

MASTER

Context awareness in ambient intelligence surroundings

van den Heuvel, H.A.C.

Award date:
2007

[Link to publication](#)

Disclaimer

This document contains a student thesis (bachelor's or master's), as authored by a student at Eindhoven University of Technology. Student theses are made available in the TU/e repository upon obtaining the required degree. The grade received is not published on the document as presented in the repository. The required complexity or quality of research of student theses may vary by program, and the required minimum study period may vary in duration.

General rights

Copyright and moral rights for the publications made accessible in the public portal are retained by the authors and/or other copyright owners and it is a condition of accessing publications that users recognise and abide by the legal requirements associated with these rights.

- Users may download and print one copy of any publication from the public portal for the purpose of private study or research.
- You may not further distribute the material or use it for any profit-making activity or commercial gain

*University of Technology Eindhoven | Computer Science department
Philips Research | Media Interaction group*

Context Awareness in Ambient Intelligence Surroundings



Student: Herjan van den Heuvel | 508407

Philips supervisor: Mark van Doorn

TU/e supervisors: Jeen Broekstra

Lynda Hardman

Eindhoven, August 8, 2007

A Abstract

In most ambient intelligence scenarios systems need to be aware of their context. This paper reports on the development of a context awareness component for the DreamScreen project. This component includes a context model that will be used by the existing Ambient Narrative system, which activates and deactivates output fragments (beats) depending on the context of the system.

In the first part we present an extensive overview of context usage in future ambient intelligence scenarios and existing systems within the home and retail domains. This broad list of context information not only shows us that the current context of a system is important but also that the context of the past is valuable information.

In the second part we describe the design of a context model that captures all the identified context requirements. We come up with a new way to store and use context history in an effective way. Next to this we propose a new format for the preconditions of beats, in order to be compatible with our context model. Finally we design a software component that checks these preconditions on the context model and context history.

In the third part we deliver an implementation of the design of our context model, context history model and software component, in order to be integrated in the Ambient Narrative system. Test results show that our context model is a powerful and expressive model that covers a broad area of ambient intelligence scenarios.

B Contents

A	ABSTRACT	2
B	CONTENTS	3
1	INTRODUCTION.....	5
1.1	BACKGROUND.....	5
1.1.1	<i>Ambient intelligence</i>	5
1.1.2	<i>Context awareness</i>	6
1.1.3	<i>DreamScreen project</i>	6
1.1.4	<i>Ambient Narrative system</i>	7
1.2	PROBLEM STATEMENT	8
1.2.1	<i>Research questions</i>	8
1.3	SCOPE	9
1.4	PREVIOUS WORK.....	9
1.5	STRUCTURE OF THIS REPORT.....	10
1.6	ABBREVIATIONS AND DEFINITIONS	11
2	REQUIREMENTS	12
2.1	CONTEXT MODEL.....	12
2.1.1	<i>General requirements</i>	12
2.1.2	<i>Literature study</i>	13
2.1.3	<i>Scenario analysis</i>	13
2.1.4	<i>Brainstorm</i>	14
2.1.5	<i>Taxonomies</i>	14
2.1.6	<i>Overview</i>	17
2.2	CONTEXT HISTORY	20
2.2.1	<i>Previous work</i>	20
2.2.2	<i>General requirements</i>	20
2.3	SYSTEM REQUIREMENTS.....	22
2.3.1	<i>Functionality</i>	22
2.3.2	<i>Operating environment</i>	22
2.3.3	<i>Design/implementation constraints</i>	22
2.3.4	<i>Interfaces</i>	23
2.3.5	<i>Non-functional requirements</i>	23
2.4	ASSUMPTIONS.....	23
3	DESIGN	25
3.1	CONTEXT MODEL.....	25
3.1.1	<i>Modelling issues</i>	25
3.1.2	<i>Conceptual model</i>	29
3.1.3	<i>Formal aspects</i>	32
3.1.4	<i>Coverage</i>	33
3.1.5	<i>Stored parameters</i>	34
3.1.6	<i>Derived parameters</i>	34
3.1.7	<i>Extensibility</i>	35

3.2	CONTEXT HISTORY	35
3.2.1	<i>Modelling solution</i>	35
3.2.2	<i>Conceptual model</i>	36
3.3	PRECONDITIONS MODEL	36
3.3.1	<i>Modelling issues</i>	37
3.4	SOFTWARE COMPONENT	41
3.4.1	<i>Precondition checking</i>	41
3.4.2	<i>Position computations</i>	44
3.4.3	<i>Relations and variables</i>	46
3.4.4	<i>Extensibility</i>	46
4	IMPLEMENTATION	47
4.1	MODELLING LANGUAGE	47
4.2	CONTEXT MODEL	48
4.3	CONTEXT HISTORY	49
4.4	SOFTWARE	50
5	EVALUATION	52
5.1	USER TEST	52
5.1.1	<i>Experiment</i>	52
5.1.2	<i>Results</i>	54
5.2	SOFTWARE TEST	55
5.2.1	<i>Test scenario</i>	55
5.2.2	<i>Test tool</i>	57
5.2.3	<i>Results</i>	59
6	CONCLUSIONS	60
6.1	CONTEXT	60
6.2	CONTEXT MODEL	60
6.3	FUNCTIONALITY	61
6.4	FUTURE WORK	62
C	REFERENCES	63
C.1	REFERENCES	63
C.2	AMBIENT INTELLIGENCE SCENARIOS	67
D	APPENDICES	69
D.1	BRAINSTORM RESULTS	69
D.2	MOVIE SCRIPTS ANALYSIS	71
D.3	HOME SCENARIOS	73
D.4	RETAIL SCENARIOS	74
D.5	XML SCHEMA	75
D.6	USER TEST SCENARIOS	78
D.7	SOFTWARE TEST SCENARIOS	80

1 Introduction

User system interaction is an important research topic nowadays. One of the aspects of this research is focused on implicit interaction. This includes sensing users, user needs, environmental variables and situations, and with this knowledge adapting computer systems and environments to the user, without asking for explicit input from this user.

Applications can be found in many fields like at home, in the office, in retail, hospitality, health care, learning, travel and sports. A very simple example of implicit input is a light connected to a motion detector, that is switched on for a moment when someone is nearby. This technology of systems sensing what is happening around them can be referred to as context awareness.

The work described in this report is taking place in the DreamScreen project. This project explores ambient intelligence in the retail domain and tries to find ways to realize large varieties of ambient intelligence surroundings in a simple and fast way. The project is focused on interactive solutions with augmented windows and is guided by a single carrier application which is the intelligent shop window. Context awareness is a key characteristic of this application and therefore an extensive model is needed to store all kinds of context information in.

1.1 Background

1.1.1 Ambient intelligence

The term ambient intelligence was coined by Philips to express their vision on the future of human computer interaction. They believe that “in the year 2020, people will relate to electronics in more natural and comfortable ways as we do now. Current inventions will make electronics smart and technological breakthroughs will allow us to integrate these smart electronics into more friendly environments.” [Philips 07]

This is the vision of ambient intelligence: “In an ambient intelligence world, devices work in concert to support people in carrying out their everyday life activities, tasks and rituals in easy, natural way using information and intelligence that is hidden in the network connecting these devices. As these devices grow smaller, more connected and more integrated into our environment, the technology disappears into our surroundings until only the user interface remains perceivable by users.” [Wiki: AmI]

Close to the term ambient intelligence are the terms pervasive computing and ubiquitous computing. The goal of pervasive or ubiquitous computing is to bring computer usage into the real physical world and allow users to interact with these computers in a more natural way by talking, moving, pointing and gesturing. [Meyer 03] [Coen 98] This is the technological side, while ambient intelligence is focused more on the design side.

1.1.2 Context awareness

In the ambient intelligence vision, systems have to be “sensitive to people's needs”, “anticipatory of their behaviour” and “responsive to their presence”. These statements are all examples of context awareness. A context aware system knows what is happening in its environment; its context. This context is not limited to users or devices, but also includes other environmental variables like temperature, time and light conditions.

Dey defines context as “any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.” [Dey 01]

A context aware system will use this information to respond to situations and adapt to users' needs, without explicit input from these users.

1.1.3 DreamScreen project

In the DreamScreen project at the Media Interaction group of Philips Research Eindhoven, research is done on how to realize a large variety of ambient intelligence surroundings in a relatively fast and cost effective way. Such a fast and cost effective method is necessary to produce large amounts of customized solutions both for professional (hotels, shops, airports) as well as consumer domains (home).

In the DreamScreen project research is done on different areas. These areas include research on sensors, like position tracking and gaze detection, research on user interaction, with augmented windows, and research on end-user programming. [Aarts 06]



Figure 1: The intelligent shop window.

An intelligent shop window in the Philips ExperienceLab is used as a carrier application to physically test the concepts developed in this project. This shop window is equipped with different kinds of sensors to detect for example people passing by or people looking at certain products. At the output side, the shop window has transparent video screens, localized audio speakers and lots of lights to respond to these situations, for example by lighting up a product or by showing product information on the window.

1.1.4 Ambient Narrative system

As a solution to realize a large variety of ambient intelligence surroundings in the desired fast and cost effective way, a mass customization method is developed.

To this end the ambient intelligence experience, that is the output of the system, is broken up in a large amount of small related fragments which are assembled in a custom experience, based on the current situation, i.e. the current context. These small fragments are called “beats” and have a precondition and an action part. The precondition part can put constraints on the context like “there have to be at least two persons in this area” or “the sun has to shine”. The action part defines what happens when the beat is activated. These beats are read by the Ambient Narrative system.

The Ambient Narrative system is the engine that manages all the beats, checks their preconditions on the current context and if a precondition holds, executes the action parts. An action part can contain statements that trigger output devices such as displays, audio speakers, lights, and etcetera. To this end a “Physical Markup Language” is under development that, in the near future, will enable us to control any electronic device. In figure 2 the Ambient Narrative system is shown as the core of the DreamScreen architecture.

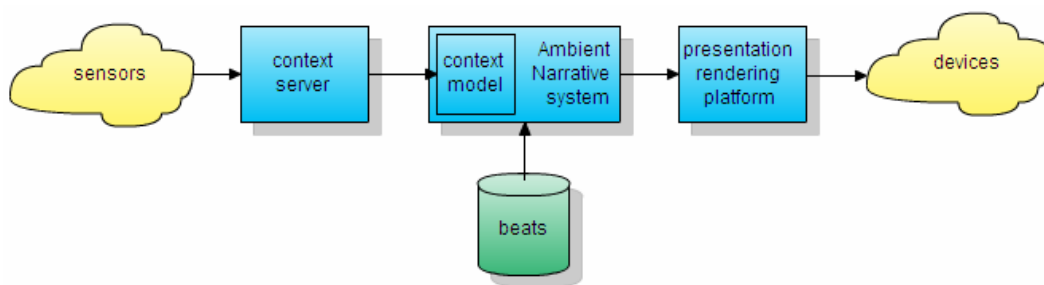


Figure 2: The DreamScreen architecture with its Ambient Narrative system.

When the Ambient Narrative system starts, an initial set of beats is loaded. The Ambient Narrative system continuously checks the preconditions of these beats. The action part of a beat can not only contain output actions, but it can also contain statements that cause other beats to be added or retracted from the beat set. Now beats can trigger other beats and this way beats can be hierarchically linked. The ambient intelligence experience can now be seen as an interactive narrative that people explore, or a hypermedia network that people browse through, by interacting with their environment.

1.2 Problem statement

The Ambient Narrative system continuously uses context information, so one of the most important aspects involved in this system is context awareness, i.e. the collecting and representing of context information.

To represent this context information, a model is used. In the current model it is possible to describe several different situations, but not as much as desired. Aspects like a user's orientation or the distance between devices and a user are not supported, for example. The object of our project within the DreamScreen project is to find a better context model which makes it possible to specify in more detail where users, objects and devices are and what their relationships are.

To come to this new context model, we first have to find out what the criteria are and which context information has to be represented by the model. This will be done by the analysis of literature, scenarios and examples of ambient intelligence surroundings and existing systems. With these criteria and the knowledge of the Ambient Narrative system, we will construct a new extensive context model that fits in the existing system.

1.2.1 Research questions

To guide this context representation problem we formulate three research questions. First we will try to find out which context information aspects are necessary for ambient intelligence scenarios and therefore have to be in our context model.

Q1: What kind of context information is needed in ambient intelligence surroundings?

Knowing which information has to be represented by our model, we have to come up with an efficient but also simple design for this model:

Q2: How can we efficiently represent this context information in a model?

Now what is the best way to implement this design so that it can easily be integrated in the existing system:

Q3: How can this model be used in the existing Ambient Narrative system?

1.3 Scope

To answer the research questions presented in the previous section, our research project focuses on context awareness in ambient intelligence surroundings. This report is written as a record of our search for a new context representation model, including its requirements, design, and implementation plus the implementation of a tool to test this model.

We limit our research to context representation for ambient intelligence in the home and the retail domains. We choose the home domain because most existing ambient intelligence scenarios are written for this domain. We choose the retail domain because it is expected that companies in this domain will be among the first adopters of ambient intelligence concepts.

1.4 Previous work

Mark Weiser was the first to write about pervasive and ubiquitous computing. In 1991 Weiser wrote about “integrating computers seamlessly into the world at large”. [Weiser 91] After this publication much research is done on these subjects. In [Meyer 03] an overview of previous research on context awareness in homes is presented.

Location awareness

In [Abowd 00] it is stated that “most context aware systems still do not incorporate knowledge about time, history, other people than the user, as well as many other pieces of information often available in our environment.” Indeed a lot of context aware applications focus solely on position information, i.e. use no other context information. Examples of location aware systems are [Hansen 04] and [Becker 05]. In [Satoh 05] a location model is presented and implemented that can work distributed over multiple computers.

Context fusion

“Context fusion assists in providing reliable context by combining sensors in parallel to offset noise in the signal, and by combining sensors sequentially to provide greater coverage.” [Abowd 00] We believe this is true but we will not focus on context fusion in this project. In [Schmidt 99] some basic experiments are done with context fusion, showing its importance and strength.

Context reasoning

A lot of work on context awareness is focused on logic-based context reasoning. To this end most papers propose RDF and OWL based models. Examples are [Kleinhou 03] and [Wang 04]. Also much research is done on reasoning by artificial intelligence, as stated in [Meyer 03]. Reasoning is out of the scope of this project, but can be an important aspect in the Ambient Narrative system in the future.

Context representation

In [Abowd 00] we can read that “the evolution of more sophisticated representations will enable a wider range of capabilities and a true separation of sensing context from the programmable reaction to that context.” [Hull 97] is one of the first to propose a context model using XML. Also [Schmidt 99] and [Schmidt 00] introduce a context model in XML and a working application of this context aware system. [Dogac 03] proposes the use of ontologies, but they only focus on security and privacy. [Ferscha 02] proposes the use of RDF to represent context information because of its simple but powerful syntax definition. [Shehzad 04] and [Hung 05] describe a context aware middleware architecture including context fusion, reasoning, context history and domain ontologies. This system uses a relational database. [Przybilski 05] provides a broadly applicable context representation and reasoning framework, but does not go into detail on the context model itself.

Adaptive hypermedia

In [Romero 03] a context aware hypermedia system is proposed to implement mixed reality. [Brusilovsky 96] states that “an adaptive hypermedia system (...) should satisfy three criteria: it should be a hypermedia system, it should have a user model, and it should be able to adapt the hypermedia model using this user model.” In [Doorn 05] the Ambient Narrative system is introduced, having an architecture based on the Dexter reference model [Halasz 94] for hypermedia systems.

The Ambient Narrative system is an adaptive hypermedia system; it is a hypermedia system with separate pieces of distributed multimedia output linked together, it has a user model, being the context model and it adapts its hypermedia by this model. More on the hypermedia aspects of the Ambient Narrative system and the ambient intelligence experiences it creates can be found in [Doorn 06].

1.5 Structure of this report

We will give an overview of the report structure here. In section 2 we perform an extensive search for requirements by analysing literature, ambient intelligence scenarios and examples of ambient intelligence systems.

Section 3 contains the design of the context model considering the requirements found in the preceding section. This includes the design process, problems that had to be solved and the final conceptual model. This section also covers the design of a new precondition format for the beats and of the software component that does the precondition checking on the context model.

An implementation of the context representation model, the beat precondition model and the precondition checking software component is presented in section 4.

In sections 5 we conduct a user test and software test and discuss the results of these verification and evaluation steps. We will conclude the report in section 6 by answering the research questions and proposing further research topics.

1.6 Abbreviations and definitions

Often used terms and abbreviations are defined here.

