in a day by day analysis to classify movements in any natural condition like real sport analysis, healthcare applications, medical or social surveillance protocols [9–11].

It is possible to recognize the movements of a person using wearable inertial sensors, as shown by various studies and applications [12, 13, 1, 4, 5, 14]. Wearable sensors usually contain inertial devices like accelerometers and gyroscopes to detect linear and torque forces: they can be placed on different segments of the body to analyze movements. Some sensors also use magnetometers in order to identify the north-pole direction to determine Euler angles in respect to a fixed reference system [8]. Also, sensors can be connected to a wireless network in order to facilitate the collection of data [13].

Many different ways to use inertial sensors for movement classification have been described in the literature; they are mainly based on Machine Learning techniques [4]. Some researchers use sensors placed on a single spot of the body [1], others use many sensors positioned on different segments of the body [12, 1, 4], others use a multimodal approach using both microphones and inertial sensors [6, 15].

The advantage of using inertial sensors for movement recognition are patent. We know with absolute certainty which segment of the body data come from. We do not have to solve hidden surfaces problems or identify the correct person in a crowded situation. Also inertial sensors can be used in any natural situation and are more respectful of privacy. On the other hand, movement classification using inertial sensors is still an open subject of research and needs to be well understood both in the field of data analysis with machine learning techniques and in the reasoning area.

The rationale is that these methods are useful to create a “lexicon” of movements: the proposed methods have significant error rates, and use small vocabularies [4]. But movements are not only isolated events, they have a “lexical context”, are causally and logically connected, and are space dependent.

We are interested in classifying movements with a Machine Learning approach aimed to create a rich user-independent vocabulary, and in exploiting our knowledge of legal sequences of movements as a pattern to reason about movements in order to validate or reject one or more actions in a given scenario.

Our modeling and reasoning technique is based on Answer Set Programming (ASP), a declarative logic programming framework, combining a compact, flexible, and expressive modeling language with high computational performance. The knowledge-based approach provides a more flexible way to reason about semantically-annotated sequences, compared to pure quantitative approaches.

We will use a new proposed Machine Learning method for “lexical analysis” that has a good accuracy 95.23% - 97.63% [12]. Our methodology is inspired by machine learning techniques used for information retrieval and text mining [16], with some adaptation.

On top of this analysis, a further level of knowledge-based validation is in charge of reasoning about meta-patterns of sequences as well as contextual constraints to reduce error rate.

The ASP based reasoning process follows the common generate and test methodology in (i) generating the space of legal patterns according to semantic validation and (ii) exploring the search space by applying efficient solving techniques to compensate for possible errors by enforcing constraint satisfaction and optimization.

The lexical analysis of features for classification of movements is described in Section 2. Section 3 introduces the ASP formalism and presents our modeling and reasoning strategies, while a preliminary evaluation of the potentials of our approach is presented in Section 4. A short discussion follows in Section 5.

2 Movements Classification: from Sensors to Activity

2.1 Sensors and Features Extraction

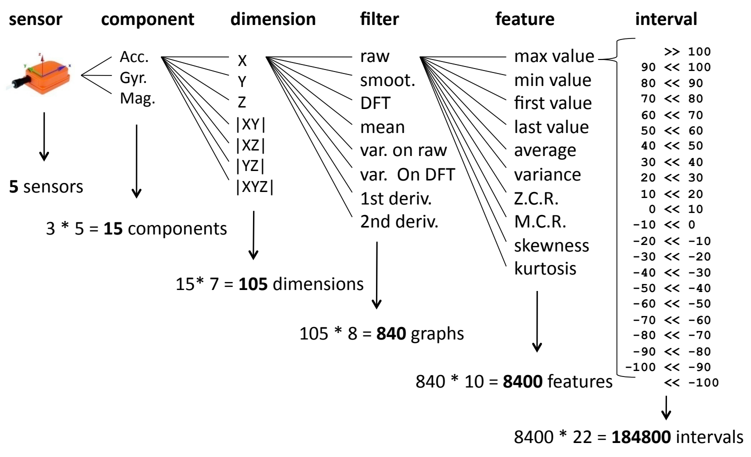
We used five MTx inertial sensors produced by XSens [8], these sensors have three different devices mounted on board: a 3D accelerometer, a 3D Gyroscope, and a 3D magnetometer. Informations are represented as a three dimensional vector in the sensors reference frame providing data about linear forces (ms/sec^2), torque forces (rad/sec) and earth-magnetic field intensity (direction) (milli-Tesla). Even if it is possible to represent vectors in a geo-referenced fixed frame with the MTx sensors, we preferred this configuration because many technologies do not have a fixed frame reference system, and we want to study the problem in the more generic and technologically neutral situation.

In order to create a flexible mechanism to classify movements, we extract very generic features that are not dependent on the application domain. Every sensor component - the accelerometer, gyroscope and magnetometer - returns one information for every spatial dimension (X, Y, Z). Every component information has also been considered in its planar norm representation ($|XY|$, $|XZ|$, $|YZ|$) and in its 3D norm representation ($|XYZ|$), for a total of 7 data per sensor. These data have been filtered using eight functions (null, smoothing, low pass, mean, variance, variance with low pass, first derivative, second derivative) generating 840 transformations of the original data. Then, ten generic features have been chosen (Maximum value, Minimum value, First Sample, Last Sample, Mean, Variance, Zero Crossing Rare, Mean Crossing Rate, Skewness, Kurtosis) for a total number of 8400 features. Finally, features have been quantized roughly into 20 intervals (see Figure1). At the end, every action generates a sparse binary vector of 184800 dimensions. This vector is used to create the classification pattern of movements that constitutes the Lexicon or Dictionary of the application scenario.

2.2 Features Analysis

All features do not have the same relevance depending on population and lexical contexts: some features are more frequent within the population, others can be more or less spread inside the given vocabulary of actions. To transform these qualitative considerations in a quantitative measurement we introduced two weights: the FF (Feature Frequency) and IVFF (Inverse Vocabulary Feature Frequency). Feature Frequency takes into account distributions of the feature per class in the population, as shown in Equation 1:

Fig. 1. Feature extraction process. Actions are transformed in a binary sparse vector of 184.400 values.



$$FF_{ij} = \frac{n_{i,j}}{|P|} \tag{1}$$

where $n_{i,j}$ is the number of occurrences of the feature σ_i in the action a_j and $|P|$ represents the cardinality of the population.

Inverse Vocabulary Feature Frequency weights features according to their discriminatory ability within the dictionary of actions as shown in Equation 2:

$$IVFF_i = log\frac{|A|}{|a : \sigma_i \in a|} \tag{2}$$

where $|A|$ is the cardinality of the dictionary, and $|a : \sigma_i \in a|$ represents the number of actions in which feature σ_i assumes the same value.

Every feature is weighted by multiplying FF and IVF as shown in Equation 3:

$$W_{i,j} = FF_{i,j} * IVFF_i \tag{3}$$

2.3 Vocabulary and Classification

Actions are feature vectors placed into a feature-actions matrix transformed with the weighting operation described in Equation 3. The action to be recognized is a n-dimensional vector weighted using this transformation. This action is considered a *query* in the feature-action space. Through a set of similarity algorithms the most similar action is calculated, then results are compared with the ground truth. We used three similarity algorithms: Ranking, Cosine Similarity and Euclidean Distance defined, respectively, as follows:

$$rank_j = \sum_{i=1}^{n} W_{i,j} \tag{4}$$

$$dist_i = \sqrt{\sum_{i=1}^{n}(W_{i,j} - q_{i,j})^2} \tag{5}$$

$$cos\theta = \frac{W_{i,j} * q_{i,j}}{|W_{i,j}||q_{i,j}|} \tag{6}$$

where $W_{i,j}$ represents the weight of the σ_i interval of action a_j of the Training-Set, and $q_{i,j}$ is the IVFF value associated to the feature of the query.

Every action to be classified, hit all features in one of the intervals; every interval for each action in the vocabulary, is associated to a weight for that feature interval in the action, as defined in Equation 3. The rank of an action is given by the sum of weights associated to the intervals of all features interested by the action. The higher the sum of weight, the more similar the action to be classified is to the action in the vocabulary.

We used a Leave One Out Cross Validation (LOOCV) method to test the accuracy of the recognition on two different databases: an internal database NIDA 1.0 with 273 samples, and a public database WARD 1.0 with 1270 samples, obtaining preliminary results illustrated in Section 4.

The accuracy of our method is high when we use five sensors on different parts of the body: we reached an accuracy of 95.23% on NIDA 1.0 and 97.74% on WARD 1.0 outperforming the results presented in the literature on WARD database [12, 13]. There are many situations where it is not possible to wear many sensors on the body for social acceptance or comfort, for example with ill or elder people in health care scenarios. If we use a single sensor accuracy decreases: with a single sensor on the hip the accuracy rate is 81.31%, with only one sensor on the right wrist we have an accuracy of 82.78%, and using just the right ankle we have an accuracy of 83.15%. In these situations the classifier makes errors in almost all the actions (see 3), and the error rate is higher on some specific actions reducing the general performance. We want to improve accuracy reducing the error rate even in the single sensor scenario, and the logic-based contextual inference can help in doing this in a flexible and performant way.

3 Knowledge-Based Support to Movement Recognition

3.1 ASP Basics

We assume the reader to be familiar with the terminology and basic definitions of ASP (see [17] for details). In what follows, we rely on the language supported by grounders *lparse* [18] and *gringo* [19], providing normal and choice rules, cardinality and integrity constraints, as well as aggregates and optimization statements. As usual, rules with variables are regarded as representatives for all respective ground instances.

3.2 Model of Movement

The logical formalization of our model of movements is provided in terms of three aspects of body motion (referred to as *classes*), each of them characterized by a set of values and one or more additional attributes for that value.

The current[1] list of values and attributes for each class are summarized in Table 1. Please note that attributes related to values of class *posture* indicate where the posture is assumed to be held, while for values of the other classes, the attribute specifies an additional description of how the movement is performed.

Class	**Value**	**Attribute**
posture	sit, stand	{chair, bed}
	lay	{bed}
motion	walk	{forward, upstairs, downstairs, fast}
	jump	{once}
	open	{circular, sliding}
	kick	{frontal, lateral}
heading	right, left	{90, 180}

Table 1. Aspects of body motion included in our vocabulary

Each tuple $< Class, Value, Attribute >$ we define, corresponds to a *word* of the vocabulary of movements introduced in Section 2.