Ambient intelligence: A vision where people live “easily in digital environments in which the electronics are sensitive to people's needs, personalized to their requirements, anticipatory of their behaviour and responsive to their presence.” [Philips 07]

Ambient Narrative system: The main software component of the DreamScreen project that manages a set of beats and activates these beats depending on the current context.

Beat: The smallest unit of a narrative in the Ambient Narrative system, consisting of a precondition and an action part.

Context: “Any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and applications themselves.” [Dey 01]

Context awareness: The consciousness of an application of its context.

Hypermedia: Hypermedia is a term used for systems that contain graphics, audio, video and plain text, interconnected by hyperlinks to create a non-linear medium of information.

Narrative: A text composed in any medium which describes a sequence of events.

OWL: Web Ontology Language. OWL is a language for defining and instantiating ontologies. Ontologies are designed for use by applications that need to process the content of information instead of just presenting this information to humans. Ontologies provide greater machine interpretability of content than that supported by XML and RDF by providing additional vocabulary along with a formal semantics. [Wiki: OWL]

Pervasive computing: → Ubiquitous computing.

RDF: Resource Description Framework. RDF is a family of World Wide Web Consortium specifications, originally designed as a metadata model but which has come to be used as a general method of modelling information, through a variety of syntax formats. The RDF model is based upon the idea of making statements about resources in the form of subject-predicate-object expressions, called triples. [Wiki: RDF]

Taxonomy: A classification in an ordered system.

Ubiquitous computing: A vision where all computer hardware including input and output devices disappear in the environment.

XML: Extensible Markup Language. XML allows information to be encoded with meaningful structure and semantics that can be understood by computers and by humans.

2 Requirements

In this project a system component is to be built that will not serve actual users, but will be totally hidden for the end-user. It only has interfaces to other system components. Having no end-users does not mean that there are no user requirements. We can identify user requirements for the whole system which also count for this component.

The system we will build is a new concept and end-users are not yet identified. Therefore requirements will mostly come from researchers and developers.

Because the component to be built is basically a model representing context information, most requirements are about which context information should be represented by the model. This is why most requirements are actually criteria for the context model.

2.1 Context model

2.1.1 General requirements

The model to be made shall be as simple as possible, but not too simple. We do not need all kind of extra functionality that is never used. What we need is a most expressive model for situation descriptions that covers a sufficiently large set of ambient intelligence scenarios in the home and retail domains. What we do not need, for example, is the possibility to get a 3d visualization of the situation described by the model. This can be a nice feature, but is not needed at this moment, so it is not within the scope of this project.

The model shall be unambiguous. When putting new context information from a certain sensor into the model, it has to be exactly clear where to put this information. And in the other direction, when reading certain information from the model, its meaning has to be totally clear.

The model shall be extensible. It is easy to imagine that in this requirements stage we can not find exactly all context information that has to be in the model. So in a later stage, when a new sensor is added to the system, for example, it has to be possible to easily extend the model with new parameters.

The final and most important requirement is that the model shall be able to describe as many situations as possible in the home and retail domains concerning ambient intelligence scenarios. To find these situations, we use a number of techniques to identify requirements [Hatley 00]. The first activity is a literature study. Second comes a broad scenario analysis and finally we will do a brainstorm session. In the next paragraphs we will discuss these activities and their correlation.

2.1.2 Literature study

As stated above we started with a literature study. We did a search for literature on the topics of context awareness, ambient intelligence surroundings, and pervasive computing. We found books, reports and papers in the library of Philips Research, the ACM digital library, the IEEE digital library and on the internet. While reading, we focused on context information parameters that were described or used. This way we came up with a first taxonomy of context parameters. Example context parameters are the “outside temperature”, a “person’s age”, a “person’s activity”, and a “device’s orientation”. The total list of literature can be found in appendix C.1.

2.1.3 Scenario analysis

For all kinds of projects, both within Philips Research as outside Philips, scenarios have been developed about future use of ambient intelligence. In most of these scenarios the systems are context aware to naturally interact with users. After collecting scenarios from different sources, we read and interpreted them one by one to fully understand what is written and what is meant in every scenario. Then we analyzed them very carefully to find all context information that would be necessary to realize these scenarios.

We matched all these context information parameters with the earlier constructed context taxonomy and, when our context taxonomy did not support a context parameter, it was added to the taxonomy. Example context parameters found in scenarios are the “air quality”, a “person’s mood” and the “object’s contents”. A complete list of scenarios used for our analysis can be found in appendix C.2. Now two matrix diagrams present the context taxonomies for respectively the home and retail domain and show which scenarios use which context information parameters. These matrices can be found in section 2.1.5.

Movie scripts

The ambient intelligence surroundings can be seen as an interactive narrative that people explore by interacting with their environment. Another form of narrative is film. Because of this similarity we looked at some movie scripts to possibly discover more context information. We have analyzed three different movies to cover different genres. The three movies are: *Slither* (horror / comedy) [Gunn 04], *V for Vendetta* (action / drama / thriller) [Wachowski 90] and *Willow* (fantasy / adventure) [Lucas 88].

From these movies we read different parts of the script. Then we indexed activities and conditions from the actors of which we thought that these activities should possibly be in our model. After having populated this list we searched in the context taxonomy for ways to detect every activity or condition. This list can be found in Appendix D.2. Example activities are “moving towards a door”, “laughing”, “being afraid” and “kissing”. Although we gathered lots of activities and situations, we did not find anything that can not be described by our context taxonomy. There are some difficult situations, but these can be detected by combining two or three context parameters.

2.1.4 Brainstorm

There is an enormous amount of literature and scenarios available for ambient intelligence in the home domain. For the retail domain there is less information. This is why we decided to set up a brainstorm session to think about ambient intelligence systems in retail environments.

Eight people participated, all directly or indirectly involved in the DreamScreen project. We already had a lot of scenarios about the outside of a shop, i.e. the shop window, and about expensive fashion shops, so we decided to focus on activities inside a common shop such as a supermarket. The result of this brainstorm is a list of 23 ideas of things that would be useful to have in such a shop or mall. We grouped these ideas by similarity into five clusters. For every cluster we listed the necessary context information, and if not already there, we added this to the context taxonomy. These lists can be found in appendix D.1. Example context parameters found by the brainstorm are a “person’s mental state” and an “object’s weight”. Treating every cluster as a scenario we added them to the context taxonomy for the retail domain.

2.1.5 Taxonomies

In this section we present the context taxonomies constructed by the activities of the previous sections. Table 1 contains the context taxonomy for the home domain and table 2 contains the taxonomy for the retail domain. These tables contain matrices with a scenario in every column and a context parameter on every row. This way it is easy to see in what scenario an how often a context parameter is used. The numbers in the header rows each represent a scenario. The list of scenarios can be found in appendices D.3 en D.4. Parameters without a mark were added to the taxonomy during the literature study, parameters marked by a * during the scenario reviews and parameters marked by a ° during the brainstorm session.

Persons	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32		
Identity	x	x	x		x										x				x	x	x			x	x		x						x	
Age																																	x	
Gender																																	x	
Location																																		
<i>absolute position</i>			x	x											x	x						x	x		x	x	x	x		x	x			
<i>relative position</i>			x							x					x	x	x	x						x	x							x		
<i>which room</i>		x	x	x						x					x	x						x				x		x						
<i>same room</i>																				x													x	
Orientation	x			x																					x	x	x							
Movement																																		
<i>direction*</i>		x	x	x							x														x		x						x	
<i>speed</i>		x	x	x							x														x		x							x
<i>acceleration</i>			x								x																							x
Activity	x		x	x											x	x				x	x	x	x			x		x						
Activity duration	x																																	
Gestures		x			x							x															x							
Speech																																		
<i>self</i>		x	x	x		x				x		x	x						x	x	x			x										x
<i>someone else</i>			x							x					x																			
Face*														x																				
Health	x																																	
<i>heart rate</i>											x											x												

2.1.6 Overview

For all context information parameters listed in the taxonomies in the previous section, there has to be a possibility to store them in and retrieve them from the context model. Therefore all these context parameters are requirements on our model. We will give a summary and analysis of these context taxonomies here.

Home domain

32 scenarios

7,3 parameters per scenario on average

Context parameters per scenario

- 11 scenarios have ≤ 5 parameters
- 17 scenarios have 6–10 parameters
- 2 scenarios have 11–15 parameters
- 1 scenario has 16–20 parameters
- 1 scenario has ≥ 21 parameters

69 unique parameters in total

3,4 scenarios per parameter on average

Scenarios per context parameter

- 28 parameters occur in 1 scenario
- 28 parameters occur in 2–5 scenarios
- 8 parameters occur in 6–10 scenarios
- 5 parameters occur in ≥ 11 scenarios

Top 10 context parameters

1. Entities nearby a device
2. Identity of a person
3. Absolute position of a person
4. Speech of a person
5. Activity of a person
6. Identity of a device
7. Relative position of a person
8. In which room is a person
9. Touch of a person
10. Capabilities of a device

Coverage

- 70% of the parameters cover 80% of the scenarios.
- 80% of the parameters cover 90% of the scenarios.

Retail domain

32 scenarios

4,5 parameters per scenario on average

Context parameters per scenario

- 25 scenarios have ≤ 5 parameters
- 6 scenarios have 6–10 parameters
- 0 scenarios have 11–15 parameters
- 1 scenario has 16–20 parameters

55 unique parameters in total

2,6 scenarios per parameter on average

Scenarios per context parameter

- 25 parameters occur in 1 scenario
- 26 parameters occur in 2–5 scenarios
- 3 parameters occur in 6–10 scenarios
- 1 parameter occurs in ≥ 11 scenarios

Top 10 context parameters:

1. Relative position of a person
2. Orientation of a person
3. Gaze of a person
4. Identity of a person
5. Absolute position of a person
6. Movement of a person
7. Gestures of a person
8. Touch of a person
9. Relative position of a device
10. Identity of a device

Coverage

- 60% of the parameters cover 80% of the scenarios.
- 75% of the parameters cover 90% of the scenarios.

2.1.6.1 Communalities and differences

Some context parameters are only used in a single domain, others are used both in the home domain as in the retail domain. In table 3 an overview of these numbers is presented. In the “intersection” row the numbers of shared context parameters are presented. In the “union” row the total numbers are listed, that is the shared parameters plus the unique home parameters plus the unique retail parameters.

	Context parameters			
	From literature	From scenarios	From brainstorm	Total
Home	48	69 (21 new)	-	69
Retail	33	45 (13 new)	32 (9 new)	55
<i>Intersection</i>	24	32 (8 new)	-	32
<i>Union</i>	57	83 (27 new)	32 (8 new)	92

Table 3: Numbers of context parameters per domain and totals.

2.1.6.2 Completeness

We did a literature study, scenario reviews and a brainstorm. Through these activities we managed to find a total of 92 unique context information parameters. Table 4 shows the numbers and percentage per activity. This shows us that already 62% of the context parameters were found in the literature study, and that the brainstorm session only added 9% to the total amount. Of course these amounts depend on the order of the activities. In the brainstorm session, for example, we actually found 35% of the context parameters, but most of them were already discovered during the literature study or scenario analysis. Given the fact that the brainstorm, being the third activity, only added a few new context parameters, we conclude that the solution space is not infinitely big, and that we found a very large part of the context parameters that are interesting for our application. With this set of parameters it should be possible to cover a sufficiently large set of scenarios.

	Context parameters			
	From literature	From scenarios	From brainstorm	Total
Totals	57	83	32	
%	62%	90%	35%	
Totals new	57	27	8	92
%	62%	29%	9%	100%

Table 4: Percentages of found context parameters.

2.1.6.3 Summary

As presented in the previous section, we have 92 unique context information parameters in our combined home and retail context taxonomy. We can reduce this to 80 by merging the “devices” with the “objects” group in our taxonomy and by merging overlapping parameters like a person’s “emotion” and “face expression”. Now the top 15 of most used context parameters in the combined home and retail domain, and thus the high priority top 15 of requirements for our context model, looks like this:

1. The relative position of a person
2. The identity of a device
3. The identity of a person
4. The absolute position of a person
5. The relative position of a device
6. Entities nearby a device
7. Speech of a person
8. Touch of a person
9. The absolute position of a device
10. The orientation of a person
11. The activity of a person
12. Contents of device
13. Movement of a person
14. Capabilities of a device
15. Gestures of a person

2.2 Context history

In the search for requirements for our model, we found eight context information parameters concerning history. This is 9% of all context information parameters. Examples are “persons met”, “past activities”, “products bought” and “time spent”. Actually these parameters are not really part of the context; in fact they are stored context information from the past. This is why these parameters will not fit in the context model, but a separate history model has to be developed.

This context history model has some extra requirements. First of all it shall support the context history information parameters found in the previous section. But actually it is more useful to store all context information. This can, for example, support data mining and pattern recognition in a later stage.

2.2.1 Previous work

We did research on existing solutions using context history. Although several papers describe it as very important en powerful, we found very few implementations of systems using context history. The system in [Spence 05] uses context history but is quite specific and can only be used in that system. [Salber 98] has a working system. It adds a timestamp to every context parameter en stores this together with the context. This works fine for very small systems, but with lots of sensors and lots of new data from these sensors, this will not work because of the enormous amount of data. In [Byun 04] a successful experiment is done on context history. They state that “in order to provide ‘intimate’ and ‘dynamic’ adaptations under Weiser's vision for ubiquitous computing environments, we propose the utilization of context history together with user modelling and machine learning techniques” [Byun 04] and “we believe that context history has a concrete role to play in supporting proactive adaptation in a ubiquitous computing environment.” It is not known whether this prototype is scalable to large applications.

2.2.2 General requirements

We will now address some issues concerning the use of context history. These issues will significantly influence the requirements on our history model.

2.2.2.1 How to use context history?

Context history will be used in the preconditions of beats in almost the same way as normal context. This means that these preconditions can put constraints on events in the history. Context history offers even more possibilities. Where the current context is a model of exactly one moment in time, context history represents lots of moments, so in the preconditions on context history, we can also use aggregative constraints, such as the number of occurrences of certain events or the average number of certain values.

What do we want to use in the scenarios? In the requirements we can find these history parameters: From a person's history: "persons met", "locations", "activities", "music heard", "food eaten", "products bought", and "time spent". And from a device / object: "has contained". Furthermore we could do pattern recognition on the context history to find certain habits. These habits can then be stored in a user profile. This pattern recognition itself is out of the scope of this project, but it can be interesting in the near future.

2.2.2.2 Which information to store?

We can limit our context history to the parameters listed in the previous section, but with the reasoning and pattern recognition in mind, we actually want to keep all context information. This raises some problems. Say we store the whole context file every second. That is $60 \times 60 \times 24$ times a day, which is 2.6 million times a month. Some experiments with extensive and rich context files showed that 50 Kb is a reasonable upper bound for the context file size of a small system. This means that in a month we get a context history of 130 Gb. Retrieving information from such an amount of data will take much too long. A possible solution is not to save everything every second, but only important changes like entities entering or leaving a room. This means that lots of things, like changes of position, are ignored. That is not what we want, so we have to find another solution for the storage problem.

2.2.2.3 When to store?

Related to which context information to store is when to store. We just mentioned to store the context every second. Another solution can be to store the old context every time a new context gets active. So every time the context server sends new context data to our model, the old contents of the model are stored in the context history database. But with lots of independent sensors, this will probably give us even more data. So also the timing of storing context history is part of the problem.

2.2.2.4 For how long to store?

We could choose to store context history for ever, but this will produce an enormous amount of data which will probably almost never be used. So there has to be a limit in the form of some expiration date on the information. Sometimes we still would like to save interesting information for ever though. A solution to this is to do some reasoning and pattern recognition on the context history and to store this derived information in a user profile of persons, devices or environments. This way interesting information is saved but the whole context history can be discarded.

We see that the most important requirement will be on storage size and related to that on speed. We can imagine that the system will have tens, maybe hundreds of sensors, and that all these sensors update their data very often, every millisecond for example. This will not only raise storage problems, but also problems with real time searching in this context history by the Ambient Narrative system. We can conclude that a smart storage algorithm is required that dramatically reduces the size of the context information to support quick context history queries.

2.3 System requirements

2.3.1 Functionality

The main functionality of the system will be described in this section. This includes precondition checking on the context model, precondition checking on context history, position computations and extensibility.

The software component to be made around the context model will receive a beat's precondition as input and shall produce "true" or "false" as output. To this end, this software component shall use the context model and the context history model.

The software component shall be able to do position computations on different kind of location information formats. This way it can match position information from preconditions with position information from the context model, without the need to have the same format.

The software component shall automatically adapt itself to the context model when the model is extended with new context information parameters.

2.3.2 Operating environment

This section describes the environment of the software component and the influence of this environment on the component and its requirements.