Whenever a new *word* is classified by the underlying mechanism illustrated in Section 2, the knowledge-based representation associates a time step to the logical classification of the action, in a predicate of the form:

$$sensed(Class, Value, Attribute, TimeStep)$$

Extending the vocabulary is a straightforward activity in our model, because it can be done by extending the range of possible attributes of a value, adding new values or adding new classes. The *state* of body motion is determined by three tuples of the form $< Class, Value, Attribute >$, one for each of the three classes, at the same time step T.

[1] In this preliminary analysis we reduced the classes of movements we want to reason about, and their values, in order to better illustrate the reasoning principles via examples.

In order to validate tuples identified by the underlying classification method, we introduce two additional properties for the movement recognition problem: the *semantic distance* measure between to subsequent tuples of the same class, and the *state coherence* of a tuple of a given class, with respect to tuples of the other classes at the same time step. It is worth mentioning that we apply the inertia law on tuples across time steps.

Semantic Distance between two tuples $X =< Class, Value_1, Attribute_1 >$ and $Y =< Class, Value_2, Attribute_2 >$ represented by predicate $dist(X, Y, N)$, is the minimum number of semantically meaningful transitions N that can lead from X to Y.

State Coherence for a tuple $X =< Class, Value, Attribute >$ is the number of (ground) state constraints that are violated at a given time step T by assuming that the identification of X is correct.

Considering our initial vocabulary, semantic distance is mainly concerned with tuples of the form $< posture, V, A >$ because for the other classes, each sequence of values is admitted, therefore we have that the semantic distance is equal to one for every possible sequence of triples where $Class \in \{heading, motion\}$. Distances between triples that are related to class *posture* are summarised in Table 2.

Source Tuple	Target Tuple	Condition	Distance
$< posture, P, V >$	$< posture, P_1, V >$	$P_1 \neq P$	1
$< posture, P, V >$	$< posture, P, V_1 >$	$V_1 \neq V$	2
$< posture, lay, V >$	$< posture, sit, V_1 >$	$V_1 \neq V$	2
$< posture, lay, V >$	$< posture, stand, V_1 >$	$V_1 \neq V$	3
$< posture, sit, V >$	$< posture, X_1, V_1 >$	$(V_1 \neq V) \vee (X_1 \neq sit)$	3

Table 2. Distances between tuples

As for state coherence, we define a set of constraints that are violated for some combination of triples at a given time step. Let us consider our reduced vocabulary of body movements described in Table 1, the state constraints we define express the following concepts:

– a tuple of the form $< posture, sit, A_p >$, for any attribute A_p[2] in the domain of *sit* is not coherent with tuples of the form $< motion, V_i, A_m >$ for any attribute A_m in the domain of the correspondent V_i and $V_i \in \{walk, open\}$;

[2] Note that in our notation, upper-case names refer to variables and have to be instantiated over their domain.

- in a similar way, a tuple of the form $< posture, lay, A_p >$, for any attribute A_p in the domain of *lay* is not coherent with tuples of the form $< motion, V_i, A_i >$ for any attribute A_i in the domain of the correspondent V_i and $V_i \in \{walk, jump, open\}$.

In the next section we illustrate how to reason about contextual coherence to validate actions recognised by the lexicographic classification of movements. Plausible sequences of actions are selected via optimization.

3.3 Movement Recognition: a Contextual View

The contextual validation of activities (or movements) is based on the definition of properties of the activity identified by the underlying classification process. In our solution, we identified two properties whose combination can determine a direct validation or the need for a change in the classification.

Shortest Distance property at a given time step T considers a tuple A_{t-1} that has been validated at time $T-1$ and verifies whether the new tuple A_t to be validated at time T is in the set of possible tuples having distance 1 from A_{t-1}.

State Coherence property at a given time step T for a tuple A_t holds whenever all (100%) of the state constraints are satisfied by the validation of A_t[3].

When the vocabulary is extended or new sensor information is introduced, we can easily re-define or extend the list of properties. An example can be the introduction of localization information in the definition of coherence: to take such information into account, we just have to introduce additional constraints to be satisfied for the *state coherence* property.

As mentioned earlier in this section, a classification represented by a tuple $A_t =< Class, Value, Attribute >$ can be associated to

- a *valid* status, identified by the fact that the logic predicate $valid(A_t)$ holds for A_t;
- a *switched* status, identified by fact that logic predicate $switched(A_t, A_t^{'})$ holds for A_t, given that A_t has been identified at time step T ($sensed(A_t, T)$) but it has been switched to the movement identified by tuple $A_t^{'}$;
- an *incomplete* status, identified by the fact the the logic predicate $incomplete(A_t, A_t^{'})$ holds for A_t, given that A_t has been identified at time step T ($sensed(A_t, T)$) but a classification is missing between $A_t^{'}$ and A_t;

Each classification for A_t can verify one, both or none of the contextual properties used for validation at a given time step T.

[3] Note that this is a simplified version of our formalization. In a more flexible version of the reasoning process we want to consider different level of coherence given by the percentage of constraits that are violated. This would allow to introduce further optimization that will be discussed in a future extended version of this paper.

In the trivial case, $dist(A_{t-1}, A_t, 1)$ holds and A_t is state-coherent: the classification is validated and $valid(A_t)$ becomes true.

Otherwise, we need reasoning to support the validation of the following situations:

1. $dist(A_{t-1}, A_t, 1)$ holds and A_t is not state-coherent;
2. $dist(A_{t-1}, A_t, N)$ holds for $N > 1$ but A_t is state-coherent;
3. $dist(A_{t-1}, A_t, N)$ holds for $N > 1$ and A_t is not state-coherent;

The easiest way to validate the classification of A_t in situation **1.**, is to provide acceptable sets of $switched(B_t, B_t^{'}),\ B_t \neq A_t,\ s.t.\ A_t$ becomes state-coherent.

In the other situations, $dist(A_{t-1}, A_t, N)\ with\ N > 1$, and we have to deal with the following two sub-cases:

a. $\exists\ A_t^{'}$ s.t. $dist(A_{t-1}, A_t^{'}, 1)\ \vee\ A_t^{'}$ is coherent at time T
   $\Rightarrow$ A_t is an erroneous classification and plausible switches $switched(A_t, A_t^{'})$ are derived such that $valid(A_t^{'})$ becomes true;
b. $\nexists\ A_t^{'}$ s.t. $dist(A_{t-1}, A_t^{'}, 1) \vee\ A_t^{'}$ is coherent at time T
   $\Rightarrow$ a classification A^* between A_{t-1} and A_t could be missing, $incomplete(A_t, A_t^{'})$ is derived and we can be in one or both of these scenarios:
   - $A^*,\ A_t$ are made state-coherent by switching appropriate tuples $B_t,\ B \neq A$, such that $valid(A_t)$ becomes derivable
   - an error in the classification of A_t is assumed and state-coherence is computed by switching A_t to $A_t^{'}$ such that $valid(A_t^{'})$ becomes derivable.

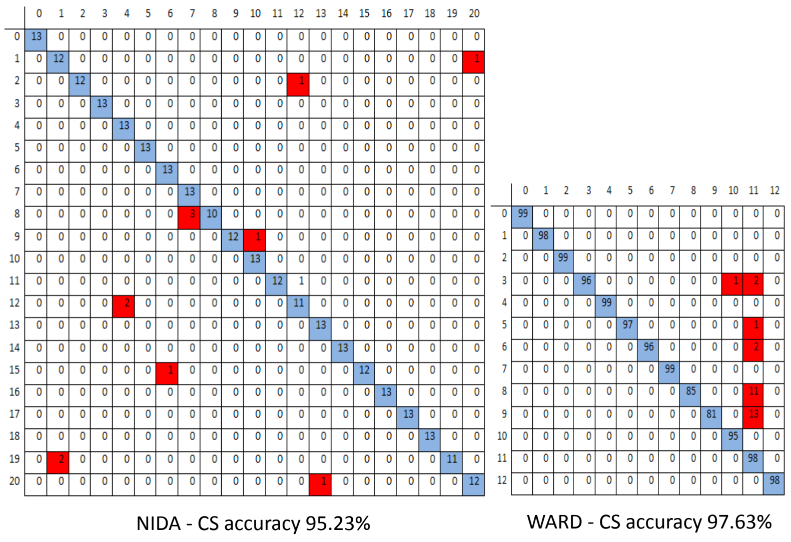
ASP inference is based on a generate-and-test approach. Plausible classifications for an action are generated using a *cardinality constraint* and then checked for *state coherence* and *shortest distance.* If the classification provided by the underlying mechanism described in Section 2 is among them, the system validates it. Otherwise, a different solution is proposed by switching one or more of the movements in the sequence, or by supposing that the sequence is incomplete and that some classifications are missing, or both, resulting in a lot of possibilities. In a similar scenario, default reasoning, non-determinism, choice rules, constraints and optimization via preferences plays a key role in devising effective deduction strategies. For lack of space, we cannot illustrate how all these constructs are used in the ASP encoding. A simple optimization criteria is based on the global minimization of switched tuples, i.e. we prefer to assume incomplete sequences rather than wrong classifications when this lead to a solution, but we can easily change our preference in a declarative way.

4 Evaluation Phase

In this section we illustrate results of preliminary tests on the identification of movements as non contextualized *words* of a *body lexicon.* Althought our machine learning process gives acceptable results, once we reduce the number of sensors, the error rate increases and misleading classifications cannot be identified. We believe that the knowledge-based contextualization of sequences of movements described in Section 3 can help reducing misinterpretations, although we need further testing to estimate percentual reduction of error rate.

4.1 Test Methods

Fig. 2. Confusion Matrix for Cosine Similarity (CS) with five sensors on NIDA and WARD databases.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	1
2	0	0	12	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0
3	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	3	10	0	0	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	0	0	0	0	0	0	0	12	1	0	0	0	0	0	0	0	0
12	0	0	0	0	2	0	0	0	0	0	0	0	11	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0	0
15	0	0	0	0	0	0	1	0	0	0	0	0	0	0	0	12	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	0	0	12

NIDA - CS accuracy 95.23%

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	99	0	0	0	0	0	0	0	0	0	0	0	0
1	0	98	0	0	0	0	0	0	0	0	0	0	0
2	0	0	99	0	0	0	0	0	0	0	0	0	0
3	0	0	0	96	0	0	0	0	0	0	1	2	0
4	0	0	0	0	99	0	0	0	0	0	0	0	0
5	0	0	0	0	0	97	0	0	0	0	0	1	0
6	0	0	0	0	0	0	96	0	0	0	0	2	0
7	0	0	0	0	0	0	0	99	0	0	0	0	0
8	0	0	0	0	0	0	0	0	85	0	0	11	0
9	0	0	0	0	0	0	0	0	0	81	0	13	0
10	0	0	0	0	0	0	0	0	0	0	95	0	0
11	0	0	0	0	0	0	0	0	0	0	0	98	0
12	0	0	0	0	0	0	0	0	0	0	0	0	98

WARD - CS accuracy 97.63%

At first we created a Test-Set of twenty-one different actions called NIDA 1.0. (Nomadis Internal Database of Actions). The NIDA 1.0 database contains movements acquired by the NOMADIS Laboratory of the University of Milano-Bicocca. These acquisitions have been obtained using 5 MTx sensors positioned on the pelvis, on the right and left wrist, and on the right and left ankle. NIDA includes 21 types of actions performed by 7 people (5 male and 2 female) ranging from 19 to 44-years-old, for a total of 273 actions. The complete list of actions is the following:

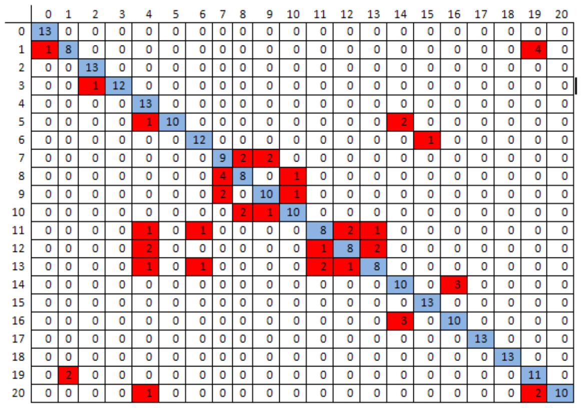
1. Get up from bed. 2. Get up from a chair. 3. Open a wardrobe. 4. Open a door. 5. Fall. 6. Walk forward 7. Run. 8. Turn left 180 degrees. 9. Turn right 180 degrees. 10. Turn left 90 degrees. 11. Turn right 180 degrees. 12. Karate frontal kick. 13. Karate side kick. 14. Karate punch. 15. Go upstairs. 16. Go downstairs. 17. Jump. 18. Write. 19. Lie down on a bed. 20. Sitting on a chair 21. Heavily sitting on a chair.

We also tested our methodology on a public database called WARD 1.0 (Wearable Action Recognition Database) created at UC-Berkeley [12, 13]. Acquisitions have been obtained positioning 5 sensors on the pelvis, on the right and left wrist, and on the right and left ankle. Each sensor contains a 3-axial accelerometer and a 2-axial gyroscope; magnetometers are not present. WARD contains 13 types of actions performed by 20 people (7 women and 13 men) ranging from 20 to 79-years-old with 5 repetition per action, for a total o 1200 actions. The list of actions can be found in [13].

4.2 Preliminary Analysis

We used a Leave One Out Cross Validation (LOOCV) method to test the accuracy of the proposed method. The accuracy of the algorithms for the NIDA database are: Ranking 89.74%, Euclidean Distance 95.23%, Cosine similarity 95.23%. The accuracy of algorithms using the WARD database are: Ranking 97.5%, Euclidean Distance 97.74%, Cosine 97.63% . We show the results of the Cosine Similarity also in a synoptic way using a Confusion Matrix (see Figure 2). In the columns the ground truth, in the rows the output of our classification algorithm. Label definitions are given in the above paragraph.

Fig. 3. Confusion Matrix for Cosine Similarity on the NIDA database. Accuracy: 81.31% using only one sensor on the hip.



	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20
0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	1	8	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	4	0
2	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	1	12	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	13	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
5	0	0	0	0	1	10	0	0	0	0	0	0	0	0	2	0	0	0	0	0	0
6	0	0	0	0	0	0	12	0	0	0	0	0	0	0	0	1	0	0	0	0	0
7	0	0	0	0	0	0	0	9	2	2	0	0	0	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	4	8	0	1	0	0	0	0	0	0	0	0	0	0
9	0	0	0	0	0	0	0	2	0	10	1	0	0	0	0	0	0	0	0	0	0
10	0	0	0	0	0	0	0	0	2	1	10	0	0	0	0	0	0	0	0	0	0
11	0	0	0	0	1	0	1	0	0	0	0	8	2	1	0	0	0	0	0	0	0
12	0	0	0	0	2	0	0	0	0	0	0	1	8	2	0	0	0	0	0	0	0
13	0	0	0	0	1	0	1	0	0	0	0	2	1	8	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0	0	0	0	0	0	0	10	0	3	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3	0	10	0	0	0	0
17	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	13	0	0
19	0	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	11	0
20	0	0	0	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	2	10

5 Discussion

We presented a hybrid approach to movement recognition by combining and extending standard quantitative methods in a knowledge-based yet qualitative framework. To this end, we took advantage of the knowledge representation and reasoning capacities of ASP for providing semantic contextualization.

Although we have not discussed the encoding in detail, it allows for easy customization and extensibility to a richer "lexicon" of movements. All in all, the contextual support of the high-level ASP specification makes the major difference of our approach to potential alternatives, and it seems hard to envisage in a purely quantitative settings. This preliminary work is only a starting point.

Future work will have to address more extensive and systematic experiments in various simulated as well as real scenarios. The problem of segmentation of a sequence of movements needs to be taken into account in order to evaluate the true scalability of our approach.

References

1. Merico, D., Mileo, A., Pinardi, S., Bisiani, R.: A Logical Approach to Home Healthcare with Intelligent Sensor-Network Support. The Computer Journal (2009) bxn074
2. Kleinsmith, A., Bianchi-Berthouze, N.: Recognizing affective dimensions from body posture. In: ACII. (2007) 48–58
3. Castellano, G., Villalba, S.D., Camurri, A.: Recognising human emotions from body movement and gesture dynamics. In: ACII. (2007) 71–82
4. Bao, L., Intille, S.S.: Activity recognition from user-annotated acceleration data. In: Pervasive. (2004) 1–17
5. woo Lee, S., Mase, K.: Activity and location recognition using wearable sensors. IEEE Pervasive Computing **1** (2002) 24–32
6. Lester, J., Choudhury, T., Borriello, G.: A practical approach to recognizing physical activities. In: Pervasive. (2006) 1–16
7. Dessere, E., Legrand, L.: First results of a complete marker-free methodology for human gait analysis. In: IEEE EMBC 2005. (2005)
8. (http//www.xsens.com/)
9. Cameron, J., Lasenby, J.: Estimating human skeleton parameters and configuration in real-time from markered optical motion capture. In: AMDO. (2008) 92–101
10. Okada, R., Stenger, B.: A single camera motion capture system for human-computer interaction. IEICE Transactions **91-D**(7) (2008) 1855–1862
11. Moeslund, T.B., Granum, E.: A survey of computer vision-based human motion capture. Computer Vision and Image Understanding **81**(3) (2001) 231–268
12. Pinardi, S., B.R.: Movement recognition with intelligent multisensor analysis, a lexical approach. (2010) To appear.
13. Yang, A.Y., Jafari, R., Sastry, S.S., Bajcsy, R.: Distributed recognition of human actions using wearable motion sensor networks. Ambient Intelligence and Smart Environments (2009) To appear.
14. Mntyjrvi J, Himberg J, S.T.: Recognizing human motion with multiple acceleration sensors. (2001) 747–752
15. Lester, J., Choudhury, T., Kern, N., Borriello, G., Hannaford, B.: A hybrid discriminative/generative approach for modeling human activities. In: In Proc. of the International Joint Conference on Artificial Intelligence (IJCAI. (2005) 766–772
16. Gerard Salton, A.W., Yang, C.: A vector space model for information retrieval. Journal of the American Society for Information Science **18**(11) (1975) 613–620
17. Baral, C.: Knowledge Representation, Reasoning and Declarative Problem Solving. Cambridge University Press (2003)
18. Syrjänen, T.: Lparse 1.0 user's manual. (http://www.tcs.hut.fi/Software/smodels/lparse.ps.gz)
19. Gebser, M., Schaub, T., Thiele, S.: Gringo : A new grounder for answer set programming. In: LPNMR. (2007) 266–271

A Unifying Framework for Situation Identification Methodologies

Olga Murdoch and Paddy Nixon

CLARITY: Centre for Sensor Web Technologies,
University College Dublin,
Ireland
olga.murdoch@ucdconnect.ie,paddy.nixon@ucd.ie

Abstract. Situation identification methodologies in pervasive computing aim to abstract low level context data into more meaningful high level contexts for use by context-aware application developers and users. Many situation identification techniques have been developed and successfully applied to limited scenarios. It is not possible to apply a single technique to a wide range of diverse applications. An ideal solution would allow the combination and and interchanging of techniques as appropriate. We propose a unifying framework for existing situation identification methodologies that will automatically select the best techniques for an application.

1 Introduction

Pervasive computing involves building highly responsive and self-adaptive systems which aim to increase the integration of technology into the fabric of everyday living [1] e.g., smart spaces. Context in pervasive systems comprises any information that characterises the situation of any person, place, or object that is considered relevant to the interaction between a user and an application [2]. Context data may be gathered from many sources including sensors deployed in the environment (sensing light or sound levels for example), digital sources such as calendar information, and static information such as a person's name, date of birth, etc. Sensors are inherently uncertain, providing an approximation of the real world rather than truely reflecting the variables they sense. They also may become damaged leading to inaccurate readings. This means that any action we decide to take may be wrong or may be invalidated by new information [3].

Individually, context data does not yield much information about the environment but when accumulated gives rise to higher-level contexts such as events, activities and situations. A higher-level context is the interpretation of context data, i.e., it describes a high level of abstraction in the environment in which the sensors are deployed. We refer to this process as situation identification. There is no general solution to the problem of situation identification for pervasive systems. Existing research provides us with an insight into how situation identification techniques can perform with high accuracy for limited scenarios [7–9] but it is not possible to utilise a single technique for a broad range of diverse problems [4–6].

Research on situation identification methodologies for contextual systems is still in its early stages. The relationship between techniques and their performance given diverse context-aware applications is little understood. We propose a decision framework for the selection of situation identification techniques by determining the characteristics of distinct scenarios for context-aware applications and how those characteristics relate to the performance of situation identification techniques. This will form the basis for a design time automatic selection tool whereby the most suitable situation identification techniques will be chosen for the context-aware application developer based on a set of application specific inputs.