As stated before the product to be made is a component of a larger system, which is the Ambient Narrative system. Figure 2 in section 1.1.4 shows a diagram of the major components of the DreamScreen architecture. As can be seen in this figure the context model will be part of the Ambient Narrative system. This system has interfaces to the context server on one side and to the rendering platform on the other.

During use the context server will continuously place new data into the context model. And the Ambient Narrative system will continuously check preconditions of beats on the context and context history.

2.3.3 Design/implementation constraints

There is one requirement on the implementation. It concerns the programming language. To make the component compatible with the existing Ambient Narrative system, we are bound to Sun's Java language. For the context model, we are not restricted to a certain modelling language.

2.3.4 Interfaces

The software component has no end-user interfaces and no hardware interfaces. The context model has a software interface with the context server for which a communication protocol should be developed. This is out of the scope of this project.

The software component has a software interface with the Ambient Narrative system. Because our component will be directly embedded in the Ambient Narrative system, this system will directly call the functions of our software component.

2.3.5 Non-functional requirements

The output of the software component shall be correct. If and only if a beat's precondition holds, the output will be "true"; otherwise the output will be "false".

Performance requirements, safety requirements and security requirements are not within the scope of this project.

2.4 Assumptions

This section lists important assumptions on the context, the context model, beats and the software.

Model

The context model is a reference model representing real world situations. This model is hidden in the software and will not be directly used by humans.

Context sensing

Before the Ambient Narrative system, there will be a context server that merges and filters context information from the sensors before it is passed to our model.

The context information data from the context server and its sensors is correct and complete. Correct means: there is no false data. Complete means: everything that can be sensed is present and there is no context information that is not sensed.

In the case of a sensor failure, the context information from this sensor will not be present in the context model. Therefore all beats that have a precondition on this context information can not be activated.

It is assumed that all necessary sensors are present. This requirement can be met by the system through the presence of an authoring tool, which is used by end-users to create new beats. This authoring tool will only allow preconditions on context information parameters for which the required sensors are available.

Beats

Users do not have explicit control over the hypermedia structure of the beats, but they implicitly browse through it by doing things that change the context of the system. This can be, for example, walking around, sitting down, pointing at something, etcetera.

At any given time, more than one beat can be active. In hypermedia style; you can browse more than one node at once.

In the authoring tool, beats are given a priority. This can be done automatically or manually by the author.

The authoring tool detects conflicts while programming new beats. Conflicts can occur within one beat's precondition, but also between different beats. The authoring tool should force all conflicts to be solved, before accepting a new beat.

Within a hypermedia system, there is the possibility of deadlocks. When adding new beats or removing existing beats, the authoring tool should check the new set of beats on deadlocks and if necessary give a warning or suggest a solution.

The hypermedia system of beats is scalable. Precondition checking can be done very efficient and fast, and by making use of the triggers between beats, the set of beats that has to be checked can be small while the total set of beats can be large.

Software

Within this project, we do not consider sensors or sensor data at the input side of the context model, but we will use given context information.

Within this project, we do not use the action parts of the beats but we will only check their preconditions and return "true" when the precondition holds and the beat should be activated, or "false" if not.

At this moment we do not think about speed or optimization of the software. These aspects are important in the Ambient Narrative system, but are not within the scope of this project.

3 Design

In the previous section we have described our requirements. In this section we present the design of our context model and related components like the beats' precondition model, context history model and the software component that checks the preconditions on the context model and its history.

3.1 Context model

3.1.1 Modelling issues

Previous research work has shown that a distinction among the abstract classes of person, thing and place is useful and sufficient when we want to map real world objects to objects in a virtual environment [Ferscha 02]. We started the modelling phase by taking some scenarios and their context parameter use from the requirements phase. When thinking about how these scenarios can be modelled in the simplest way, we came to the following basic hierarchical model. Inspired by The Experience Economy [Pine 99] which states that “work is theatre and every business a stage”, the names of the entities come from the theatre world.

```
context
  stage // place or location
  performance // activity
    actor // person
    prop // device or object
  relation // relation between actors and props
```

Modelling some less straight-forward things needed a bit more attention. Issues were:

- How to model orientation
- How to model positions
- How to model emotions
- How to model capabilities
- How to model activities
- How to model relations
- How to model gestures

3.1.1.1 Orientation

One and the same problem arises when modelling orientation, relative positions or movement directions: In all three cases we have a starting position and we want to model a direction or angle from that position. There is a simple solution to model a direction; the 360 degrees scale, but the main problem is: what to define as the 0 degrees angle? We choose to let this be defined by the orientation of the anchor entity or, if this entity has no orientation or there is no anchor, by the parent entity, which is a “stage”. A stage always has an origin and x, y and z axes, and thus every entity can have an orientation based on

this system. This means that the 0 degrees angle of an object is not always explicitly defined; relative positions and directions have an anchor entity and use this entity's orientation as a reference.

Besides the angle, which is the rotation around the z axis, there are two more orientations. These will probably not be used for persons, but for objects they will. We define the "roll" as the rotation around the x axis and the "tilt" as the rotation around the y axis. These three axes are shown in figure 3.

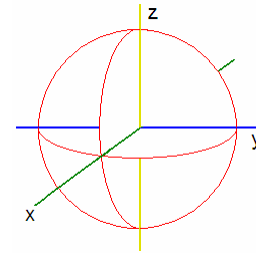


Figure 3: Three axes.

3.1.1.2 Positions

In the requirements we have found three kinds of positions; an absolute position, the distance to another object called relative position, and the room in which a person or object is. Starting with the third one, this can easily be derived from the model because every object or actor is in a certain "stage", which is a room or a part of a room. This leaves us with two kinds of positions: the absolute and the relative one.

Relative positions

A relative position is always relative to another object. We call this the anchor. From this anchor there is a distance and a direction, called angle. An example:

```
position
  class = relative
  anchor = table_1           // actor or object id
  distance = 200cm
  angle = 45°
```

Absolute positions

An absolute position uses a certain common system to give every location a unique identifier. This system however can vary. There are, for example, local systems and there are global systems like GPS. This system has to be known, because the units of the latitude and longitude can vary per system. Two examples:

```
position
  class = absolute
  system = xyz               // local system
  latitude = 800cm
  longitude = 200cm
```

```
position
  class = absolute
  system = gps               // global system
  latitude = 51°24'36"
  longitude = 5°27'28"
  altitude = 14500cm
```

Notice that both relative and absolute positions can have an optional altitude to define the height.

3.1.1.3 Emotions

Emotions and mood are important context parameters, because they really characterize the user. After having done some research on human emotions we know that basic emotions can be described by one word and that these emotions can easily be derived from a face expression. [Bielefeld] There are six basic facial expressions proposed by Ekman and Friesen [Ekman] conveying emotion. They are listed here:

- Anger
- Disgust
- Fear
- Happiness
- Sadness
- Surprise

Since these emotions can be described by one word, we will model emotion as the more generic “health” parameter. This parameter is used to describe, for example, a person’s heart rate, weight or blood pressure and has a “class” and “value” attribute. Using this parameter means that there is no need for a special emotion entity in the model. An example:

```
health
  class = emotion
  value = happiness
```

3.1.1.4 Capabilities

When a certain device is detected or needed for some scenario, one of the most important and interesting things to know are its capabilities. This includes things like communication protocols, display possibilities, etcetera. A problem of such capabilities is that modelling only the capability itself will not be enough. Some scenarios have certain requirements on these capabilities, for example on the size of a display, on the amount of speakers, on the brightness of a lamp, etc. To find output capabilities, we take the five human sense organs and from that we search for possible capabilities of devices and the parameters belonging to these capabilities:

Human	Solution	Solution’s parameters
See	screen	size, amount of colours, resolution
	light	colour, amount of colours, brightness, contrast
Hear	speakers	kind, amount, watt
Feel	force-feedback	strength
Smell	scent	odour
Taste	-	-

Next to these output capabilities, there are input capabilities like a touch-screen, keyboard, speech recognition, gesture controller, etc. We can conclude that modelling all these varying parameters will be too much, so we come up with another solution. We

offer the possibility to define keywords which can be put together to create the right description. Example keywords are: “small-screen”, “big-screen”, “white-light”, “big-speakers”, etc. To make this really work, these keywords should be linked to some kind of ontology to create a common understanding. For now, this is out of the scope of the project. An example:

```
capability  
class = big-screen, small-speakers
```

3.1.1.5 Activities

To come to a model for activities, we listed the activities of some scenarios and thought of a way to model these as simple as possible. Most activities are described by a verb and a starting time. Some activities need extra information though. This is the case with all kinds of movements like walking or running, because here we also have to model things like speed, direction and acceleration. Therefore these attributes will be available depending on the activity. Furthermore in every activity at least one actor or object is involved, possibly even more, so there has to be a possibility to add actors to an activity. Some examples:

```
performance  
activity = reading  
start = 21:48:37           // time stamp  
actor = ...                // actor
```

```
performance  
activity = running  
start = 14:36:52           // time stamp  
speed = 4 m/s  
angle = 10°  
acceleration = 1 m/s2  
actor = ...                // actor
```

```
performance  
activity = speaking  
start = 09:14:29           // time stamp  
words = hello              // spoken words  
actor = ...                // actor
```

3.1.1.6 Relations

Relations between actors and actors, between objects and objects and between actors and objects are very important context information. Examples of relation types are:

- touch
- gaze
- point
- speak
- hear
- smell

Having a possibility to define relations in our context model is a very powerful tool to model more complex situations and to group actors and devices together. An example:

```
relation  
  class = touch  
  object = jessica           // actor or object id  
  subject = tv_2            // actor or object id
```

3.1.1.7 Gestures

We did some research on gestures and listed the most common gestures below. In the model we split these gestures in to two groups: those with a subject and those without. The ones with a subject will be modelled as a relation, those without as an activity. This way we do not need a special gesture entity in the model.

Hand gestures

- shake hands relation
- shake activity
- wave activity / relation
- point relation
- clap activity
- thumbs up activity

Head gesture

- shake head activity
- kiss relation
- nod activity / relation

Body gestures

- bow activity
- jump activity

3.1.2 Conceptual model

After having solved all modelling issues of the previous section, the remaining requirements can be modelled as simple attributes belonging to certain entities. We construct a conceptual model covering all these requirements. This is a hierarchical model with several entities and attributes. This conceptual model is displayed on the next page. It includes entities (in bold), attributes and sub entities (in italic), together with their type, unit and value constraints.

	Type	Unit	Constraints
stage			
id	string	-	-
indoor_outdoor	string	-	"indoor" "outdoor"
<i>condition</i>	-	-	min=0 max=1
<i>performance</i>	-	-	min=0 max=unbounded
<i>relation</i>	-	-	min=0 max=unbounded
condition			
light	integer	lux	0 <= x
colour	string	sRGB	000 <= x <= FFF
sound_db	integer	dB	0 <= x
music_genre	string	-	"classic", "ambient", "dance"...
temperature	integer	°C	-
air_quality	integer	%	x <= 100
rain_chance	integer	%	x <= 100
water	string	-	"yes" "no"
performance			
activity	string	-	"sitting", "walking", "speaking"...
start	time	hh:mm:ss	00:00:00 <= x <= 23:59:59
speed	integer	m/s	-
angle	integer	°	-180 <= x <= 180
acceleration	integer	m/s ²	-
words	string	-	-
<i>actor</i>	-	-	min=0 max=unbounded
<i>prop</i>	-	-	min=0 max=unbounded
relation			
class	string	-	"touch", "gaze", "point"...
object	string	-	prop id actor id
subject	string	-	prop id actor id
actor			
id	string	-	-
face	URI	picture	-
age	integer	years	0 <= x
gender	string	-	"male" "female"
length	integer	cm	0 <= x
role	string	-	-
arrival	time	hh:mm:ss	00:00:00 <= x <= 23:59:59
<i>position</i>	-	-	min=0 max=1
<i>orientation</i>	-	-	min=0 max=1
<i>health</i>	-	-	min=0 max=unbounded
<i>intention</i>	-	-	min=0 max=1

prop

id	string	-	-
class	string	-	“book”, “phone”, “tv”, “table”...
colour	string	sRGB	000 <= x <= FFF
width	integer	cm	0 <= x
height	integer	cm	0 <= x
depth	integer	cm	0 <= x
weight	integer	kg	0 <= x
capability	string	-	-
<i>orientation</i>	-	-	min=0 max=1
<i>position</i>	-	-	min=0 max=1
<i>contains</i>	-	-	min=0 max=1

position

class	string	-	“relative” “absolute”
system	string	-	“xyz” “gps”
latitude	string	cm °	-
longitude	string	cm °	-
altitude	integer	cm	-
anchor	string	-	prop id actor id
distance	integer	cm	0 <= x
angle	integer	°	-180 <= x <= 180

orientation

angle	integer	°	-180 <= x <= 180
tilt	integer	°	-180 <= x <= 180
roll	integer	°	-180 <= x <= 180

health

class	string	-	“emotion”, “heart rate”...
value	string	-	-

intention

activity	string	-	-
duration	time	hh:mm:ss	00:00:00 <= x
speed	integer	m/s	-
angle	integer	°	-180 <= x <= 180

contains

in_order	string	-	“yes” “no”
count	integer	-	0 <= 0
<i>prop</i>	-	-	min=0 max=unbounded
<i>actor</i>	-	-	min=0 max=unbounded

3.1.3 Formal aspects

We can put our conceptual model in a UML format which gives a graphical overview of the total model. We have a model of 12 unique entities with a total of 54 attributes. These attributes have different types. Most common are the string type (26) and the integer type (23). Furthermore there are some attributes with a time stamp type (4) and URI type (1).

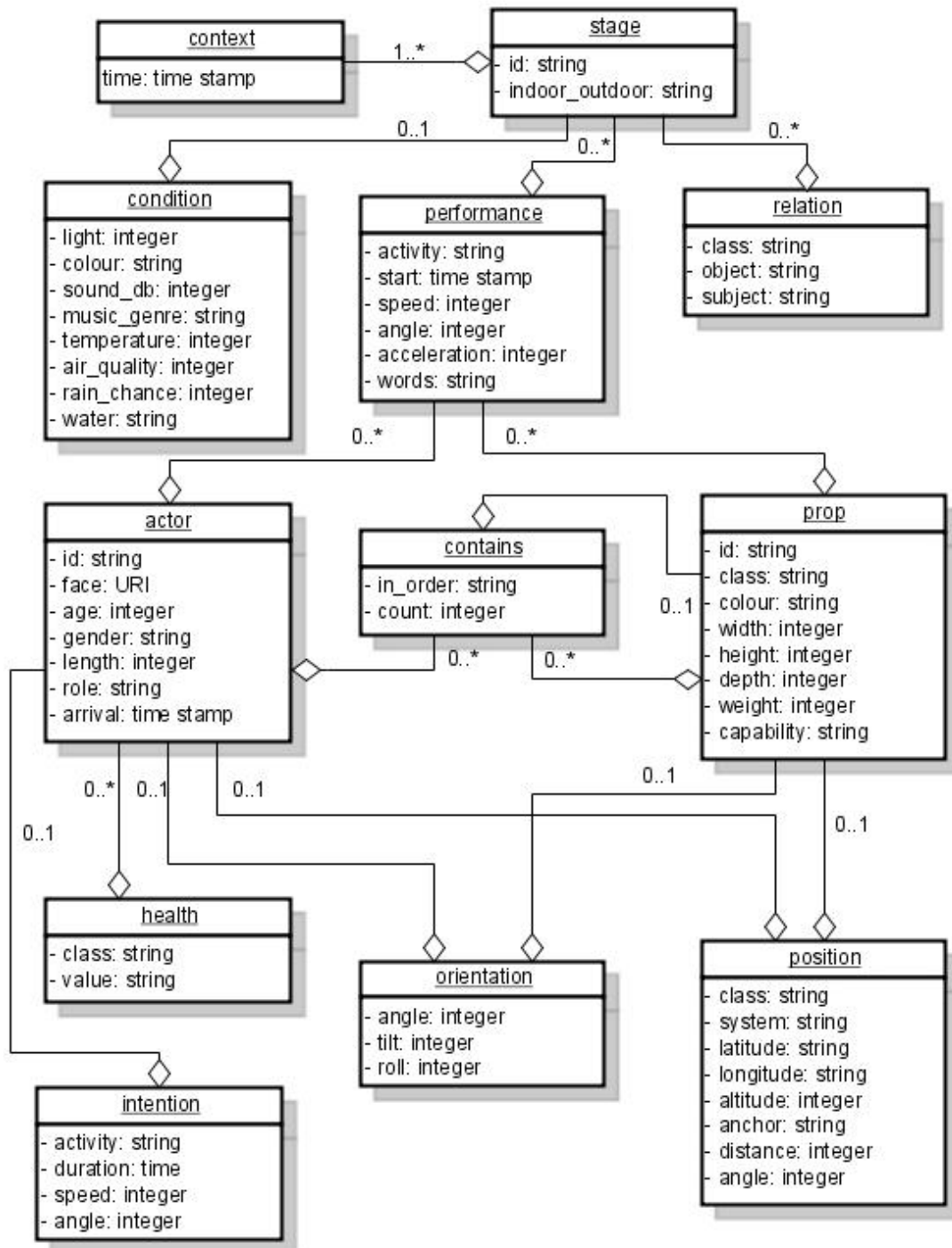


Figure 4: UML diagram of the conceptual context model.