We discuss related work in section 2. In section 3 we characterise applications and describe their relationship to existing situation identification techniques. We present the architecture for our framework in section 4. We conclude with a summary of this research and an outline future work.

2 Related Work

2.1 Situation Identification Methodologies

Situation identification methodologies abstract low level context data into more meaningful high level contexts. Many techniques have been developed and successfully applied to specific scenarios.

A **Bayesian network** (BN) is a directed acyclic graph that can learn the relationships between contexts. It is a probabilistic model allowing higher level contexts to be inferred based on prior and conditional probabilities. Leaf nodes in the graph represent sensed contexts and internal nodes and the root node represent higher-level inferred contexts. BN's have been used for device location and activity recognition [12, 7] .

Neural networks have been used as a way to learn a mapping between context data and higher level abstractions. They comprise a network of weighted values and transfer functions used to derive output values. While they have been proved useful in identifying links between contexts and abstractions such as events or activities, they use a "black-box" learning process which is hidden from the user/application and developer. Applications of Neural networks include user location detection [13] and activity recognition [8].

In **Support Vector Machines** (SVM) classification is performed by construction of an N-dimensional hyperplane that seperates training samples. Qian et. al., [14] use SVM's to identify activities using video data.

Case Based Reasoning (CBR) is a way of solving new problems based on previous similar solved problems. Each case consists of a problem (set of features describing current state of environment) and a solution (correctly identified situation). CREEK is a case based reasoning architecture which has been used to identify situations of hospital staff [10, 11] . Knox et al. [15] used CBR to identify situations in a home environment.

Hidden Markov Models (HMM) are probabilistic models consisting of hidden and observable variables at specific time-steps. A HMM is a finite set of hidden states whose transitions are governed by transition probabilities and each state is associated with probability distribution. An observable outcome/evidence can be generated from a hidden state according to its probability distribution. HMMs have been used to infer user situations from wearable sensors [9] and for recognising activities in the home [23].

Decision trees model decisions and consequences in order to predict an outcome based on a set of input variables. They have been applied to activity detection in [20] and [21].

Discussion Each of the described methodologies have been developed and tested by researchers focusing on specific application areas. Despite techniques successfully identifying situations in many of the evaluations performed, the same technique may not perform as well given another application setup. Also, techniques are not directly comparable unless identical evaluation setups and datasets are used [17]. As previously discussed, it is not possible to take one technique and solve the situation identification problem for a diverse range of applications. An ideal solution would allow us to interchange techniques on demand. Current research does not give a decisive description of what each technique is best at and what application characteristics they are best suited to.

2.2 Middleware and Frameworks

The fact that there is no one general solution to situation identification for context-aware applications has been taken into account in some systems where multiple techniques are employed. GAIA [4, 24] is middleware for context-aware application development that supports the use of fuzzy logic, probabilistic logic and Bayesian networks. A development framework is provided within which an application developer can define rules and build a BN according to their needs. Similarly the CAMUS middleware [5, 25, 26] supports the use of Bayesian, ontological and rule-based reasoning about context data. Dargie [6] discusses the benefits of incorporating multiple probabilistic situation identification techniques. Making multiple techniques available to a developer is very beneficial since one single technique will not meet every developers needs. Without decision aids the context-aware application developer chooses situation identification techniques based on intuition and experience. This is not an ideal solution.

Przybilski et. al., [18, 19] propose a reasoning framework for context-aware applications. They suggest an architecture encorporating multiple situation identification (reasoning) techniques in order to support a more diverse range of applications. The focus of the work presented however, is on the distribution of such a framework.

Discussion Research in situation identification does not identify the links between techniques and how and why they perform differently for diverse applications. A method for the automated selection of optimal situation identification techniques for applications has not, to our knowledge, been implemented or proposed. Based on our research in the area an automated selection framework for situation identification methodologies is needed but such a tool has not yet been developed. We propose a unifying framework for existing situation identification methodologies. We define the important characteristics of context-aware application scenarios and use their relationship to the performance of situation identification techniques as a basis for automatic technique selection. This will allow a unifying framework to determine the situation identification technique(s) that are best suited to the characteristics of a diverse range of applications.

3 Understanding the Relationship Between Applications and Optimal Situation Identification Methodologies

To automatically select situation identification techniques we must first understand the characteristics of context-aware applications. We must identify the important characteristics that differentiate between applications and those that will have an impact on how a situation identification technique performs. By analysing evaluations of situation identification techniques over a range of scenarios we can begin to understand the links between scenario characteristics and technique performance.

3.1 Application Characteristics

Previous research has catalogued context-aware applications in terms of the types of context used (activity, identity, location, time) and context-aware features of the application (presentation, automatic execution, tagging) [2]. For the purpose of this research and based on analysis of existing context-aware applications and datasets we provide classifications on two levels, physical deployment and specific application scenarios. We choose this level of separation as the physical deployment is relevant to all aspects of the application. On the other hand, applications can often be split into independent tasks whose characteristics differ greatly, so it is necessary to characterise such scenarios individually.

First however, we define types of context used in context-aware applications. There have been other classifications used but in our version we expand on that of Dey et. al., [2]. As well as identity, location and time, we also include device, environment, and user-profile. In our classification we have left out activity since this is a more abstract concept. We use device to describe contexts related to specific objects, for example when state change sensors detect the use of household appliances. By environment we mean the information sensed such as temperature and light, and by user-profile we refer to information gathered about the user from sources such as calendars, emails, or instant messengers.

Deployment Characteristics We define the important characteristics of an application's physical deployment. The number of sensors, types of sensors, number of users, and duration of annotated dataset (if any) differ greatly between

Dataset	Size	Sensor Types	Users	Dataset
PlaceLab [22]	900+	heterogeneous	1	120h
CASL [15]	15	heterogeneous	3	5d
Van Kasteren [23]	14	State change	1	28d

Table 1. Application deployment characteristics

applications and deployments. This can be seen in table 1 where we characterise three diverse datasets. While it may seem obvious how these characteristics differ between applications, they were also selected on the basis that they influence the performance of situation identification techniques. For example, the size of the deployment may lead to scalability issues for some situation identification techniques.

Scenario Characteristics We also need to further dissect the application in terms of the scenarios involved. A smart office application, for example, may rely on the identification of high level situations such as meeting, presentation, or coffee break. The contexts required for identifying the different situations in an application vary. This can also be said for the same situation across different applications where the physical deployment differs. We refer to these high level situations and their application specific characteristics as scenarios. We define the important characteristics of scenarios as frequency, duration, regularity, and required contexts (Time, Location, Identity, Device, Application). We illustrate this classification in table 2, using four scenarios that occur in the PlCouple1PlaceLab dataset [22].

Scenario	Frequency	Avg. Duration	Est. Regularity	Req. Contexts
Eating	197	1.58 mins	4 times per day	kitchen sensors, location
Hygeine	38	3.05 mins	2 times per day	bathroom sensors, location
Watching TV	22	33.29 mins	none	tv sensor, location
Using Computer	132	19.24 mins	3 times per day	computer sensor, location

Table 2. Scenario characteristics example using Plcouple1 PlaceLab dataset

3.2 Relating Situation Identification Performance to Application Characteristics

We have identified the evaluations by Knox et. al., [15, 16] as being one of the most comprehensive in terms of comparing the performance of techniques using diverse datasets. They evaluated the performance of Case Based Reasoning (CBR), Naive Bayes (NB), Suppor Vector Machines(SVM), and C4.5 Decision Trees (DT) in situation identification using three differing datasets. The PlaceLab home dataset [22], the CASL office dataset [15] and the home dataset by Van Kasteren et. al., [23]. The differences between these datasets can be seen in Table 1. Since these techniques were evaluated using identical setups they are directly comparable. We further analyse these evaluation results as a basis for understanding the relationship between scenarios and situation identification techniques.

The evaluation results presented show that one solution does not outperform the others in all scenarios. By scenario we mean identifying a specific situation given the characteristics of the dataset. By analysing these results in terms of how each technique performed for each scenario we can begin to understand the links between the physical characteristics and requirements of scenarios and how this effects the performance of a technique for that scenario.

Technique	**Scenario**	**F-Measure**
CBR	Eating+PlaceLab	0.39
DT	Eating+PlaceLab	0.41
NB	Eating+PlaceLab	0.0
SVM	Eating+PlaceLab	0.0
CBR	Watching TV+PlaceLab	0.46
DT	Watching TV+PlaceLab	0.47
NB	Watching TV+PlaceLab	0.73
SVM	Watching TV+PlaceLab	0.65
CBR	Go to bed + Van Kasteren	0.02
DT	Go to bed + Van Kasteren	0.52
NB	Go to bed + Van Kasteren	0.58
SVM	Go to bed + Van Kasteren	0.63

Table 3. Performance of techniques for PlaceLab Scenarios. Taken from [16]

SVMs and NB performed very well in a limited number of scenarios that occur frequently, i.e., have more available training data. This is illustrated in table 3 where the performance in terms of f-measure is listed for each technique for four scenarios. The first scenario identifies the situation where someone is eating and the second identifies someone watching TV, both using the PlaceLab dataset. From table 2 it is clear that there is many more instances of the eating scenario compared to that of watching TV, but the watching TV scenario has a much longer average duration meaning that there is more data relevant to that scenario. This suggests that SVM's and NB are better solutions for scenarios that are longer in duration. Both techniques failed to identify any of the brief scenarios in this from the PlaceLab dataset and this further demonstrates the inability of a single technique to solve all problems.

CBR and DT's were identified as the most consistent techniques across scenarios although CBR struggled to distinguish between similar scenarios in some cases. The "go to bed" situation from the Van Kasteren dataset relies the context-type time to be identified. As can be seen in table 3 CBR is the only technique that performed poorly for this scenario.Interestingly however, CBR was by far the most successful in identifying the "Prepare breakfast" situation from the same dataset. This scenario relied on time and device contexts. This demonstrates that CBR is not the optimal solution for solely time based scenarios. For scenarios where multiple contexts are available, and the scenario is distinct from others, CBR performs well.