3.1.4 Coverage

In this section we will discuss the coverage of the requirements by our conceptual model. Some context information parameters listed as a requirement in the requirements section are not included in the model. We will discuss these context parameters here and give an explanation of their absence.

Person's / Device's location: "nearby": This information is derivable from two positions. We can compute their inner distance and decide whether this is nearby or not. A second reason for not including this context parameter in our model is that there is no unique definition for "nearby".

Person's health: "mental state" and "physical state": Both these context parameters are very difficult to sense and, next to that, very difficult to describe.

Person's intention: "good/bad": This property of the intention is derivable from the intention's activity and therefore will not have to be modelled on its own.

Person's intention: "time to spend": This property of the intention is impossible to be sensed directly. It can only be found in a stored user profile, which can be derived from previous similar situations, i.e. from the context history.

Person's history: "persons met", "location", "activities", "music heard", "food eaten", "products bought" and "time spent": History parameters are not sensible context, but are stored information, i.e. context history. These context parameters do not belong to the context model, but have to be derived from stored context history.

Person's / Device's profile and rights: Profiles and rights are not sensible, but can be found in a stored user profile. Some aspects of such a profile can be derived from context history by data mining or pattern recognition, other aspects like the rights of a user will have to be programmed explicitly by the owner of the system.

Device's history: "has contained": Here the same applies as to a person's history; history parameters are not sensible context, but are stored context information.

Groups: All group parameters from our requirements can be derived from the distinct entities forming this group. Therefore no special "group" entity is needed in our model.

Environment's dimensions: As with profiles and rights, stage dimensions are not sensible, but stored information. Therefore every stage will have a profile in which its insensible properties like dimensions can be found.

Some of these parameters are actually not real context parameters. History, for example, is not something sensible, but is the context of the past. This also applies to user profiles. The user identity is sensible, but user's profiles, rights and preferences have to be retrieved from some kind of database. We see this information as some kind of background information of the context, that is stored context data to support the actual context. We will address these issues in the next section.

Looking back at the requirements of the model, we can conclude that most context parameters of the requirements are covered. Only from the required context parameters of the retail domain, two are missing. These are already mentioned above and are a person's mental and physical state. These two parameters are very difficult to sense and very difficult to describe. For this reason they are left out. A summary of the coverage is given in table 5.

Domain	Requirements	In model	Background information	Total	Not included	Coverage
Home	69	59	10	69	-	100%
Retail	55	45	8	53	2	96%
Total	92	75	15	90	2	98%

Table 5: Coverage of the requirements by our model.

3.1.5 Stored parameters

As mentioned above, some parameters are not real context but are stored information. One of the aspects of stored information are user profiles, which are not only real user profiles, rights and preferences but also profiles and rights of devices and properties of environments. Another aspect is context history, which will be an important part of this project. A third aspect of stored information is the set of currently active beats. The user profiles part is left out of the scope of this project for now. History issues will be addressed in section 3.2 and the active beat set will be handled in section 3.3.

3.1.6 Derived parameters

From the context parameters that are included in the model, some are not sensible by a single sensor. These parameters belong to the higher level information and should be derived from lower level information, i.e. from other context parameters. To derive this higher level information there has to be some kind of reasoning component with specific rules. This reasoning part is out of the scope of this project, but nevertheless these parameters are in our model. These parameters are listed here:

Parameter	Derivable from
Persons	
Role	activity, time, position, other actors/props, (or user profile)
Health / Emotion	face, heart rate, activity, music
Intention	activity, time, position, other actors/props, (or user profile, history)
Environments	
Indoor / Outdoor	light, temperature, pressure, humidity
Music genre	bpm, volume
Rain chance	temperature, pressure, humidity

3.1.7 Extensibility

Our model has to support easy additions of new entities and new attributes. When new information has to be represented in the model, we can just introduce new entities with new attributes or new attributes in existing entities. The only restriction here is caused by the software that will work with the model. This software shall be constructed in such a way that it can handle these new entities and attributes without the need to be rewritten.

3.2 Context history

An interesting context aspect is context history. Although much research is done on context awareness, not so much is done on context history. Context history issues we have seen in the requirements section are which context information to store, when, how often and for how long to do this, and how to use this information. The conclusion of the requirements was that a smart storage algorithm is needed to handle the large amount of context information.

3.2.1 Modelling solution

Since storing all context data is too expensive and space consuming we come up with a new solution using a different approach. Our solution stores the ids of active beats instead of the actual context. At first this seems strange, but let us take a closer look. For a beat to get active, its whole precondition has to match with the current context. This means that with one beat id, we can “store” a whole part of the context in our history, because we know the precondition of this beat. So every time a beat is activated, this is stored in the history and every time an active beat is deactivated, this is also stored. We now have a log file containing all beats with start and stop dates and times. So for every moment in the history, we can look up which beats were active. From this list we can simply compute the context at a specific moment, by combining all preconditions from these beats. These preconditions had to be true at that moment, so now we have our historical context. Of course this context history can be incomplete because not all context parameters have to be in the preconditions of the active beats.

There is even more we can do with this solution. Storing beat ids gives us another nice functionality. We can define a beat with a precondition matching exactly the context condition we want to store in the history. This beat can have an empty action part so there will be no output, but later we can search the history for this beat and then do something. This way an author of the system can exactly choose which events will be stored in the history. To make this history more useful we add an extra field to the history in which every beat can store some optional data. A beat that gets active at cash register transactions, for example, can store the customer id and a list of sold products in the history. This format makes our history not an absolute context history, but it certainly is a very generic history with lots of possibilities. If an author, for example, still wants to store

the whole context in the history, he simply defines a beat that gets activated at a certain time interval and that stores the whole context in the history.

3.2.2 Conceptual model

There are four parameters we want to store in the history:

- the beat id
- the activation date/time
- the deactivation date/time
- optional extra data

Two examples:

```
history
  beat
    id = cashRegister
    activated = 2007-02-12 14:15:16
    deactivated = 2007-02-12 14:15:37
```

```
history
  beat
    id = cashRegister
    activated = 2007-02-14 20:13:19
    deactivated = 2007-02-14 20:14:03
    data
      customer = 54643
      product = jeans
```

3.3 Preconditions model

In this section we describe the precondition model for the beats. Although the idea is to exactly match a precondition with the context to trigger a beat, this does not mean that preconditions use exactly the same syntax. The preconditions have almost the same format, but extended with some mechanisms to test values on ranges. In a precondition we can use upper and lower bounds for example, where in the context we have an exact value. Furthermore there is another important issue in preconditions. There has to be a possibility to test the presence / absence of other active beats. This can, for example, prevent a beat from starting twice at the same time. But it also makes sure that no two overlapping beats will be active simultaneously.

3.3.1 Modelling issues

The complete list of differences between the context model and the preconditions format are given in the overview below. Note that not mentioning an element in the precondition does not mean that an element may not be there, but that it means that it does not matter. Hence the absence rule.

	Context	Preconditions
Values	<exact value>	between <min> and <max>
Position	<exact position>	<area>
Absence	-	not <element>
Time	-	between <min> and <max>

Together with testing on other active or inactive beats and testing on context history, this gives us six issues to solve in the preconditions:

- min / max values versus exact values
- matching positions
- testing absence of context
- testing on time
- testing presence / absence of other beats
- testing on context history

Notice that the first three are tests on the context, the fourth is a test on the current time, the fifth is a test on the active beat set and the sixth on context history. In the next sections we will deal with all these issues.

3.3.1.1 Min / Max values

Most of the context parameters are numerical and have a value in a broad range. It is useless to test these parameters on a single value, because that will almost never lead to success. Instead we want to test these parameters on a certain range. This can be done by introducing extra parameters available in the precondition. For every context parameter of the type integer there will be two extra parameters in the precondition. These parameters have the same name as the original preceded by "min_" or "max_". This way the value can be bound to a certain range. When only one of these parameters is used, the range is only bound by this one value and open at the other end. An example:

```
actor
  min_age = 12
  max_age = 52
```

3.3.1.2 Positions

In the requirements phase we found that there is often something like a “nearby” relation. The problem with this, is that the definition of “nearby” is unclear and varies per scenario. For this reason we can not include this functionality as a relation. However the same functionality can be reached by using a relative position.

In the context there are two possible positions: relative and absolute. The absolute one will probably be used most of the time. In the preconditions however both notations will be used a lot. The relative position will be used to make constraints on the distances between two objects, and the absolute position will be used to define areas. This demands special functionality to compute relative positions from absolute ones and vice versa. Below we give some examples of an actor with a relative positions to define the “nearby” relation (figure 5a):

```
actor
  position
    class = relative
    anchor = table
    max_distance = 200cm
```

An optional addition is a minimum distance and a minimum and maximum angle (figure 5b):

```
min_distance = 150cm
min_angle = 90°
max_angle = 230°
```

Another addition is an offset from the anchor (figure 6):

```
actor
  position
    class = relative
    anchor = table
    max_distance = 200cm
    offset_angle = 45°
    offset_distance = 300cm
```

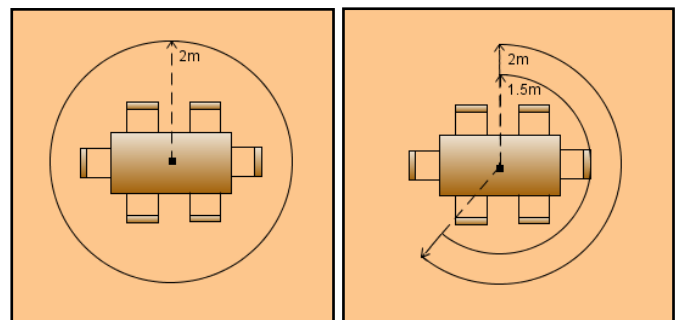


Figure 5a and 5b: Relative preconditions.

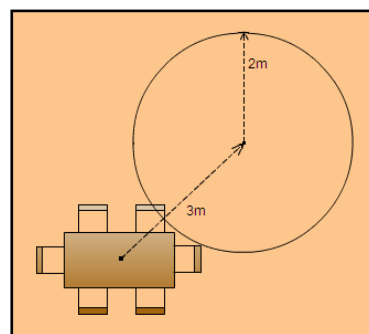


Figure 6: Relative precondition with offset.

Instead of a circular area we can also define a rectangular area around or nearby an anchor entity (figure 7a and 7b):

```
actor
  position
    class = relative
    anchor = tv
    width = 200cm
    length = 300cm

    offset_angle = 25°
    offset_distance = 300cm
```

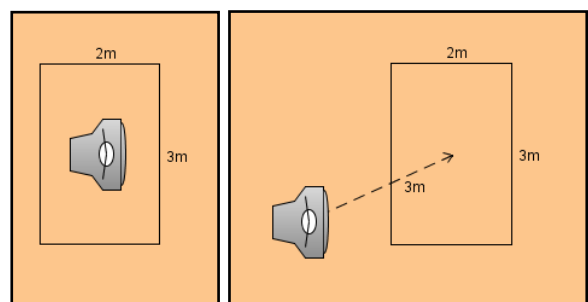


Figure 7a and 7b: Square relative preconditions.

The second possibility to define a position is by an absolute position, defining a fixed area. Below we give two examples of an actor with an absolute position. The first is a rectangular area (figure 8) and the second a circular area (figure 9):

```
actor
  position
    class = absolute
    min_latitude = 200cm
    max_latitude = 800cm
    min_longitude = 0cm
    max_longitude = 200cm
```

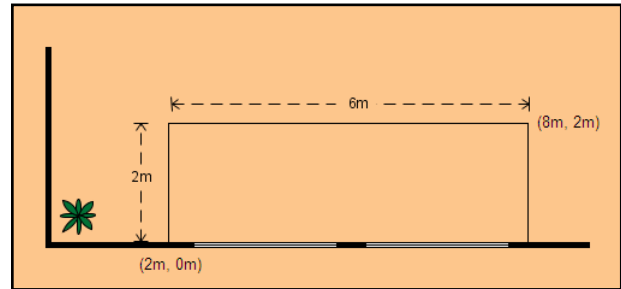


Figure 8: An absolute precondition.

```
actor
  position
    class = absolute
    center_latitude = 500cm
    center_longitude = 200cm
    radius = 150cm
```

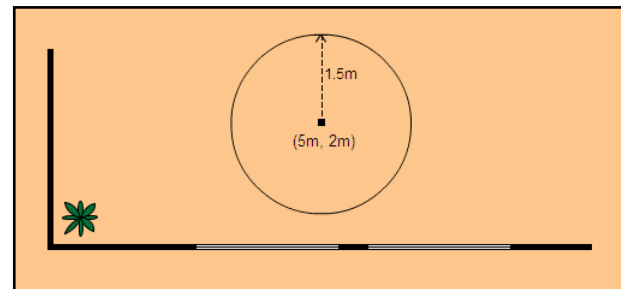


Figure 9: A circular absolute precondition.

3.3.1.3 Absence of context

Some scenarios can only be active when certain things are definitely not in the context. These constraints need to be in the precondition. We introduce two formats for this purpose. One is to put a constraint on whole entities like props and actors. The other enables us to exclude certain values from parameters. Examples of the two solutions:

```
actor
  count = 0 // no actor present

actor
  role = !father // an actor, but not a father
```

3.3.1.4 Time

Time is an important context parameter, although it can not be sensed. But there is no need to, because every computer system has an internal clock. There are many different preconditions related to time. There are preconditions on the time of the day but there can also be constraints on the day of the month or on the week of the year. To handle all these varying requirements we introduce five parameters which all can have a minimum and maximum or an exact value. These values can be added to the “stage” entity in the precondition model.

	Range	Iteration
Time of the day	00:00:00 - 23:59:59	daily
Day of the week	1 – 7	weekly
Day of the month	1 – 31	monthly
Week of the year	1 – 52	yearly
Month of the year	1 – 12	yearly

An example of a precondition on the time:

```

stage
  min_week = 20
  max_week = 40
  week_day = 5
  min_time = 13:30:00
  max_time = 17:00:00

```

3.3.1.5 Other beats

Some scenarios can only be active when others are not. Other scenarios can only be active together with certain other ones. These constraints need to be in the precondition too. For these kinds of constraints, we introduce an extra section in the precondition model, named “present”. In this section we have an “active” and an “inactive” section which can both contain beat ids or beat classes. An example:

```

present
  active
    beat_id = 02324
  inactive
    beat_id = 17454
    beat_class = display_beats

```

3.3.1.6 History

Some scenarios put constraints on the history. As described in section 3.2 we have a context history consisting of beat ids. There are five constraints we want to test the beats in the history on:

- the beat id or class
- the date and time
- the number of occurrences in the history
- the activation duration
- extra data

An example:

```

history
  beat
    id = cashRegister
    start_period = 2007-02-01
    end_period = 2007-03-01
    min_count = 5
  data
    customer = 54643

```

3.4 Software component

Our model is a detailed and formal representation of the context, but has no functionality on its own. So there has to be accompanying software to process the data from this model. As stated in the requirements section, there are three major requirements on this software part; it has to do precondition checking on the context and context history models, it has to be able to do position computations on different location information formats during these precondition checks, and it has to automatically adapt itself to the model when the model is extended with new context information parameters.

We will give an overview of the design of our component first and then handle these three requirements. The diagram in figure 10 shows the design of our component (the dotted lines), its subcomponents and its environment, the Ambient Narrative system. As we can see, our component has two databases and a software component. In the “context” database the current context will be stored and the “history” database will store the context history. The remainder of this section will describe the design of the software component.

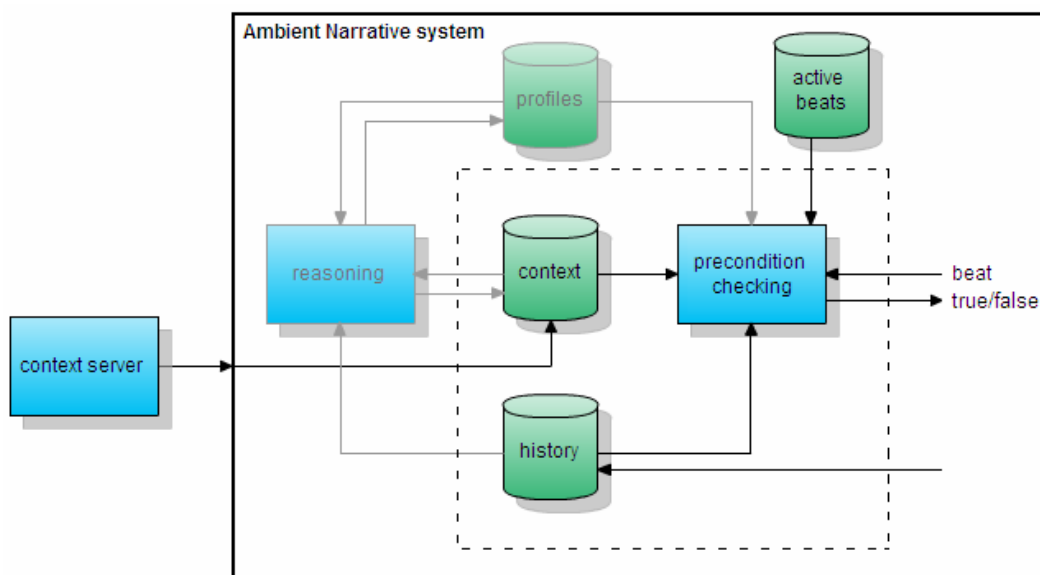


Figure 10: Diagram of our component (dotted line) within the Ambient Narrative system.

3.4.1 Precondition checking

The precondition checker will be started by the Ambient Narrative system and provided with a beat. The precondition of this beat consists of three parts; the first and most of the time largest part contains the preconditions on the context, the optional second part contains the preconditions on other currently active beats and the optional third part contains preconditions on the history, that is on beats that were active in the history.

We will give an example beat here. This beat will get active in the evening, after dinner, when there are at least two persons in the living room, when it is bad weather outside and when the TV is turned off. When all these preconditions are true, this beats can start for example colourful visualizations on the windows and a relaxing music. The precondition of this beat will look like this:

```
pre
  stage
    id = LivingRoom
    min_time = 18:00:00
    performance
      actor
        min_count = 2
  script
    present
      active
        beat_id = badWeather
      inactive
        beat_id = tvOn
        beat_id = havingDinner
    history
      active
        beat
          id = havingDinner
          min_period = -05:00:00
```

We see that in the “stage” part, three preconditions are present; it has to be the living room, it has to be evening, that is 18:00 hours or later, and there have to be at least two people. This part of the precondition can be checked on the context model. The second and third part of the precondition are the “present” and “history” parts within the “script” section. The “present” part contains three preconditions on the active beat set; the “badWeather” beat should be active, and the “tvOn” and “havingDinner” beats should not be active. The “history” part contains the precondition that it should be after dinner and states that the “havingDinner” beat should have been active in the last 5 hours.

When it receives such a beat, our precondition checker will take three steps to check the total precondition of this beat:

1. Check preconditions on other active beats (the “present” part)
2. Check preconditions on the history (the “history” part)
3. Check preconditions on the context (the main part, before the “scripts” section)

Figure 11 visualizes these three steps in a flowchart, with the names of the algorithms used for each step. These algorithms will be discussed in the next sections. We see that if one step returns false, the next steps will be skipped. Steps one and two are relatively simple, and will therefore be done first. These two steps can be performed by the same algorithm that searches for beat ids in the active beatlist or in the history beat log. Step three is a bit more heavy and verifies our preconditions on the context model.

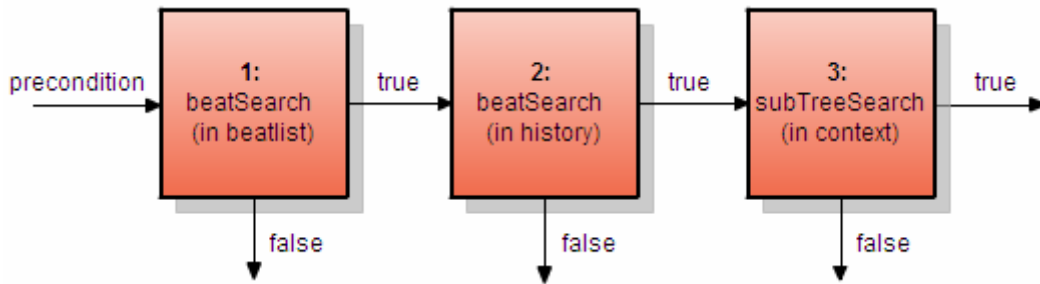


Figure 11: Flowchart of the precondition checking process.

3.4.1.1 Step 1 and 2

The next algorithm solves the problem of searching beat ids in a list of active beats or a list of beats that were active in the history. When checking preconditions on active beats, we only search for a beat id or beat class and return true or false. When checking preconditions on the history, we have to check more constraints. The precondition can put constraints on the period in which we have to search, on the duration of a beat, i.e. for how long a beat was active, on the amount of activations and on the optional data that every beat can store in the history. This brings us to the following algorithm:

```

beatSearch(precondition, beatlist) [
  if (preconditions on 'period') ► get period constraints;
  if (preconditions on 'duration') ► get duration constraints;
  if (preconditions on 'data') ► get data constraints;

  if (precondition on 'id') ►
    get all beats with this id, in the given period, of the given duration, with the given data;
  elseif (precondition on 'class') ►
    get all beats of this class, in the given period, of the given duration, with the given data;

  if (precondition on 'count') ► check count constraints on the results;
]
  
```

3.4.1.2 Step 3

The third step of the precondition checking algorithm needs a bit more attention. The context model and the beat's precondition are both hierarchically build, so basically this precondition checking process is a sub tree search in the context model. This sub tree search is not so straightforward though. It is not a simple one on one match, but it has to take many issues in account. We have seen these issues in section 3.3; the sub tree search has to map "min" and "max" values on real values, it has to check the absence of context, it has to match position information in different notations, it has to check relations and it has to check constraints on the time. To handle these requirements, we design an algorithm that recursively walks through the precondition and context model trees and checks every single precondition. When it finds a precondition on a position or on a relation, it uses separate algorithms, called "positionMatch" and "relationSearch". These algorithms will be discussed in the next two sections. This brings us to the following algorithm:

```

subTreeSearch(precondition, context) [
  for each context ►
    for each precondition ►
      if (preconditions on 'position') ►
        positionMatch(pre, context);
      elseif (preconditions on 'relation') ►
        relationSearch(pre, context);
      elseif (preconditions on 'time') ►
        check time constraints;
      else ►
        get all attribute constraints on min values, max values, and absence;
        results := get all context with these attributes;

        subTreeSearch(precondition children, results);

      if (precondition on 'count') ► check count constraints on the results;
]

```

3.4.2 Position computations

The different kinds of position notations, as seen in sections 3.1.1.2 and 3.3.1.2, demand special functionality to match position constraints from the preconditions with positions in the context. Relative positions and absolute positions exist and they can both have rectangular or circular shapes. This raises the problem of positions being defined in two different ways. We will handle all possible situations that can occur while matching preconditions with the context model.

- *Absolute precondition and absolute context:* This is trivial; the software can exactly match these two positions, i.e. compute whether the context position lies within the precondition area.
- *Absolute precondition and relative context:* An absolute precondition defines an area for a certain entity. When this entity has a relative position in the context, we recursively compute its absolute position by taking its anchor's position and its offset. Now we have two absolute positions.
- *Relative precondition and absolute context:* A relative precondition defines the distance and angle between two entities. Because we have absolute positions in the context, we can compute the distance between these two entities and match this distance with the one from the precondition.
- *Relative precondition and relative context:* Again, a relative precondition defines the distance between two entities. If these entities are also defined relatively in the context, we can match the distances only when the anchors are the same. If this is not the case, we recursively compute the absolute positions of the precondition or the context by taking the anchors' positions and their relative offsets, until we have at least one absolute position. Then we can recursively use this same algorithm to match these positions.

Positions with variables are handled by replacing the variable with the real ids and then calling the position search algorithm for every instance. This brings us to the following algorithm:

```

positionMatch(precondition, context) [
  if (precondition.position == context.position) ►
    true;
  elseif (precondition.position.class is 'absolute') ►
    x := find absolute position of context;
    positionMatch(precondition, x);
  elseif (precondition.position.class is 'relative') and (context.position.class is 'absolute') ►
    y := compute distance(precondition.position.anchor, context);
    check (precondition.position.distance < y);
  elseif (precondition.position.class is 'relative') and (context.position.class is 'relative') ►
    x := find absolute position of context;
    y := find absolute position of precondition.position.anchor;
    positionMatch(precondition, x) or positionMatch(precondition + y, context);
]

```

We will give an example of a precondition and a context fragment. The precondition contains a relative position. The context contains an entity that matches with this precondition but has a relative position to another anchor. Figure 12 visualizes this situation. The circle around the TV matches with the precondition.

A precondition fragment:

```

actor
  min_age = 45
  position
    class = relative
    anchor = TV
    max_distance = 200cm

```

A context model fragment:

```

prop
  id = door_1
  position
    class = absolute
    latitude = 200cm
    longitude = 0cm
prop
  id = TV
  position
    class = relative
    anchor = door_1
    angle = 90°
    distance = 400cm
actor
  age = 51
  position
    class = relative
    anchor = door_1
    angle = 60°
    distance = 350cm

```

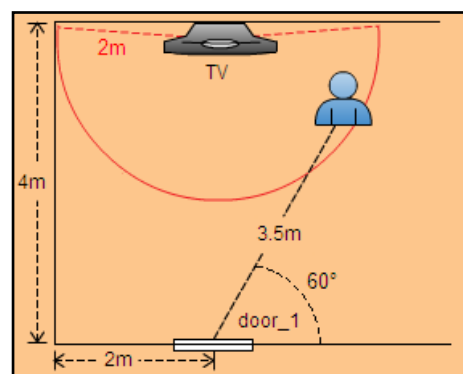


Figure 12: Context and precondition.

3.4.3 Relations and variables

An important aspect of the software component is the ability to understand and handle variables in the precondition. These variables are necessary to link entities together. When there is an actor and an object in the precondition for example, and there has to be a relation between these two, we want to point to these specific entities from the relation entity. An example fragment of a beat's precondition with two variables:

```
performance
  prop
    id = $book
  actor
    id = $actor
    role = customer
    min_age = 45
relation
  class = point
  object = $actor
  subject = $book
```

To handle these variables we design some functionality that stores all variables with their corresponding context ids during the sub tree search. This way we can look up the real ids later, when checking the relation preconditions. To check these relation preconditions, a separate algorithm is used. This relation search algorithm handles variables by replacing them for real ids before checking the relation preconditions on the context.

```
relationSearch(precondition, context) [
  if (precondition on relation 'class') ► get relation.class constraints;
  if (precondition on relation 'object') ► get relation.object constraints;
  if (precondition on relation 'subject') ► get relation.subject constraints;

  get all relations of the given class, with the given object, with the given subject;

  if (precondition on 'count') ► check count constraints on the results;
]
```

3.4.4 Extensibility

To make the software accept new extensions to the precondition model and context model, we make it generic in a way that it accepts all entities and attributes in the precondition and always tries to match these with the context model. There is no real restriction on naming, but new entities or attributes will only be checked on exact value, minimum value or maximum value. The software will not do computations on these values. Furthermore it is not possible to add new attributes to the position entity. So introducing new entities or attributes in the context model is almost always possible, as long as we make sure to use the same names and types in the preconditions and in the context model.

4 Implementation

4.1 Modelling language

In the previous section, we presented the design of our context model. A UML diagram of this design can be found in figure 4 in section 3.1.3. In this section we will look at the implementation of the model. To this end we have to choose a modelling language. As we have seen in section 1.4 on previous work, existing solutions either use XML or RDF as a context modelling language.

Our precondition checking consists of two main tasks. These tasks are sub tree searches and computations. From the requirements phase we know that position information is the most important context information. It covers the 1st, 4th, 5th, 6th and 9th position in the list of most used context parameters. To be more precise, position information appears in 86% of all ambient intelligence scenarios we have seen. While checking beats' preconditions for these scenarios, computations have to be done to match different positions. So we want to be able to do fast sub tree searches and fast computations on our model.

XML

The eXtensible Markup Language is a general-purpose markup language for documents containing structured information. As the name implies, the language is extensible, which means that developers can define their own tags. Tags describe the information contained in these tags and form an XML tree. Reasons to use XML are its compact notation, simple validation, available tooling and easy integration with development platforms.

RDF

The Resource Description Framework (RDF) is a framework for describing information on the web. RDF is a language based on XML and describes information in triplets. These triplets form a RDF graph. Because RDF uses an XML syntax, it inherits all properties of the XML language. Reasons to use RDF are easy automated understanding, integration with the semantic web and reasoning possibilities.

Because of the tree property of XML in comparison with the graph property of RDF, and because of the compact notation, it is faster to do sub tree searches in XML. Because of our focus on fast precondition checking and position computations, and not on reasoning, we choose XML as a modelling language.

4.2 Context model

To define the structure, content and semantics of our XML model, we create an XML schema. This schema is created from the UML diagram of our conceptual model, presented in the previous section. The schema defines entities and attributes, their types and relations, and puts restrictions on their minimum and maximum occurrences. The XML schema of the model can be found in appendix D.5. Below we give an example of a context description by our model. This context description example is visualized by figure 13.

```

<context>
  <stage id="insideShop">
    <condition>
      <light>1000</light>
      <temperature>25</temperature>
    </condition>

    <performance>
      <activity>presence</activity>
      <prop id="book_x">
        <name>Lord of the Rings</name>
        <colour>green</colour>
        <position class="absolute" system="xyz">
          <latitude>400</latitude>
          <longitude>40</longitude>
        </position>
      </prop>
      <prop id="mirror_b">
        <name>Interactive Mirror</name>
        <position class="absolute" system="xyz">
          <latitude>10</latitude>
          <longitude>300</longitude>
        </position>
      </prop>
    </performance>

    <performance>
      <activity>walking</activity>
      <speed>2</speed>
      <angle>-135</angle>
      <actor id="actor_1">
        <role>customer</role>
        <age>40</age>
        <position class="relative">
          <anchor>mirror_b</anchor>
          <angle>0</angle>
          <distance>240</distance>
        </position>
      </actor>
    </performance>

    <relation class="gaze">
      <object>actor_1</object>
      <subject>book_x</subject>
    </relation>
  </stage>
</context>

```

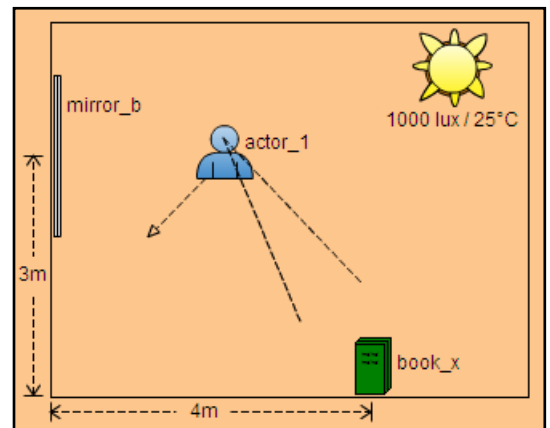


Figure 13: Situation described by the model.

4.3 Context history

As described in the design section, we will store beat ids with corresponding activation and deactivation dates and times in the history model, together with optional extra information in the “data” field. This model will be implemented by a simple XML model. An example:

```
<beatlog>
  <beat id="buyTransaction" class="customerMode">
    <activated>2007-03-24 14:15:30</activated>
    <deactivated>2007-03-24 14:15:36</deactivated>
    <data>
      <customer>cust_00123</customer>
      <product>xyz</product>
      <product>abc</product>
      <total_price>125,99</total_price>
    </data>
  </beat>
</beatlog>
```

This implementation offers us two possibilities to use context history. As described in the design section we can put preconditions on beat ids with the possibility to define certain periods, minimum or maximum occurrences and tests on the extra data field. This already is a powerful way to create complicated scenarios.

However, as we have already mentioned in section 3.2, our history model offers a second possibility to use context history. We could also use the history as if it were real context information. The great plus of this method would be that it lets us create preconditions in the same way as we do on the current context. The idea behind this method is that it will not check preconditions on the history model itself, but first recreate the history of a certain time or period by combining all preconditions of beats that were active in that period. This way we can recreate the real context of the history. And then we could check preconditions in exactly the same way as we do with normal context.

Because of time constraints we only implement the first method, i.e. the possibility to put preconditions on beat ids, in the software. The other method is a very interesting one though and should definitely be implemented and tested in the future. The context history model supports both methods, so the model will not have to be changed. It is the software doing the precondition checking that has to be extended. This software will have to lookup the beat ids in a certain period in the history model, then has to find these beats' preconditions, combine them and then check the preconditions on this derived history.