Discussion Analysis of the evaluations performed by Knox et. al., [16] shows the relationship between the situation identification techniques and application characteristics. Support Vector Machines and Naive Bayesian were identified (out of the techniques in question) as being optimal solutions for scenarios that have long lasting durations where there is more training data. They should be avoided however for brief scenarios as they often fail to identify the situation at all. Case Based Reasoning was shown to perform well for scenarios that involve multiple context types, but poorly for scenarios that were solely time based. Case Based Reasoning also struggles to distinguish between similar scenarios. These evaluations have given us an initial insight into how the characteristics of applications relate to the performance of situation identification techniques. This understanding forms the basis for a profile of situation identification techniques which we will use in order to automatically select the most suitable solutions.

4 Architecture: Unifying Framework for Existing Situation Identification Methodologies

Our architecture aims to provide a decision framework for the selection of situation identification techniques on the behalf of the developer. It is divided into three layers as illustrated in figure 1. The developer layer supports application specific inputs by providing templates to be filled in by the developer. These templates are based on the important characteristics of application deployments/scenarios defined in section 3.1. This layer also displays the output from lower layers, i.e., the recommended situation identification technique(s). The decision layer contains the logic for selection of technique(s) from the situation identification layer, based on the application profile. We describe these layers in more detail using a smart home application example.

4.1 Developer Layer

The deployment template supports the input of information unique to the physical deployment for the application. The required information may be provided as domain knowledge, e.g., information provided from the proposed user of the system, and/or from datasets gathered from the deployment.

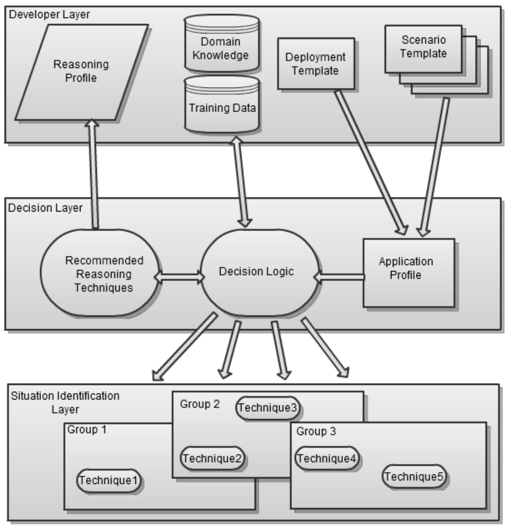


Fig. 1. Architecture of a unifying framework for situation identification methodologies

In our smart home example the deployment consists of 30 state change sensors attached to household objects and 5 location sensors monitoring the users movements throughout the house. The state change sensors are present in the kitchen, living room and bathroom. There is a dataset with 100 hours of annotated data available.

```
<Deployment name = "Case Study Deployment">
    <Characteristics>
        <Size = "40"  />
        <SensorTypes = "statechange, location"  />
        <NumParticipants = "1"  />
        <DatasetDuration = "100h"  />
    </Characteristics>
</Deployment>
```

Scenario templates support the input of scenario specific characteristics. These characteristics have been chosen as they have been shown in previous evaluations to have a connection with the performance of specific situation identification techniques. Using the scenario templates, the developer describes each of the scenarios that need to be identified in the application. This information comes from domain knowledge and/or training data. Training data can also be used to validate any manually entered domain knowledge.

We describe three common scenarios for home applications, eating, hygiene, and gone to bed. Watching TV occurs about once every two days lasting 30 minutes on average, there is no pattern to it's occurrence and the contexts it relies on are location and device. Gone to bed lasts 8 hours on average, usually occurs once daily between 11pm and 1am and relies on location and time contexts. The eating scenario template is illustrated below:

```
<Scenario name = "Eating" >
    <Characteristics>
        <AverageDuration = " 10m"  />
        <ExpectedFrequency = " 2/1d "  />
        <Regularity = "~9am, ~6pm "  />
    </Characteristics>
    <ContextTypes>
        <Location = "yes"  />
        <Time = "yes"  />
        <Identity = "no"  />
        <Device = "yes"  />
        <Application = "no"  />
    </ContextTypes>
</Scenario>
```

The reasoning profile displays the output of the decision framework to the developer. The situation identification technique(s) selected will be presented to the developer illustrating which scenario(s) each technique is most suited to. We give an example of this output after describing the decision making process.

4.2 Decision Layer

The application profile component combines information provided in the deployment and scenario templates and builds a profile for the application as a whole. Here, scenarios and relevant deployment characteristics are analysed and grouped according to the characteristic similarities and contextual similarities. This step allows us to identify similar scenarios which can prove difficult to identify for some situation identification techniques. This was shown to be an issue for Case Based Reasoning for example. This profile will serve as a more concise description of application characteristics for use by the decision logic.

Based on the application and technique profiles, the decision logic component selects technique(s) based on their ability to effectively perform situation identification for the application as a whole, i.e., the applications physical and scenario specific characteristics. For example, a scenario may rely on context from sensors that are prone to error/failure such as environmental sensors. This needs to be accounted for in the decision making process. The selected technique should be able to identify such a situation while managing the uncertainty of the data. Techniques that can not manage a particular scenario's characteristics with the training data available are ruled out for that scenario, but not for others.

A profile of the relevant situation identification technique(s) is built incrementally by the recommended reasoning techniques component. This again is assessed by the decision logic component, reducing the number of techniques available based on the similarities of scenarios. In order to prevent the recommendation of an unnecessarily large number of techniques the decision logic component analyses the trade-offs between techniques, recommending as few techniques as possible while maintaining an optimal solution. The complete recommendation forms the basis for a reasoning profile in the developer layer. The final reasoning profile is presented to the developer via the reasoning profile in the decision layer, illustrating the grouping of similar scenarios and the optimal solutions for each group. The following is a sample output for our smart home example.

```
<ReasoningProfile>
    <RecommendedTechnique name = "Case Based Reasoning">
        <Scenario name = "Watching TV" />
    </RecommendedTechnique>
    <RecommendedTechnique name = "Support Vector Machines" >
        <Scenario name = "Eating" />
        <Scenario name = "Gone to bed"/>
    </RecommendedTechnique>
</ReasoningProfile>
```

4.3 Situation Identification Layer

This layer contains the situation identification techniques employed by the framework and a profile of each. Techniques are grouped according to their previously discussed capabilities and may belong to more than one group. It is important here, to allow for additional techniques to be added to the framework as they become available. The technique profiles provide a description of the strengths and weaknesses of each technique.

5 Summary and Future Work

Situation identification techniques are limited in terms of the variety of applications they can be successfully applied to. We identified an ideal solution to be one where techniques can be interchanged according to application needs. We characterised applications in terms of their deployment and scenario characteristics. We analysed evaluations of situation identification techniques in relation to these characteristics and gave an initial description of the relationship between application scenarios and how the applied techniques perform. We proposed a unifying framework that automatically selects the most suitable combination of techniques for a given set of application characteristics. It is argued that a unifying framework will support a single development platform for a diverse range of context-aware applications.

While we used a smart home example to illustrate the workings of our architecture the framework aims to be accessible to a diverse range of applications.

Our initial evaluations analysis identified some relationships between techniques and the scenarios they are applied to. We will perform further evaluations focussing on diverse scenarios in order to identify further the characteristics (and combinations of characteristics) that affect the performance of situation identification techniques. We also intend to include application requirements into the decision making process. Until now we assume that all scenarios simply require the best solution available. Realistically however, some scenarios may be safety critical and as such may require the combination of techniques used in that scenario in order to reach a satisfactory level of certainty. While our current aim is to gain a greater understanding of the technique-scenario relationships, it would also be interesting to apply machine learning to the decision making process in order to learn optimal solutions. The recommended reasoning techniques from this architecture have been described using XML. The presentation of such output to the user, illustrating the benefits of one technique over the others for particular scenarios, will be further investigated.

References

1. M. Weiser, The Computer for the 21st Century, Scientific Am., pp. 94- 104. (1991)
2. Dey, Anind K. and Gregory D. Abowd (2000b). Towards a better understanding of context and context awareness. In the Workshop on the What, Who, Where, When and How of Context-Awareness, affiliated with the 2000 ACM Conference on Human Factors in Computer Systems (CHI 2000), The Hague, Netherlands. April 1-6, 2000
3. Simon Dobson and Paddy Nixon. Whole-system programming of adaptive ambient intelligence. In Proceedings of HCI International. LNCS. Springer-Verlag. Beijing, CN. 2007.
4. Anand Ranganathan, et al. A Middleware for Context- Aware Agents in Ubiquitous Computing Environments, USENIX International Middleware Conference, 2002.
5. Truong BA, Lee Y-K, Lee S-Y (2005) Modeling and Reasoning about Uncertainty in Context- Aware Systems. IEEE International Conference on e-Business Engineering 2005: 102-109.
6. H. Hagras, V. Callaghan, M. Colley, G. Clarke, A. Pounds-Cornish, and H. Duman. Creating an ambient-intelligence environment using embedded agents. IEEE Intelligent Systems, 19(6):12-20, 2004.
7. Panu Korpipaa, Miika Koskinen, Johannes Peltola, Satu-Marja Mkel, and TapioSeppanen. Bayesian Approach to Sensor-Based Context Awareness, Per- sonal and Ubiquitous Computing J., vol. 7, no. 4, 2003.
8. L. R. Rabiner and B. H. Juang, An introduction to hidden Markov models, IEEE Acoust., Speech, Signal Processing Mag., pp. 4-16, Jan.1986.
9. Englebienne VanKasteren, Noulas and Krose. Accurate activity recognition in a home setting. In In Proceedings of Tenth International Conference on Ubiquitous Computing, South Korea, September 2008.
10. Kofod-Petersen, A., and Aamodt, A. 2009. Case-based reasoning for situation-aware ambient intelligence: A hospital ward evaluation study. In ICCBR, 450464.
11. Cassens, J. and Kofod-Petersen, A. (2007b). Explanations and case-based reasoning in ambient intelligent systems. In Coyle, L. and Schwarz, S., editors, Case-Based Reasoning and Context-Awareness, The Seventh International Conference on

Case-Based Reasoning (ICCBR 07), Workshop Proceedings, pages 167176, Belfast, Northern Ireland. University of Ulster

12. P. Castro, P. Chiu, T. Kremenek, R. Muntz. A Probabilistic Room Location Service. Proceedings of Ubicomp 2001: Ubiquitous Computing. Atlanta, Georgia, September 2001.
13. Randell, C., Muller, H., Context Awareness by Analyzing Accelerometer Data, Fourth International Symposium on Wearable Computers (ISWC'00), p. 175-176, Atlanta, Georgia, October 18 - 21, 2000.
14. Huimin Qian, Yaobin Mao, Wenbo Xiang, and Zhiquan Wang. Recognition of human activities using svm multi-class classifier. Pattern Recogn. Lett., 31(2):100111, 2010.
15. Stephen Knox, Lorcan Coyle and Simon Dobson. Using ontologies in casebased activity recognition. In Proceedings of the 23rd International Conference of the Florida Artificial Intelligence Research Society (FLAIRS- 23). Daytona Beach, FL. May 2010.
16. S. Knox, Combining Case-Based Reasoning and the Semantic Web in Recognising Situations. PhD thesis, University College Dublin, School of Computer Science and Informatics, 2010
17. Juan Ye, Lorcan Coyle, Susan McKeever and Simon Dobson. Dealing with activities with diffuse boundaries. In Proceedings of the Workshop on How to do good activity recognition research: Experimental methodologies, evaluation metrics and reproducility issues at PERVASIVE 2010. Helsinki, FI. May 2010.
18. Michael Przybilski, Petteri Nurmi, Patrik Floren: A Framework for Context Reasoning Systems Proceedings of the 23rd IASTED International Conference on SOFTWARE ENGINEERING (SE 2005), pages 448 - 452
19. P. Nurmi, P. Floren, M. Przybilski and G. Linden, A Framework for Distributed Activity Recognition in Ubiquitous Systems In proceedings of the International Conference on Artificial Intelligence (ICAI '05), pp. 650 - 655, Las Vegas, 27 - 30 June, 2005.
20. Nishkam Ravi, Nikhil Dandekar, Preetham Mysore, and Michael L. Littman. Activity recognition from accelerometer data. In AAAI, pages 1541-1546, 2005.
21. L. Bao and S. Intille. Activity Recognition from User-Annotated Acceleration Data. Proc. Pervasive, 1-17, Vienna, Austria, 2004.
22. B. Logan, J. Healey, M. Philipose, E. M. Tapia, and S. S. Intille. A long-term evaluation of sensing modalities for activity recognition. In Ubicomp, pages 483500, 2007.
23. T. van Kasteren, A. Noulas, G. Englebienne, and B. Krose. Accurate activity recognition in a home setting. In UbiComp 08: Proceedings of the 10th international conference on Ubiquitous computing, pages 19, New York, NY, USA, 2008. ACM.
24. Hung Q. Ngo et al: Developing Context-Aware Ubiquitous Computing Systems with a Unified Middleware Framework. In Proceedings of the EUC 2004: 672-681.
25. B. A. Truong, Y.-K. Lee, S.-Y. Lee: Modeling Uncertainty in Context-aware Computing. 4th Annual ACIS International Conference on Computer and Information. Science (ICIS05)
26. Waltenegus Dargie. The role of probabilistic schemes in multisensor contextawareness. IEEE International Conference on Pervasive Computing and Communications Work-shops, pages 27 32, 2007.

Mental Models of Disappearing Systems: Challenges for a Better Understanding

Felix Schmitt, Jörg Cassens, Martin C. Kindsmüller, and Michael Herczeg

Institute for Multimedia and Interactive Systems
University of Lübeck
23538 Lübeck, Germany
{schmitt|cassens|mck|herczeg}@imis.uni-luebeck.de

Abstract. In this paper, we describe our current research concerning users' mental models of what can be called "disappearing computer systems". This notion comprises computer systems, applications, and appliances related to ubiquitous, pervasive, or ambient computing which blend more or less seamlessly into the users' natural environment. Mental models enable users to formulate expectations about which interactions with the system are possible and how the system will react to certain interactions. Disappearing computers lack certain cues regarding the inner workings of the system. We thus hypothesize that the mental models users build of such a system will show defects and inaccuracies that are directly related to the distributed character of interface and interaction. Our current research aims at identifying the nature of these defects, understanding their effects on human computer interaction, and developing means of avoiding them through appropriate design of both user interface and underlying system. For this purpose, we are developing an ambient, context aware computing framework with which we generate and test hypotheses in an action research paradigm. One of its components will be described along with possible future additions. The theoretical foundation of our work lies in such diverse fields as systemic functional theory of language, activity theory, and cognitive science approaches to mental models.

1 Introduction

The computer as we know it, namely as a box equipped with monitor, keyboard, and mouse, is appended with systems that are not easily identifiable as computer systems. For the last three decades, these boxes have become an increasingly important part of most peoples' everyday life in work, education, and leisure, and the exploding feature sets of modern Personal Computers have in practice turned them into the "universal machines" that they were theoretically from the very beginning. But for some ten years now, another tendency can be observed. The Personal Computer as a single, distinct, and discrete information processing device is being augmented and partially supplanted. Advances in silicon circuitry and wireless data transmission technologies have made it possible to put more processing power, storage space, and communication interfaces in ever shrinking devices, which in turn leads to the transformation of formerly analog or mechanical, unconnected devices into computer-enhanced, connected systems.

1.1 Disappearing Computers

These recent developments that usually go by the name of pervasive or ubiquitous computing have been the object of extensive scientific and public discourse (cf. [15,25,29]). The interconnections between these miniature computers form a sort of meta-computer that is spatially (and temporally) spread across potentially large areas: instead of a single computer serving a single person we now have several computers serving one or several persons.

In addition, instead of utilizing distinct user interfaces, such as screens, keyboards, and mice, they are often embedded into other artifacts, and any interaction with any such artifact becomes interaction with the computer. To capture these changes, the term ambient computing was coined. The term ambient intelligence describes the ability of systems to become more proactive and make assumptions about their surroundings or their users on their own accord [7].

For the purpose of this paper, all computer systems that can be named under either of these terms (ubiquitous, pervasive, or ambient computing) share one important property: partial or complete invisibility. Such systems do not appear as distinct and discrete computers as the PC once did. Instead, they vanish within the artifacts they are embedded in. They usually also do not have the prominent and familiar in- and output devices for user interaction that PCs have. So it is not only the physical box that is disappearing, but also the interaction with the system. Formerly, human-computer interaction took place at one particular place, namely at one's desk, and at a particular point in time, namely when one typed on the keyboard or used the mouse to point and click. With disappearing computers, the artifacts and events shaping the in- and output of the system are not necessarily recognizable as interfaces of and interaction with computer systems.

In summary, these are the reasons why we speak of disappearing computer systems (cf. Norman [20]). The twofold disappearance of computers raises numerous questions with regard to the design principles of their user interfaces and the way the inner working mechanisms of the system are conveyed to the user. Section 2 below will outline our thoughts on this topic.

1.2 Mental Models

In virtually all situations in everyday life, people rely on mental models of relevant aspects of the world in order to evaluate the situation, plan their actions, and formulate expectations of the outcome of these actions [19]. Literature reveals that different domains have different definitions of mental models and have established a wide range of theories about how these models are built, chosen to fit a particular situation, and made use of [23]. In their comprehensive review of literature dealing with mental models, Rouse and Morris [23] deliver a functional definition of mental models that is well suited for the needs of designers of complex systems: "Mental models are mechanisms whereby humans are able to generate descriptions of system purpose and form, explanations of system functioning and observed system states, and predictions of future system states". It is repeatedly shown that mental models can be extremely complex and are individual to the respective person, since they are usually elaborated over time and linked to other mental models and knowledge about the world (cf. [11,30]).

This leads to three main characteristics of mental models (cf. Dutke [8]) that must be taken into account when using them to assess the usability of disappearing computer systems.

1. Mental models are never complete or exhaustive when compared to the real world. There are always more details to include in the model, and this refinement seems to be limited only by the trade-off between the model's complexity and its useful applicability in actual situations.
2. Mental models are resistant to changes. This is basically due to the fact that changing existing models requires learning effort, and thus there is a trade-off between this effort and the perceived usefulness of the resulting improvement of the mental model.
3. Mental models are unstable, since people tend to forget details of their once-acquired mental models over time.

Norman [19] summarizes that "most people's understanding of the devices they interact with is surprisingly meager, imprecisely specified, and full of inconsistencies, gaps, and idiosyncratic quirks". And, we would like to add, it is the foremost duty not only of user interface designers, but of system engineers as well, to account for these quirks and build their systems to deliver an untroubled user experience. How this can be done is the subject of our current research.

2 Understanding and Shaping Mental Models

We have recently begun working on the topic of mental models in users of disappearing computer systems, because this class of systems raises questions that cannot be answered satisfyingly with current theories of mental models. Our approach is twofold. On the one hand, we address the analysis of the mechanisms which affect the building of mental models in such users. On the other hand, we aim at identifying design principles that foster development of useful and adequate mental models.

Research on ambient and context aware systems needs to address different levels of scientific insight. On the one hand, the concepts of "context" and "awareness" need to be explored, as they form a crucial part of our understanding of the situatedness of a user's perception and use of disappearing computer systems. On the other hand, building upon this theoretical foundation, we also need real ambient systems with which we can test our hypotheses, derive new ones, and explore the regularities of various facets of human-computer interaction in this domain.

2.1 Challenges for Mental Models of Disappearing Systems

The functionality of disappearing computer systems as described in the Introduction is often achieved by collecting a large variety of data via distributed sensor nodes. Often this data is also (pre-) processed and the results are delivered back to the user in a distributed way. Our research concerns primarily systems that become more proactive and learn, often referred to as ambient intelligence. Ducatel and others [7] give a definition of ambient intelligence: At the core of an ambient intelligent system lies the ability to appreciate the system's environment, be aware of persons in this environment, and respond intelligently to their needs.

We postulate that challenges to the development of suitable mental models for ambient intelligent and disappearing computer systems emerge along two dimensions:

First, the environment and artifacts become the interface, with the difficulty of exhibiting suitable affordances for its use. For traditional computer systems, it is considered good practice to provide good affordances, that is to provide interaction possibilities that are readily perceivable by a user [21]. This becomes difficult when artifacts in the environment are charged with additional functionality. For example, Bob might usually take a bowl of muesli in his lunch breaks. Taking this bowl and going out of the office can now be perceived by an ambient system as a sign that Bob is having a break, thus overloading the bowl with additional functionality. It is not obvious how this function of the bowl can be easily communicated to the user.

Second, as the technical system becomes proactive, it might produce new and surprising results, or change itself as well as the environment, possibly without direct user interaction. For example, if one of Bob's meetings takes longer than expected, an advanced ambient system might re-schedule Bob's later appointments without disturbing the current meeting. This is the classical problem of traditional, non-ambient artificial intelligent systems. A promising way to deal with this challenge is to provide explanations about actions taken and reasoning performed [17].