4.4 Software

As stated in the requirements section, we will use Java as a programming language. We create one class that does the total preconditions checking. The main method of this class is called with three parameters, all pointers to an XML tree. The first one is the context, the second is a beat and the third is a list of active beats merged with the history beatlist.

In the design phase we found that there are three steps to take; first check preconditions on other active beats; then check preconditions on the history; and finally check preconditions on the current context. The first two steps are implemented by “beatSearch” method. The third step is implemented by the method “subTreeSearch”. Figure 14 shows these methods and the other methods of our class in a call hierarchy. To be more specific, we can map the four algorithms we designed in the design section on one or more of these methods. The algorithm “beatSearch” is implemented by the methods “beatSearch”, “singleBeatSearch”, “checkDuration”, “dateDifference” and “beatDataSearch”. The algorithm “subTreeSearch” is implemented by the methods “subTreeSearch” and “timeCheck”. The algorithm “relationSearch” is implemented by the method “relationSearch”. The algorithm “positionMatch” is implemented by the methods “positionSearch”, “positionVariableSearch”, “findAbsolutePosition” and “findAnchorPosition”.

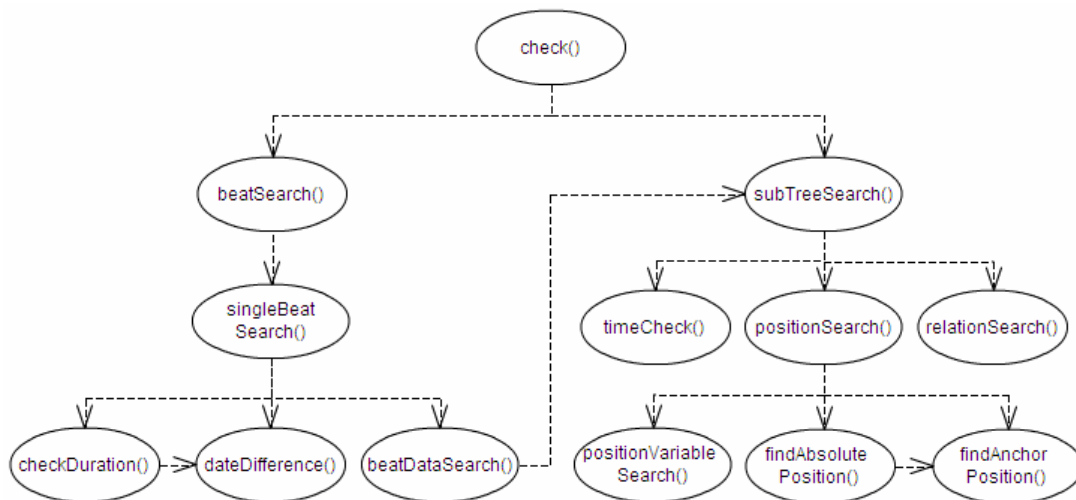


Figure 14: Call hierarchy of our component.

We will very shortly handle all these methods to explain their purpose.

check(context, beat, beatlist) checks the complete precondition of a beat in three steps: 1) check preconditions on the active beatlist by calling *beatSearch()*; 2) check preconditions on the history beatlist by calling *beatSearch()*; 3) check preconditions on the current context by calling *subTreeSearch()*. Returns true or false.

beatSearch(pre, beatlist, in_history) checks the preconditions on the active beatlist or on the history beatlist, by a loop over the preconditions on active and inactive beats and calling *singleBeatSearch()* for all of them. Returns true or false.

5 Evaluation

5.1 User test

Because expressiveness is one of the most important requirements on our model, we have to evaluate this expressiveness. To test our context model on expressiveness we set up a user test. We will explain the test first and then discuss the results.

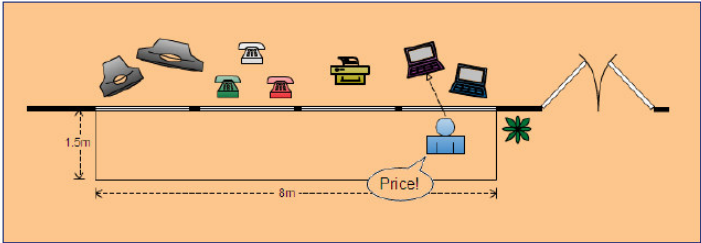
5.1.1 Experiment

During this test we ask a group of users to come up with a random set of new scenarios. Afterwards, we will try to express these scenarios in our model. This second step will be performed by us, because end-users will never get to see the XML model itself.

Nine people (3 female and 6 male researchers and students) participated in the test. These nine participants were divided in four groups and every group was asked to think like an end-user, that is the author of the system, i.e. the person who writes the precondition and action parts of the beats. These groups were asked to come up with some scenarios they would want to create, as they were the end-user of the system. We did not show our context model, but to show some possibilities we first provided a couple of examples of possible scenarios. See figure 16 for such an example. There were no restrictions on the location (inside, outside, in a car, etcetera) or domain (home, retail, work, etcetera). We asked the groups to write down some scenarios in either a short text or in a drawing.

PHILIPS

Scenario example 1



“A person is close to the shop window, looks at a certain product, and says ‘Price!’.”

Action: The product’s price is shown on the window.

Research Media Interaction 9

Figure 16: An example scenario shown to the user test participants.

The four groups produced in total a list of 13 scenarios, which are listed here:

- I. "After dinner, a smart home senses that there are many people (guests) in the living room. It thinks it is a party, so it plays nice music and gives visualizations on the windows of the living room."
- II. "A shop window senses that a family or a group of friends or people with pets stand in front of the shop window and shows targeted advertising."
- III. "During bad weather, the window of the living room shows nice weather."
- IV. "A shop window senses people talking, close to the shop window after hours and triggers an alarm."
- V. "A shop window senses an animal in front of the shop window and triggers an ultrasonic sound."
- VI. "When the shop window senses a female customer, older than 65, in front of the shop window pointing to a product, it starts the 65+ interaction mode."
- VII. "A customer in a flower shop shows emotion while looking at and smelling a specific flower. The system detects this situation and emotion and responds to it."
- VIII. "The system detects specific needs, for instance bored children in a shop and responds to it."
- IX. "The system detects the taste of ice cream and knows your preferences. It gives a warning for those that are too sweet for you."
- X. "A shop window senses a person walking in the cold and the rain, not close to the shop window, with a dog and an umbrella and looking straight ahead. It responds to this in a very specific way to attract the person's attention."
- XI. "A husband, wife and child are shopping. The woman is inside the shop, and the father and child wait outside. This is sensed by the system and, when there is nobody else in front of the shop window, it responds to this situation by offering a game at the eye level of the child."
- XII. "In the evening, a customer is reading information on the shop window. Then the system senses a pickpocket behind the customer and gives a warning to watch out."
- XIII. "A couple is shopping together. A shop window senses this and shows one information zone with a product they are both talking about, instead of two separate information zones."

5.1.2 Results

When we analyse the context information aspects used by these user test scenarios, we come to a list of 31 context parameters. This list is shown in table 6.

Persons	1	2	3	4	5	6	7	8	9	10	11	12	13
Identity										x	x		
Age						x		x			x		
Gender						x					x		
Length											x		
Location													
<i>absolute position</i>								x				x	x
<i>relative position</i>		x	x	x	x	x			x	x	x	x	x
<i>which room</i>	x		x										
Orientation										x			
Activity				x						x	x	x	
Speech				x									x
Emotions / Mood	x						x	x					
Intentions													x
History													
<i>Activities</i>	x												
Profiles / Rights		x			x	x			x				
Relations													
Entities nearby		x								x	x	x	
Touch										x			
Gestures						x							
Gaze							x						
Devices	1	2	3	4	5	6	7	8	9	10	11	12	13
Identity	x									x	x		
Location													
<i>absolute position</i>									x				
<i>relative position</i>										x			
Relations													
Location													
<i>same room</i>	x												
Entities nearby		x		x	x	x			x		x	x	x
Groups	1	2	3	4	5	6	7	8	9	10	11	12	13
Number of individuals	x												
Relations													
Is family		x											
Is friends		x											
Environment	1	2	3	4	5	6	7	8	9	10	11	12	13
Indoor / Outdoor						x							
Light			x									x	
Temperature			x							x			
Time of day	x			x									x
Water			x							x			

Table 6: Context usage of the user test scenarios.

To get a quick idea of the context usage of the user test scenarios, we show the top 10 of the context parameters, when ordered by their usage frequency, in table 7. We can compare this list to the context parameter list from our model's requirements by the third column, in which the position of each context parameter in the initial requirements is displayed.

To complete the user test, we modelled all test scenarios as a precondition of a beat. As an example, two beats of the user test scenarios can be found in appendix D.6.

Position in user test	Context parameter	Position in requirements
1	Relative position of a person	1
2	Entities nearby a device	6
3	Activity of a person	11
4	Entities nearby a person	18
5	Identity of a person	3
6	Identity of a device	2
7	Absolute position of a person	4
8	Emotions of a person	24
9	Time of the day	21
10	Age of a person	45

Table 7: User test top 10 of context information parameters

When we compare the whole set of 31 context parameters of the user test to our model's requirements, we find that 29 of the 31 context parameters are covered by our model. One of the user test scenarios contains two context parameters that are not supported by the model. These two context parameters are actually of the same type, they both describe a relation between a group of people: our model is unable to describe a "family" and a "friends" relation between a group of entities.

This means that 94% (29/31) of the context parameters from the user test are supported by the model. Based on the fact that this was a broad and random user test, we state that our model is able to describe a very large part of the situations in the home and retail domain. And with the new precondition format, beats can be tested for these situations on the model.

5.2 Software test

5.2.1 Test scenario

To test the software working with the context model, the existing DreamScreen project and Ambient Narrative system have a test scenario. This test scenario consists of a set of ten beats. It is a simple but effective scenario, using only one sensor, which is a "smart floor" that does position detection. There are five different areas, defined by the preconditions of the beats. People walking in and out these areas trigger one or more of these beats:

Beat overview

1 AttractorMode	Active when there is nobody within 2 meters of the shop window.
1 AttentionMode	Active when there is somebody walking by the shop window.
4 InteractionModes	Active when there is somebody standing still in front of one of the four shop windows.
4 BlankModes	Active on the other windows when the InteractionMode is active on one of the shop windows.

This scenario does not use the features of the new context model. So we have to create a new scenario with more beats, triggered by different kinds of context information, to show the possibilities of the system using the new context model. To this end we take the list of parameters from the requirements phase and come to the following list of context information that we want to use in our test scenario:

Context parameters

The position of a person or a device
The identity of a person or a device
Entities nearby a person or a device
Speech of a person
Touch of a person
The orientation of a person or a device
The activity of a person
Capabilities of a device
Gestures of a person
Gaze of a person

Table 8: Context requirements for the test scenario

Basically we create the same scenario as the existing test scenario, but now using the new context model with additional features. First we translate the current beat set into the new modelling syntax, and then we add new features using other context information. We use daylight and time information to select an appropriate AttractorMode. We adapt the AttentionMode to the time and have different ones for a single person or for a group of people walking by. We change the InteractionModes depending on the capabilities of devices of the people in front of the window. Also different InteractionModes will be shown when people look or point at certain products in the shop window. We add an extra AttentionMode that responds to people skating by, instead of walking by. We add an extra InteractionMode for people pointing at a product and simultaneously saying the word "price". Finally we create a beat that gets active when a person is standing against the window with his/her back.

Besides this, we create two versions of this new beat set; one using absolute positions and the other using only relative positions, to show all the possibilities of the context model. So we duplicate the entire beat set. In the first set we assume that there is an absolute position of everything. In the second set we only use relative position information to show that the same scenario can be realized.

In the beat overview below we list all beats, their purpose and, between parentheses, the context information used. We see that the list of table 8 is totally covered by one or more beats. Two example beats from the test set can be found in appendix D.7. With this beat set, we will test our software component.

Beat overview

4 AttractorModes	Depending on day/night (light) & summer/winter (time).
5 AttentionModes	Depending on day/night (light) & group/alone (amount of actors) & skaters (person's activity).

8 InteractionModes	Four windows, depending on with/without Bluetooth (capability of devices nearby a person).
4 InteractionModesGaze	Active when looking at a product (position, person's gaze).
4 InteractionModesPoint	Active when pointing at a product (position, person's gestures).
1 InteractionModePointSpeak	Active when pointing at a product with voice recognition (position, person's gestures, person's speech).
1 BackOffMode	Active when standing with the back against the window (person's position, orientation and touch).
1 KnownCustomerMode	Active when a known customer returns to the shop (person's identity).

5.2.2 Test tool

The new scenarios introduced in the previous section have to be tested in practice on different context situations. To this end, we created a test program that loads a beat set and a set of context files, i.e. snapshots of the context at specific moments in time. The program will check this beat set on every context, using our precondition checking software component. We will show this test tool and its functionality here, using the test beat set.

First we have to open a folder containing context files and a folder containing beat files, and then the tool can start the precondition checking. So first we load a beat set and a context set. This is shown in figure 17. We can inspect the individual context and beat files by opening the tree views. This can be seen in figure 18, where we see part of a context on the left and complete beat on the right.

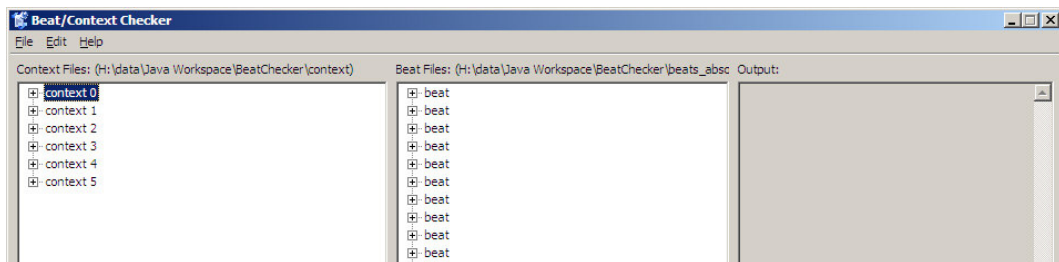


Figure 17: Context files and beat files are loaded.

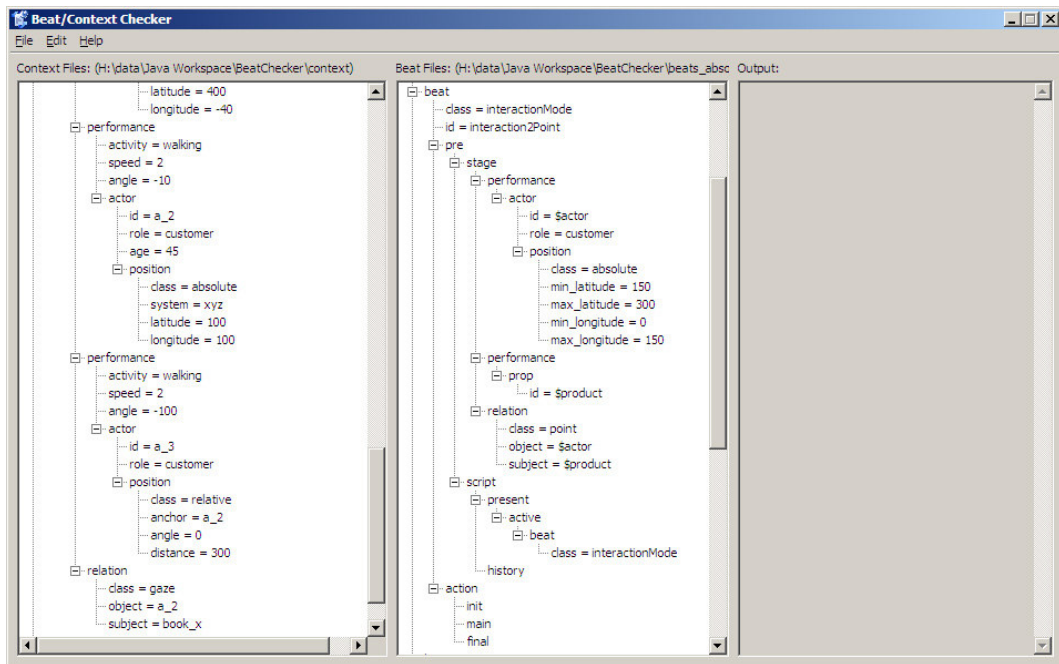


Figure 18: Parts of a context file and a complete beat file.

The test tool also supports context history. The history log can be inspected and, if necessary, changed. When opening the history, we see a beat log as shown in figure 19. This history can be edited for testing activities. Here it can be seen that there are two beats in the history.

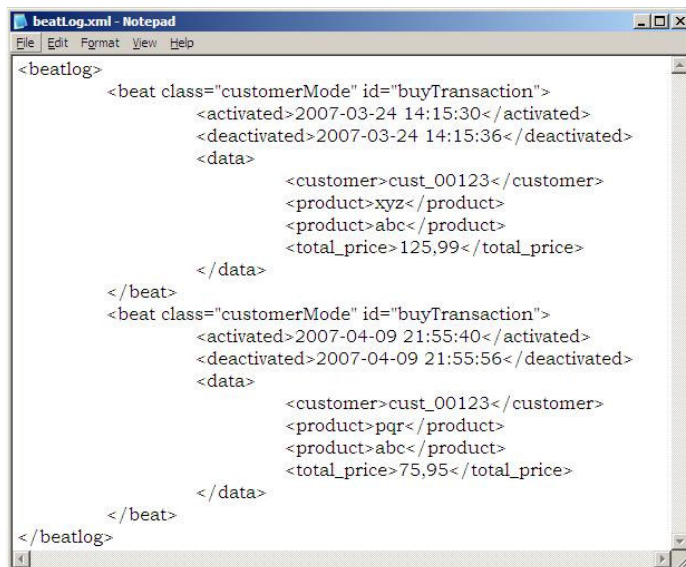


Figure 19: The beat log file, containing beats that were active in the history.

After loading context and beat files, we can start the actual precondition checking. A new window will appear that shows the context file that is currently used and the beats that are active in this context. This can be seen in figure 20. The window shows a picture and the name of every active beat. The picture describes the situation in the precondition of the corresponding beat. These pictures are hand made and added to a beat by the beat's author.

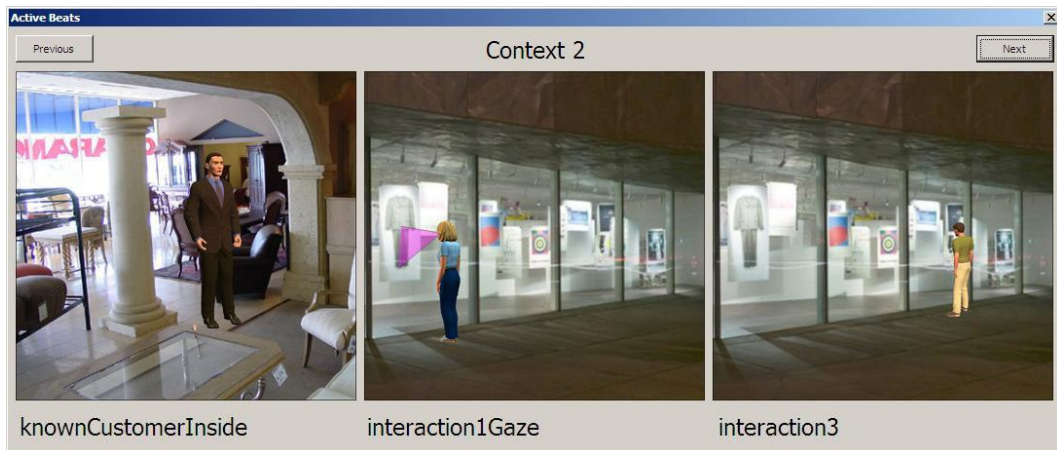


Figure 20: Three beats are active in context 2.

With the “next” and “previous” buttons we can go to another context, so that all beats’ preconditions will be checked again on a new context. This way we can walk through all contexts and test our beat set on different situations.

5.2.3 Results

While creating the context files for this test, we listed per context situation which beats should be activated. During the test we can verify the results by comparing the activated beats with the ones from these lists. The software should select the right set of beats at every context and no other beats than that.

The final results of our software tests show us that the precondition checking component does exactly what it is expected to do. In a broad range of context situations and beats, it selects exactly the right beats for every single context situation.

The precondition checking tool works correct and precise on position computations of persons and devices. Both absolute and relative positions are used and this raises no problems. All other aspects of context information like, for example, speech, gestures, gaze, touch or activities of a person or the orientation or capabilities of a device are detected and processed in a correct way.

6 Conclusions

In this section we will answer our research questions, draw conclusions and evaluate the project. Finally we will propose some future work topics.

6.1 Context

To answer our first research question, “What kind of context information is needed in ambient intelligence surroundings?”, we performed several research tasks. The main objective of these research tasks was to identify existing context usage. To this end we first of all performed a thorough literature study, in which we focused on proposed context usage and the use of context information in existing context aware systems. As a second step we researched a number of future scenarios and indexed their use of context information. To complete our search for context information we organized a brainstorm session to come up with new future scenarios using context. During these research activities we focused on two domains for ambient intelligence, which are the home and retail domains.

After having collected all kinds of context information parameters, we ordered, grouped and analysed these results. This way we came up with two partially overlapping lists of context information parameters, which can be found in tables 1 and 2 in section 2.1.5. After merging these two lists, we ended up with 80 unique context information parameters. The most important parameters of this set, depending on the amount of their occurrences, include the position, orientation and movements of a person or device, the identity and activity of a person and the contents and capabilities of a device. We used this total set of context parameters as input for our next research steps. This list can also be a useful list for other future research projects.

6.2 Context model

Knowing which context information is needed, we are able to answer our second research question: “How can we efficiently represent this context information in a model?”. To answer this question, we made a design of a compact model that represents the context of our system. To fulfil this task, the model is able to describe every context information parameter we found earlier.

We created a hierarchical model using names from the theatre world. Every context has one or more “stages” which are areas like rooms, floors or gardens. A “stage” can host zero or more “performances”, which are activities, and “relations”. A “performance” has one or more “actors”, people, or “props” which are objects or devices. The “relations” define relations between “actors” or “props”. All these entities in our model have different attributes. An “actor”, for example, has attributes like “age” or “gender” and a

“performance” has an “activity” and “start time”. This resulted in a compact model of 12 entities with 54 attributes. A diagram of this model can be found in figure 4.

During our research on context information, we found that almost 10% of the context information was not about the current context, but about context history. Therefore we designed a second model next to the context model, to represent context history information. There was not so much information about context history in the literature, so we came up with a totally new method to store context history in a compact but still useful way.

To test the expressiveness of our context model, we conducted a user test. We asked a group of people to think of random situations and we tried to express these situations in the model. The model was able to describe 94% of all aspects of these user test situations. This result demonstrates that our model satisfies our design goal of being richly expressive and broadly applicable; the model covers a broad range of ambient intelligence scenarios and can be used to describe these scenarios in small detail.

6.3 Functionality

After we designed the model to represent the context, we can answer the third research question, “How can this model be used in the existing Ambient Narrative system?”, and integrate our model in the Ambient Narrative system. The beats of the Ambient Narrative system put preconditions on the context, i.e. on our model. To make this work, these preconditions have to be compatible with the model. Therefore we first designed a precondition format based on our model. After that we started the implementation.

To this end we decided to use XML as a modelling language to implement our model and Java as a programming language to build the surrounding software component. This component performs the precondition checking, which includes comparing values to exact values, upper and lower bounds, counting entities, working with relations and variables and performing computations on position data. As an implementation of the model, we translated the conceptual model into an XML schema. This schema can be found in appendix D.5. The implementation of the software component resulted in a Java class that can easily be used by other programs to check a certain precondition on the context model.

The most important things checked by this software component on the context model are geographical positions, which turned out to be the most important context information aspects. The software component performs computations on position information to compare different position formats and compute distances. We can, for example, check preconditions like “is there a person, in an area of 6 by 4 meters in the front left corner of a room, who is within 1 meter of a phone, which is at least 3 meters away from any wall”. Next to this position information, other preconditions can be checked like, for example, the age or emotions of the person, the size or capabilities of the phone or the temperature in the room. Next to that, we can add history constraints to this precondition like “the person should not have been in this room before” or “the phone should have ringed at least twice, in the last hour”.

We tested our software component with a broad set of beats. To this end we created a test tool that loads a set of beats and a set of context files and then uses our software component to check all beats' preconditions on every context. We created a set of 64 beats and a set of 6 instances of our model representing different context situations. This test was successful; the software computes exactly the right set of beats for every context situation. Also the use of history was tested and found to be a powerful tool to create more advanced beats. We tested a beat that detects customers in a shop that bought certain products in this shop before. This procedure works very accurate and it shows with a concrete example how history can help to increase the possibilities of the system.

6.4 Future work

Context history

We have done some work on context history and found that the main problem is the enormous amount of data. We provided a mechanism to put preconditions on events in the history, without the need to always store all context information. A disadvantage of this method is that we lose the real context, so we proposed another method, in which we can really recreate the context of a certain moment in the history. More research has to be done on this method to be able to test and implement it.

User profiles

To make the system more useful, there has to be a user profile component on which those preconditions can be checked that can not be verified in another way. For example, we can sense the identity, weight or length of a person, but we can not sense a person's preferences or rights.

Context future

While working with historical and present context, the idea of future context comes up. We think about context prediction mechanisms to enable the system to adapt itself even more to the user's needs. Research on context precondition is ongoing and it would be very interesting to integrate this kind of functionality in our system.

Context reasoning

We mentioned context reasoning several times throughout this report. We did not focus on these mechanisms, but reasoning is a very powerful tool, for example, to extract patterns and find preferences of certain entities in the context. So reasoning can help to build user profiles and predict future context.

Sensor failure and uncertainty

A different aspect of our system is the dependence on lots of sensors. This raises the problem of sensor failure – what if a sensor does not provide output and certain beats have preconditions on its values – and the problem of uncertainty – what if a sensor is not very reliable and provides wrong values now and then. Research has to be done on these areas to make the system more robust.

C References

C.1 References

- [Aarts 03] Aarts, E.H.L., Marzano, S. "*The new everyday; views on ambient intelligence*", ISBN 9064505020, 010 Publishers, Amsterdam, 2003
- [Aarts 06] Aarts, E.H.L., Diederiks, E. "*Ambient Lifestyle, from concept to experience*", p. 198-202, ISBN 9789063691615, BIS Publishers, Amsterdam, 2006
- [Abowd 00] G. D. Abowd, E. D. Mynatt, "*Charting Past, Present and Future Research in Ubiquitous Computing*", ACM Transactions on Computer-Human Interaction, 7(1):29-58, March 2000.
- [Becker 05] Becker, C., Dürr, F. "*On location models for ubiquitous computing*", in Journal of Personal and Ubiquitous Computing 9, 20-31, Springer, 2005
- [Bielefeld] www.techfak.uni-bielefeld.de/ags/ai/projects/mimic/welcome.html
(November 2006)
- [Brusilovsky 96] P. Brusilovsky. "*Methods and Techniques of Adaptive Hypermedia*", User Modeling and User-adapted Interaction, 6(2-3):87-129, 1996
- [Byun 04] H. Byun, K. Cheverst. "*Utilizing Context History To Provide Dynamic Adaptations*", Applied Artificial Intelligence 18(6): 533-548, 2004
- [Coen 98] M. H. Coen. "*Design Principals for Intelligent Environments*", Proc. Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI'98), Madison, Wisconsin, 547-554, Mostow, C.R.A.J. (ed) AAAI Press / MIT Press, 1998
- [Decker 03] C. Decker, U. Kubach, M. Beigl. "*Revealing the retail black box by interaction sensing*", in Proceedings of the ICDCS 2003, Providence, Rhode Island, 2003
- [Dey 01] Dey, A. "*Understanding and Using Context*", in Personal Ubiquitous Computing 5, January 2001, 4-7
- [Dijk 01] Dijk, E.O., "*Position and tracking technology for Context Awareness applications in the Home*", Technical Note 2001/439, Phenom project, Philips Research, Eindhoven, November 2001
- [Dogac 03] Dogac, A., Laleci, G., Kabak, Y. "*Context Frameworks for Ambient Intelligence*". eChallenges, October 2003, Bologna, Italy

- [Doorn 05] M. van Doorn, A.P. de Vries, "*Ambient Narratives and Storytelling in Ambient Intelligence*", 2005
- [Doorn 06] M. van Doorn, A.P. de Vries, "*Co-creation in Ambient Narratives*", in *Ambient Intelligence for Everyday Life (Aml-Life'05)*, Lecture Notes in Computer Science 3964, 2006
- [Doorn 07] M. van Doorn. "*Specification End-user Programming Architecture*", Philips Research, draft 0.5, Eindhoven, CRE 2007
- [Ferscha 02] A. Ferscha, S. Vogl, W. Beer, "*Ubiquitous Context Sensing in Wireless Environments*", presented at Workshop on Distributed and Parallel Systems, Austria, 2002
- [Garate 05] Gárate, A., Herrasti, N., López, A. "*GENIO: an ambient intelligence application in home automation and entertainment environment*", in *Proceedings of the 2005 Joint Conference on Smart Objects and Ambient intelligence: innovative Context-Aware Services: Usages and Technologies, sOc-EUSAI '05*, vol. 121. ACM Press, New York, 2005, 241-245
- [Gronbaek 03] Gronbaek, K, J.F. Kristensen, P. Orbaek, M.A. Eriksen. "*Physical Hypermedia: Organising Collections of Mixed Physical and Digital Material*", *Proc. ACM Conference on Hypertext*, Nottingham, 2003, pp. 10-20
- [Gunn 04] James Gunn. "*Slither*", September 10, 2004, www.imsdb.com/scripts/Slither.html (October 2006)
- [Halasz 94] F. Halasz, M. Schwartz. "*The Dexter Hypertext Reference Model*", *Communications of the ACM*, 37(2):30-39, February 1994
- [Hansen 04] F. A. Hansen, N. O. Bouvin, B. G. Christensen, K. Grønbaek, T. B. Pedersen, J. Gagach. "*Integrating the Web and the World: Contextual trails on the move*", in D. de Roure and H. Ashman, editors, *Proceedings of the 15th ACM Hypertext Conference*, Santa Cruz, CA, USA, August 2004, ACM Press
- [Hatley 00] Hatley, DJ, Hruschka, P, Pirbhai, I. "*Process for system architecture and requirements engineering*", Dorset House, 2000
- [Hull 97] R. Hull, P. Neaves, J. Bedford-Roberts. "*Towards Situated Computing*", *ISWC '97: Proceedings of the 1st IEEE International Symposium on Wearable Computers*, IEEE Computer Society, 1997
- [Hung 05] Hung Quoc Ngo, Anjum Shehzad, Kim Anh Pham Ngoc, S. Y. Lee, Manwoo Jeon, "*Research Issues in the Development of Context-Aware Middleware Architectures*", 11th IEEE International Conference on

Embedded and Real-Time Computing Systems and Applications (RTCSA'05), pp. 459-462, 2005

- [Kleinhou 03] Kleinhou, J.C., Hollemans, G., Lashina, T.A., Korst, J.H.M. "*Context information: representation and reasoning*", Technical Note 2003/00525, Easy Access project, Philips Research, Eindhoven, July 2003
- [Korkea 00] M. Korkea-aho, "*Context-Aware Applications Survey*", Internetworking Seminar (Tik-110.551), Department of Computer Science, University of Helsinki, April 2000, users.tkk.fi/~mkorkeaa/doc/context-aware.html
- [Kostianen 05] M. Kostianen, M. Kiesila, "*The intelligent retail store*", Department of Computer Science, University of Helsinki, October 2005
- [Lucas 88] George Lucas, Bob Dolman. "Willow", 1988, www.imsdb.com/scripts/Willow.html (October 2006)
- [Meyer 03] S. Meyer, A. Rakotonirainy, "*A Survey of research on context-aware homes*", Australasian Information Security Workshop Conference on ACSW Frontiers, 2003
- [Nijholt 04] A. Nijholt, "*Smart Exposition Rooms: The Ambient Intelligence View*", in Proc. Electronic Imaging & the Visual Arts (EVA 2004), V. Cappellini & J. Hemsley, Pitagora Editrice Bologna, March 2004, 100-105
- [Philips 07] "*What is Ambient Intelligence?*", Technologies, Philips Research, www.research.philips.com/technologies/syst_softw/ami (June 2007)
- [Pine 99] B.J. Pine II, J.H. Gilmore. "*The Experience Economy: Work Is Theater & Every Business a Stage*", ISBN 0875848192, Harvard Business School Press, 1999
- [Przybilski 05] M. Przybilski, P. Nurmi, and P. Floréen. "*A Framework for Context Reasoning Systems*", in Proceeding Software Engineering 2005, 455-808, Innsbruck, Austria, February 2005
- [Rocker 05] Röcker, C., Janse, M. D., Portolan, N., Streitz, N. "*User requirements for intelligent home environments: a scenario-driven approach and empirical cross-cultural study*". In Proceedings of the 2005 Joint Conference on Smart Objects and Ambient intelligence: innovative Context-Aware Services: Usages and Technologies, sOc-EUSAI '05, vol. 121. ACM Press, New York, 2005, 111-116
- [Romero 03] Romero, L, Nuno C. "*HyperReal: a Hypermedia Model for Mixed Reality*", in Proceedings of the Fourteenth ACM Conference on Hypertext and Hypermedia, pp. 2 – 9. New York, NY: ACM Press, 2003