Ambient intelligent systems face challenges on both dimensions at the same time: the aforementioned muesli bowl is now only one part of the sensor network, and its absence or presence might or might not be taken into account to decide whether Bob is having a break, all depending on other sensor data and system states. This function of the bowl has to be communicated to the user, but there is no easy way to deliver explanations: probably, there are no classical interfaces. Even if there were, using them would mean that the artifact immediately looses its ambient character.

2.2 Theory Construction

With regard to the theoretical foundations of our work, we draw from recent work in context awareness [12], explanation awareness [4,13], and theories of ambient user interface design [5].

Our take on disappearing systems as having some form of artificial intelligence forms the backdrop for exploring methods to better understand how mental models of such systems are formed, what influences their shaping, and for formulating methods to use the understanding gained to support the development process of ambient intelligent systems.

To achieve this, we make use of theories from the fields of cognitive science, human-computer interaction, psychology, sociology, and linguistics. The specific theoretical frameworks we draw on are, first and foremost, activity theory [14] and the systemic functional theory of language (SFL) [10], a social semiotic framework.

A major problem with research of this nature, research which attempts to integrate theories from diverse areas of practice, is the perception that there are very different underlying philosophies. This is of particular importance when trying to integrate the strengths of theories for the purposes of solving real world problems. Both activity theory and SFL have been used in the field of ambient intelligence and explanation awareness [4,28,31].

Following Carley and Palmquist [3], who have examined mental models from a symbolic, language perspective, we utilize SFL to understand how users' mental models are formed. We deem SFL particularly useful because it looks at language in a very general sense as a means of interaction. We interact not just with each other, but with our own constructions and with our natural world, and this interaction is inherently multimodal [9]. Another perspective on mental models in relation to language is given by van Dijk's [6] theory of mental and context models. Van Dijk himself is influenced by SFL and episodic memory as proposed by Schank [26].

But our mental models are not only shaped by acts of communication, they are also construed by our acting on and with artifacts. Therefore, in our research, we explore to which extent it is possible to relate a semiotic approach to activity theory. Bødker and Andersen have outlined some properties of a socio-technical approach taking advantage of ideas from both theoretical frameworks [2], and we would like to extend this to cover specific aspects of SFL and cultural-historical activity theory (CHAT, cf. [18]). This will potentially lead to a richer understanding of the different aspects of mental models as both result and prerequisite of communication and of manipulation of artifacts.

2.3 Empirical Evaluation

Our primary scenario for the empirical part of our work is that of an ambient, intelligent system aimed at facilitating cooperation and mutual awareness in work teams (cf. Sect. 2.4). We chose this scenario since, due to the system's distribution in time and space, the users' interaction with the system's intelligent functions will partly be implicit. That means that the system might, for example, change users' interruptibility status or spatial location information without being told to do so by explicit user interaction or without direct announcement of such changes to its users. The system should also be able to infer correct information even from incomplete or inconsistent source data. This reasoning capability requires a large amount of data from various sources, aggregated and processed into a knowledge model, and suitable inference mechanisms. Even if we expect the system to draw the correct inferences, it seems obvious that the system would need some kind of explanation component in order to make clear to the user why a specific decision has been made. In particular, what peculiarities in the data set or reasoning mechanism have led to a different decision than in a previous, albeit superficially similar situation [4].

In a system with sufficiently high complexity and numerous data sources to be useful for real-life tasks, simply printing out the chain of firing production rules or best-matched cases on a screen would surely confuse the user more than it would help [16] as well as if, e.g. a neural network or other subsymbolic system was part of the decision process, this display of the reasoning trace would either not be possible at all or very difficult. Thus, it is of great importance for the usability of ambient computer systems that they are able to explain themselves in an unobtrusive, yet comprehensive and sufficiently detailed way. How this can be achieved is subject to our current research.

2.4 The MATe Framework

The MATe system is a family of appliances, client- and server-based software that aims at improving situation awareness in work teams. The acronym MATe stands for "Mate for Awareness in Teams". The system is designed to blend seamlessly with the team members' everyday routine, enabling unobtrusive in-situ interaction and facilitation of cooperation and communication. Currently, MATe consists of several components (see Fig. 1), which we will briefly describe, and is being implemented at our institute.

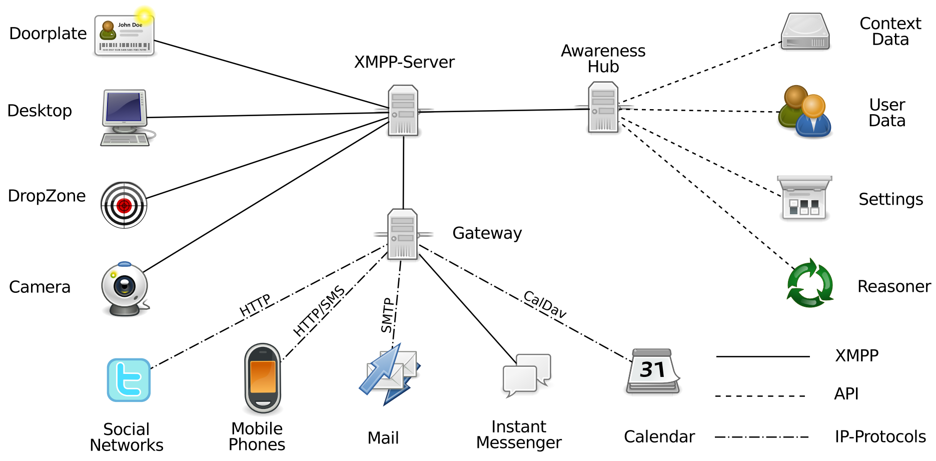


Fig. 1. MATe: Architectural Overview

AwarenessHub. The AwarenessHub is the central service to which all other components connect. It handles all communication among the components and takes care of the correct routing of messages and the appropriate processing of different pieces of information. Technically, the AwarenessHub is implemented as a client which can connect to any instant messaging server using the Extensible Messaging and Presence Protocol (XMPP[1]). It has all other components of one specific MATe installation in its contact list and can thus exchange XMPP messages conforming to the MATe protocol specification with them.

Since it is not guaranteed that the information available to the AwarenessHub will be consistent at all times, a component for resolving conflicting status indications is needed. It might be that a user forgot his keys in someone else's office, but went out for lunch anyway, while not having corrected a calendar entry concerning a meeting with another colleague. MATe should be able to deduce reliable information about the users' current status and spatial position by applying appropriate reasoning techniques to all available source data.

In addition, the AwarenesHub will feature reasoning capabilities to induce the current interruptibility status of MATe users. MATe's reasoners and necessary ontologies are currently under development [24] and will vastly improve the MATe system's usefulness in actual field use.

DoorPlate. The DoorPlate is the component which offers the most comprehensive interface to MATe and thus requires special consideration regarding the user interface design. Every team member using MATe has a DoorPlate on his office door. It serves numerous functions: First, it displays the usual information found on door plates, such as room number and the occupant's name. Second, it conveys information about the occupant's current status to visitors, i.e. whether he is present or absent, and, in the former case, whether he agrees to be disturbed by visitors or not. Third, if the person is not interruptible, visitors can leave one of several predefined messages via the DoorPlate's touch screen. Fourth, the occupant of the room can use the touch screen to set his status and view messages in case there are any.

Desktop PC. For convenient interaction with the MATe system, users find a desklet on their computer screen. It allows the user to check the current status of his colleagues, set his own interruptibility status, and read and send messages to other users. Basically, the desklet offers the same functionality as the DoorPlate right at the user's workplace. In addition, the user can spawn a web-based interface for setting up the desired information channels and configuring privacy settings, i.e. which colleagues should be able to access which information about his status and current spatial position.

DropZone. The DropZone is an ambient frontend to the MATe system, which tracks the presence of users in the room it is installed in (e.g. personal office,

[1] http://xmpp.org/

conference room, laboratory). Users carry a token, and, when entering the room, put this token inside the DropZone. A small software program then reads the token and signals the respective user's presence to the AwarenessHub. There is no back channel. The DropZone is implemented using optical markers in combination with an ordinary consumer-grade web cam.

Optical Recognition. Since the DoorPlate is currently based on an embedded ARM platform focused at low electrical power consumption, local computing power is also weak. With regard to the DropZone, computing power is readily available in modern desktop PC systems. But our intention is to keep MATe's impact on the users' usual working routine as low as possible, which includes the performance of their desktop application software. For both reasons, we opted for a server side component for the optical marker recognition.

Gateways. Several other services can be accessed via a gateway to the MATe XMPP server. For example, the SMS gateway is a server side component, which is able to send and receive short text messages via the HTTP interface of common HTTP-to-SMS providers. These messages must follow a very simple syntax and may either communicate status changes to the AwarenessHub or request information from it. Incoming text messages are parsed, converted to appropriate XMPP packages, and forwarded to the AwarenessHub, while outgoing XMPP messages are recoded to short natural language sentences and sent to the user's mobile phone. Likewise, the user can communicate with MATe via his instant messaging client.

Our plan is to include other services as well, for communication outwards and as additional sensors. For example, micro blogging sites like Twitter can be used for status updates, or a user's calendar can be accessed via an appropriate protocol. Future planned enhancements include a mail gateway with rudimentary natural language understanding capabilities so that MATe can make use of implicit information contained in mails between co-workers.

2.5 Methods

For some time now, our research group has been developing hardware appliances and software systems with which we intend to test our hypotheses as well as develop new ones, and derive design principles for ambient and context aware computer systems. One of these systems has been described in Sect. 2.4 above. Further on, our research group's expertise in alternative user interfaces and interaction devices enables us to try out different ways of communicating the systems' output and reasoning to their users. With several multi-touch tables, 3D projection systems, and a diverse array of wearable devices at hand, we are exploring new representations of our systems' states and processes in a variety of usage contexts.

In addition, besides MATe, several new systems are under development which will further add ambient and contextualized capabilities. In this sense, the research on mental models is a kind of action research: our hypotheses are derived from the existing literature, they are put to the test on existing systems, and adapted to suit the empirical findings. The changed hypotheses will then lead to a revision of our development models, before systems built with the according to changed principles will be tested again to confirm or challenge our reformed hypotheses.

The experimental settings with which we intend to test our hypotheses are designed to fit the MATe system. One of MATe's functions is to convey the user's current interruptibility status to colleagues and visitors, that is, whether it is acceptable to disturb the user. This information will be displayed on the user's DoorPlate, but can also be accessed via Desktop PC applications, by querying the AwarenessHub via the SMS Gateway or an automatically generated Twitter announcement. MATe can obtain the data from which this status information is derived from different sources. These sources can be categorized with regard to the challenges outlined in Sect. 2.1 above.

– A button press in the desktop PC application or on the DoorPlate categorizes as an explicit interaction. The effect of this interaction is also explicit and thus can be looked at with the usual methods of HCI and usability studies.
– Setting one's interruptibility status by taking a muesli bowl to the lunch room is an implicit interaction. It is a voluntary and conscious action, but the user has to learn that it effects his interruptibility status.
– Automatically setting the user's interruptibility status according to his current use of application software on his PC or his typing behavior counts as an implicit interaction. The system's input does not depend on any voluntary and/or conscious action by the user, and the effects of each action by itself are not obvious.

These different interactions will be evaluated with the help of several scenarios. Typical for the use of MATe is the following situation: Alice is in the library reading a paper, while Bob would like to go out for lunch. Usually, he would have to search the institute for several minutes to find Alice in the library. He then would approach her to ask about lunch, thus interrupting her concentration on the paper. With MATe's assistance, Bob will not have to search for Alice, but he will know where to find her after a quick glance at his PC's screen or Alice's DoorPlate. At the same time, MATe will have inferred that because Alice is in the library, but has not typed on her laptop's keybard for some time, she is probably reading and not to be disturbed. Bob can then decide to wait for Alice, to leave without her, or to interrupt her anyway.

Several methods for data collection are currently under evaluation. Since users' mental models are usually complex and difficult to extract (see Sect. 1.2), let alone to interpret and evaluate, research methods suitable to cope with these properties are needed. Established methods from cognitive psychology and usability engineering do not seem to fit our needs because they are created to measure the deviation of the mental model of the user from the conceptual model of the target system. Due to the fact that the system image [19] of a disappearing computer cannot easily delivered by affordances of the user interface the system image has to be actively constructed by the users. Therefore we have to employ methods of elicitating these constructed system images and mental models from the users. This will be done by qualitative user interviews (see, e.g., [1]) and a newly adopted version of the structure-formation-technique by Scheele and Groeben [27] using concept maps (cf. Novak and Cañas [22]) drawn by the users of the MATe system. We plan to collect these data at several times during the prolonged usage of the system in an everyday office context. Besides primary tests at our research group, we will continuously evaluate other options for field tests to ensure that our user base is comprised of persons with different levels of expertise.

2.6 Future Enhancements

Although MATe already makes a valuable framework for the empirical testing of our theoretical work and also for the explorative research that helps generate new hypotheses, we plan to extend this framework to include other services and devices which we believe will help us to understand certain aspects of the field better and gain additional data sources for those components that already exist. Since mobile computer systems and thus also mobile human computer interfaces are present in most peoples' lives, we would like to cover mobility and mobility-related computer systems with our framework.

One attempt we are currently making to this end is the adaption of personal navigation and planning appliances to the MATe system. Think of your car being connected to your PDAs calendar application and offering you a pre-planned route to your next appointment. It might even tell your PDAs calendar that it should remind you of the appointment a little earlier than usual, since there is a traffic jam along the way and you need to fill up gasoline anyway. Another interesting service might be to check public transport facilities' time tables and not only propose the best connections, but even buy a ticket online and order a taxi to take you the last mile to your destination.

3 Conclusions and Further Work

We have outlined a research program focusing on how mental models of ambient, distributed systems come into being, and how these mental models differ from those arising from interaction with traditional computer systems. Our main axioms are:

1. Mental models can be understood from a perspective which unifies aspects of communication and artifact manipulation of human-computer interaction.
2. Distributed, pervasive, ambient systems and environments pose unique challenges for the development of users' mental models.
3. A better understanding of these challenges can lead to enhanced methods for the development of ambient systems which give better clues for the development of mental models.

Hypotheses about mental models of disappearing computer systems will be generated starting from these axioms. They are put to the test with the help of our research prototypes. We start off with the MATe system. MATe is currently developed to be put into use in a team of knowledge workers to increase awareness about other users' states and interruptibility. We are collecting empirical data about how the users manage interruptibility and their work status in general, and can examine how mental models of the system develop during its use.

References

1. Berg, B.L.: Qualitative Research Methods for the Social Sciences. Allyn and Bacon, Needham Heights, MA, 4 edn. (2001)
2. Bødker, S., Andersen, P.B.: Complex mediation. Journal of Human Computer Interaction 20(4), 353–402 (2005)
3. Carley, K., Palmquist, M.: Extracting, representing, and analyzing mental models. Social Forces 70(3), 601–636 (March 1992), `http://www.jstor.org/stable/2579746`
4. Cassens, J.: Explanation Awareness and Ambient Intelligence as Social Technologies. Thesis for the degree doctor scientiarum, Norwegian University of Science and Technology, Trondheim, Norway (May 2008), `http://urn.ub.uu.se/resolve?urn=urn:nbn:no:ntnu:diva-2122`
5. Dey, A.K., de Guzman, E.: From awareness to connectedness: the design and deployment of presence displays. In: Grinter, R., Rodden, T., Aoki, P., Cutrell, E., Jeffries, R., Olson, G. (eds.) Proceedings of the SIGCHI conference on human factors in computing systems. pp. 899–908. ACM Press (2006)
6. van Dijk, T.A.: Discourse, context and cognition. Discourse Studies 8(1), 159–177 (2006)
7. Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., Burgelman, J.C.: ISTAG scenarios for ambient intelligence in 2010. Tech. rep., IST Advisory Group (2001)
8. Dutke, S.: Mentale Modelle: Konstrukte des Wissens und Verstehens. No. 4 in Arbeit und Technik, Verlag für Angewandte Psychologie, Göttingen (1994)
9. Halliday, M.A.: Language as a Social Semiotic: the social interpretation of language and meaning. University Park Press (1978)
10. Halliday, M.A., Matthiessen, C.M.: An Introduction to Functional Grammar, Third edition. Arnold, London, UK (2004)
11. Hollnagel, E.: Mental models and model mentality. In: Goodstein, L.P., Andersen, H.B., Olsen, S.E. (eds.) Tasks, errors, and mental models, pp. 261–268. Taylor & Francis, Inc., Bristol, PA, USA (1988)
12. Kofod-Petersen, A.: A Case-Based Approach to Realising Ambient Intelligence among Agents. Thesis for the degree doctor scientiarum, Norwegian University of Science and Technology (2007)
13. Kofod-Petersen, A., Cassens, J.: Explanations and context in ambient intelligent systems. In: Kokinov, B., Richardson, D.C., Roth-Berghofer, T.R., Vieu, L. (eds.) Modeling and Using Context – CONTEXT 2007. Lecture Notes in Computer Science, vol. 4635, pp. 303–316. Springer, Roskilde, Denmark (2007), `http://dx.doi.org/10.1007/978-3-540-74255-5_23`
14. Leont'ev, A.N.: Activity, Consciousness, and Personality. Prentice Hall, Upper Saddle River, NJ, USA (1978)

15. Lueg, C.: Representation in pervasive computing. In: Proceedings of the Inaugural Asia Pacific Forum on Pervasive Computing (2002)
16. Majchrzak, A., Gasser, L.: On using artificial intelligence to integrate the design of organizational and process change in us manufacturing. AI and Society 5, 321–338 (1991)
17. Mao, J.Y., Benbasat, I.: The use of explanations in knowledge-based systems: Cognitive perspectives and a process-tracing analysis. Journal of Managment Information Systems 17(2), 153–179 (2000)
18. Mwanza, D.: Mind the gap: Activity theory and design. Tech. Rep. KMI-TR-95, Knowledge Media Institute, The Open University, Milton Keynes (2000)
19. Norman, D.A.: Some observations on mental models. In: Gentner, D., Stevens, A.L. (eds.) Mental Models, pp. 7–14. Lawrence Erlbaum Associates, Hillsdale, New Jersey (1983)
20. Norman, D.A.: The Invisible Computer. The MIT Press, Cambridge, MA (1999)
21. Norman, D.A.: The Design of Everyday Things. Basic Books (2002)
22. Novak, J.D., Cañas, A.J.: The theory underlying concept maps and how to construct and use them. Tech. rep., Florida Institute for Human and Machine Cognition (IHMC) (2008), `http://cmap.ihmc.us/Publications/ResearchPapers/TheoryCmaps/TheoryUnderlyingConceptMaps.htm`
23. Rouse, W.B., Morris, N.M.: On looking into the black box: Prospects and limits in the search for mental models. Psychological Bulletin 100(3), 349–363 (1986)
24. Ruge, L.: CoRAL – Context Reasoning and Adaptive Learning. Master's thesis, University of Lübeck (To appear)
25. Satyanarayanan, M.: Pervasive computing: Vision and challenges. IEEE Personal Communications 8(4), 10–17 (August 2001), `citeseer.ist.psu.edu/satyanarayanan01pervasive.html`
26. Schank, R.C.: Dynamic Memory: A Theory of Reminding and Learning in Computers and People. Cambridge University Press, Cambridge (1983)
27. Scheele, B., Groeben, N.: Die Heidelberger Struktur-Lege-Technik (SLT). Eine Dialog-Konsens-Methode zur Erhebung Subjektiver Theorien mittlerer Reichweite. Beltz, Weinheim/Basel (1984)
28. Wegener, R., Cassens, J., Butt, D.: Start making sense: Systemic functional linguistics and ambient intelligence. Revue d'Intelligence Artificielle 22(5), 629–645 (2008), `http://dx.doi.org/10.3166/ria.22.629-645`
29. Weiser, M.: The computer for the 21st century. Scientific American pp. 94–104 (September 1991)
30. Wilson, J.R., Rutherford, A.: Mental models: theory and application in human factors. Human Factors 31(6), 617–634 (1989)
31. Zacarias, M., Pinto, H.S., Tribolet, J.: Automatic discovery of personal action contexts. In: Kofod-Petersen, A., Cassens, J., Leake, D., Zacarias, M. (eds.) HCP-2008 Proceedings, Part II, MRC 2008 – Fifth International Workshop on Modelling and Reasoning in Context. pp. 75–88. TELECOM Bretagne (June 2008)

Author Index