- [Roussos 02] G. Roussos, L. Koukara, P. Kourouthanasis, J.O. Tuominen, O. Seppala, G. Giaglis, J. Frissaer. "A case study in pervasive retail", Proc. ACM Mobile Commerce 2002, pp. 100-105
- [Salber 98] Salber, D., Abowd, G. "The Design and Use of a Generic Context Server", Technical Report GIT-GVU-98-32, Georgia Institute of Technology, 1998
- [Salber 99] D. Salber, A. K. Dey, R. Orr, G. D. Abowd, "Designing for Ubiquitous Computing: A Case Study in Context Sensing". GVU, Technical Report GIT-GVU-99-29, July 1999
- [Sato 05] Sato, I. "A location model for ambient intelligence", in Proceedings of the 2005 Joint Conference on Smart Objects and Ambient intelligence: innovative Context-Aware Services: Usages and Technologies, sOc-EUSAI '05, vol. 121. ACM Press, New York, 2005, 195-200
- [Schmidt 99] Schmidt, A. et al. "There is more to context than location", in Computer & Graphics, 23(6), 1999, pp 893 -901
- [Schmidt 00] Schmidt, A. "Implicit Human Computer Interaction Through Context". Personal Technologies Volume 4(2&3), 2000, 191-199
- [Schmidt 05] Schmidt, A., Kranz, M., Holleis, P. "Interacting with the ubiquitous computer: towards embedding interaction". In Proceedings of the 2005 Joint Conference on Smart Objects and Ambient intelligence: innovative Context-Aware Services: Usages and Technologies, sOc-EUSAI '05, vol. 121. ACM Press, New York, 2005, 147-152
- [Shehzad 04] A. Shehzad, H. Q. Ngo, K. Anh Pham, S. Y. Lee. "Formal Modeling in Context Aware Systems", in Proceedings of The 1st International Workshop on Modeling and Retrieval of Context (MRC'2004), Ulm, Germany, 2004
- [Spence 05] M. Spence, C. Driver, S. Clarke. "Sharing Context History in Mobile, Context-Aware Trails-Based Applications", Department of Computer Science, Trinity College Dublin, Ireland, 2005
- [Wachowski 90] Andy Wachowski, Larry Wachowski. "V for Vendetta", early 90's, www.imsdb.com/scripts/V-for-Vendetta.html (October 2006)
- [Wang 04] X. Wang, et al., "Ontology-Based Context Modeling and Reasoning using OWL". Context Modeling and Reasoning Workshop at PerCom 2004. citeseer.ist.psu.edu/wang04ontology.html
- [Weiser 91] Weiser M. "The computer for the 21st century", in Scientific American, 1991, 265 (3): 94 – 104

- [Wiki: Ami] en.wikipedia.org/wiki/Ambient_Intelligence (July 2007)
- [Wiki: OWL] en.wikipedia.org/wiki/Web_Ontology_Language (June 2007)
- [Wiki: RDF] en.wikipedia.org/wiki/Resource_Description_Framework (June 2007)

C.2 Ambient Intelligence scenarios

Home scenarios

- [AmI Light 02] E. Diederiks, B. Eggen, J. van Kuijk. “*Scenarios for Ambient Intelligent Lighting*”, Technical note, Philips Research / Philips Lighting, Eindhoven, April 2002
- [Amigo 05] C. Magerkurth, N. Streitz, Fraunhofer, M. Janse, N. Portolan, M. Barone, S. di Marco, A. Larrannaga, I. Lucas, J. Arribas, S. Carro-Martinez. “*Report on User Requirements: State of the Art, Volume II*”, IST Amigo project, April 2005
- [Hydrogen 00] H. Eggenhuisen, G. Jorna, “*Hydrogen Workshop Report, Selected scenarios and trend information*”, Philips Design, Eindhoven, December 2000
- [Intelligent 05] C. Röcker, M. Janse, N. Portolan, N. Streitz. “*User requirements for intelligent home environments: a scenario-driven approach and empirical cross-cultural study*”, in Proceedings of the 2005 joint conference on Smart objects and ambient Intelligence, ACM Press, New York, October 2005
- [ISTAG 01] Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J., Burgelman, J. C. “*ISTAG: Scenarios for Ambient Intelligence in 2010*”, CORDIS IST, ISTAG, February 2001
- [MUSCLE 05] R. van de Sluis, E. Diederiks, L. Geurts, A. Andrews. “*Selected Multi-user Issues for MUSCLE and SoCoHo*”, Philips Research / Philips Design, MUSCLE project / SoCoHo project, Eindhoven, April 2005
- [Portable 01] I. Halters, V. Buil, R. van de Sluis. “*Portable Displays in the Home, Interaction and Application Scenarios for Context Awareness*”, Philips Research, Commuter project, Eindhoven, August 2001

Retail scenarios

- [IILAH 05] J. Witsenburg, E. Diederiks, R. van de Sluis. “*Shop lighting scenarios, Gaining domain knowledge and evaluating scenarios with shop owners, shoppers and retail experts*”, Philips Research, Media Interaction group, IILAH project, Eindhoven, August 2005

- [Pervasive 02] G. Roussos, J. Tuominen, L. Koukara, O. Seppala, P. Kourouthanasis, G. Giaglis, J. Frissaer. "*A case study in pervasive retail, Proceedings of the 2nd international workshop on Mobile commerce*", Atlanta, Georgia, USA, September 2002
- [SWD 05] T. Lashina, E. van Loenen, G. Hollemans, K. van Gelder, M. van Doorn, S. van de Wijdeven, V. Buil. "*Shop Window Display brainstorm*", Philips Research, Media Interaction group, DreamScreen project, Eindhoven, July 2005

D Appendices

D.1 Brainstorm results

Brainstorm ideas

1. Finding faster what you are looking for.
2. What is the fastest line at the cash register?
3. A smart purchase list.
4. What do I still have at home?
5. What to do when a product is unavailable?
6. Product A needs product B, where do I find it?
7. How to pack everything in the shopping cart.
8. Info about what may I have. (think about calories, allergies, etc.)
9. Choosing a mood to be guided around. (i.e. for an Italian night)
10. Where are my kids?
11. Finding your way in an unknown supermarket.
12. Scent applications.
13. Try out corners.
14. Shopping cart anti theft system.
15. Shopping cart automatic return feature.
16. Recommender when you forgot something.
17. No shopping cart, just pointing at products.
18. Presenting alternative products.
19. Coinless shopping carts.
20. Checking the stock for more products.
21. An automatic cash register.
22. Automatic warnings for over date products.
23. Getting recipe info.

Brainstorm idea clusters

Path finding (ideas 1, 6, 10, 11)

- Location
 - self
 - target (product / person) (3d)
 - route
 - car (bring products automatically to car)
 - absolute (within shop)
 - relative (on a fixed route)
- Identity
 - of the product (size, weight, price, readable by shopping cart)
 - of a person (child)
 - role (clients, staff)
- Time
 - in a hurry
 - shopping for social contacts

- Personal
 - vegetarian, allergies, kosher, illness, disabled, intellectual state
 - alone / with a group
 - good or bad intensions
 - mood (feel like cooking or not)
- History
 - what do you usually buy
 - how much time do you usually spend

Product info (ideas 2, 7, 20, 21, 22)

- Contents of shopping cart
 - Identity
 - Order
- Identity
 - of the products (size, weight, price, date)
 - of a person (average speed of cash register employee)
- Time

Recipe info (ideas 3, 5, 8, 9, 16, 18, 23)

- Identity
 - of the product (locations, size, weight, price, ingredients, taste, scent, colour, alternatives, substitutes)
- Contents of shopping cart
 - Identity
- Personal
 - preferences
 - vegetarian, allergies, kosher, illness, disabled, intellectual state
 - mood (feel like cooking or not)
- History
 - what do you usually buy

Smart Carts (ideas 14, 15, 19)

- Location
 - self (cart)
 - movement (speed, direction)
 - entities nearby
- Contents of shopping cart

Other (ideas 4, 12, 13, 17)

- Products at home
 - in cupboards, in fridge, etc
 - amount
- Identity
 - of the product (locations, weight, price, ingredients, taste, scent, colour)
- Gestures (pointing, touching, gazing, etc)

D.2 Movie scripts analysis

Slither [Gunn 2004]

Activity	Can be modelled by
bell ringing	sound
chattering	speech + face expression
giggling	speech + face expression
humming	sound
kneeling	movement
laughing	speech + face expression
listening	activity
locking a door	movement + gestures
looking	orientation + gaze
lounging	movement + activity
moving towards a door	movement
nodding	movement
opening a door	movement + gestures
pointing	gestures
raising a hand	gestures
smiling	face expression
snickering	speech + face expression
staring	orientation + gaze
staring in a mirror	orientation + gaze
starting to leave	movement
swirling towards	movement
trembling highly	movement + activity
turning towards a door	orientation
walking into a room	movement
walking towards a door	movement
yelping with fear	speech + face expression

V for Vendetta [Wachowski 1990]

Activity	Can be modelled by
being afraid	face expression
being outraged	face expression
being relieved	face expression
being stunned	face expression
bending	movement
dragging	movement + gestures + activity
dropping	movement
widening your eyes	face expression
giving a speech	speech + face expression
grabbing	gestures
grinning	face expression
handing it to her	gestures

hooting	speech
howling	speech
lifting your arm	gestures
light sweeping across a room	light
pausing	movement + activity
phone ringing	sound
popping a beer	gestures + sound
punching a button	gestures
rubbing tears from your eyes	gestures
rushing out of a room	movement
screaming	speech + face expression
singing	speech
sneaking up	movement
starting to cry	speech + face expression
whispering	speech

Willow [Lucas 1988]

Activity	Can be modelled by
being dragged	movement
bouncing	movement
carrying a baby	activity + entities nearby
clumping together	movement + entities nearby
falling	movement + orientations
flaring	face expression
flashing your eyes	face expression
folding	gestures
igniting	gestures + sound + light
jumping	movement
kissing	movement + face expression + entities nearby
poking	gestures
pounding	movement + gestures
rolling	movement
scrutinizing	orientation + gaze
shaking your head	movement
shrugging	movement
shutting his eyes	face expression
sliding	movement
snapping your finger	movement + sound
squinting	face expression
turning away	movement + orientation

D.3 Home Scenarios

ISTAG: Scenarios for Ambient Intelligence in 2010 [ISTAG 01]

1. Dimitrios, the digital me
2. Carmen, traffic, sustainability & commerce

Portable Displays in the Home, Interaction and Application Scenarios for Context Awareness [Portable 01]

3. All

User requirements for intelligent home environments: a scenario-driven approach and empirical cross-cultural study [Intelligent 05]

4. All

Selected Multi-user Issues for MUSCLE and SoCoHo [MUSCLE 05]

5. Sharing one large display
6. Ad hoc sharing of activities and content on the move
7. Seamless continuation of activities in a social setting
8. Ad hoc transition of single-user to multi-user activities
9. Co-creation of a media album or play list

Hydrogen Workshop Report, Selected scenarios and trend information [Hydrogen 00]

10. Louise
11. Monitoring Inlay
12. Elderly Home
13. Position Aware
14. Person Aware
15. Context aware personal assistant
16. HomeCast
17. Interactive Table
18. Bathroom mirror
19. A moment's peace & quiet
20. Cyber pets
21. Friday the 13th

Scenarios for Ambient Intelligent Lighting [Aml Light 02]

22. Activity Spot
23. Light Timer 2
24. Tale-Telling Stones
25. Almost Home
26. Night Lights
27. Teaching Surroundings
28. Easy Light / A Warm Welcome
29. The Moody Blues

Report on User Requirements: State of the Art, Volume II [Amigo 05]

30. Voice command based home

- 31. Gemini: Accumulating Context for Play Applications
- 32. Context Aware Information Retrieval in the Home

D.4 Retail Scenarios

Shop lighting scenarios, Gaining domain knowledge and evaluating scenarios with shop owners, shoppers and retail experts [IILAH 05]

- 33. Microclimates / Light bubble
- 34. Reactive Spots
- 35. Reactive Counter
- 36. Natural Light / Daylight attraction
- 37. Light trendsetter
- 38. Full interaction
- 39. Adaptive spotlight
- 40. Anti-shoplifter lighting
- 41. Anti night-visitor lighting
- 42. Reactive clothing rack
- 43. Point 'n' aim

A case study in pervasive retail, Proceedings of the 2nd international workshop on Mobile commerce [Pervasive 02]

- 44. All

Shop Window Display brainstorm [SWD 05]

- 45. MTS01, 02, KV02
- 46. MTS06, 08, 11
- 47. MTS07, 12
- 48. MTS09
- 49. MTS13, 14
- 50. GE01
- 51. GE07
- 52. KV03
- 53. ETS04
- 54. KG11
- 55. EV01, KTS01
- 56. KTS03
- 57. MG01
- 58. MG03
- 59. MG08

Supermarket Brainstorm

- 60. Path finding
- 61. Product info
- 62. Recipe info
- 63. Smart Carts
- 64. Miscellaneous

D.5 XML schema

```
<?xml version="1.0" encoding="utf-8"?>
<xs:schema xmlns="" id="context" xmlns:xs="http://www.w3.org/2001/XMLSchema">
  <xs:element name="position">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" name="latitude" type="xs:string" />
        <xs:element minOccurs="0" name="longitude" type="xs:string" />
        <xs:element minOccurs="0" name="altitude" type="xs:integer" />
        <xs:element minOccurs="0" name="anchor" type="xs:string" />
        <xs:element minOccurs="0" name="angle" type="xs:integer" />
        <xs:element minOccurs="0" name="distance" type="xs:integer" />
      </xs:sequence>
      <xs:attribute name="class" use="required">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="relative" />
            <xs:enumeration value="absolute" />
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
      <xs:attribute name="system">
        <xs:simpleType>
          <xs:restriction base="xs:string">
            <xs:enumeration value="xyz" />
            <xs:enumeration value="gps" />
          </xs:restriction>
        </xs:simpleType>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
  <xs:element name="orientation">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="angle" type="xs:integer" />
        <xs:element minOccurs="0" name="tilt" type="xs:integer" />
        <xs:element minOccurs="0" name="roll" type="xs:integer" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="health">
    <xs:complexType>
      <xs:simpleContent>
        <xs:extension base="xs:string">
          <xs:attribute name="class" type="xs:string" use="required" />
        </xs:extension>
      </xs:simpleContent>
    </xs:complexType>
  </xs:element>
  <xs:element name="intention">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="activity" type="xs:string" />
        <xs:element minOccurs="0" name="duration" type="xs:duration" />
        <xs:element minOccurs="0" name="speed" type="xs:integer" />
        <xs:element minOccurs="0" name="angle" type="xs:integer" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="actor">
    <xs:complexType>
      <xs:sequence>
        <xs:element minOccurs="0" name="arrival" type="xs:time" />
        <xs:element minOccurs="0" name="role" type="xs:string" />
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

```

    <xs:element minOccurs="0" name="age" type="xs:positiveInteger" />
    <xs:element minOccurs="0" name="gender" type="xs:string" />
    <xs:element minOccurs="0" name="length" type="xs:positiveInteger" />
    <xs:element minOccurs="0" name="face" type="xs:anyURI" />
    <xs:element minOccurs="0" ref="position" />
    <xs:element minOccurs="0" ref="orientation" />
    <xs:element minOccurs="0" maxOccurs="unbounded" ref="health" />
    <xs:element minOccurs="0" ref="intention" />
  </xs:sequence>
  <xs:attribute name="id" type="xs:string" use="required" />
</xs:complexType>
</xs:element>
<xs:element name="contains">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="actor" />
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="prop" />
    </xs:sequence>
    <xs:attribute name="in_order" type="xs:string" />
    <xs:attribute name="count" type="xs:integer" />
  </xs:complexType>
</xs:element>
<xs:element name="prop">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="name" type="xs:string" />
      <xs:element minOccurs="0" name="capability" type="xs:string" />
      <xs:element minOccurs="0" name="color" type="xs:string" />
      <xs:element minOccurs="0" name="width" type="xs:positiveInteger" />
      <xs:element minOccurs="0" name="height" type="xs:positiveInteger" />
      <xs:element minOccurs="0" name="depth" type="xs:positiveInteger" />
      <xs:element minOccurs="0" name="weight" type="xs:positiveInteger" />
      <xs:element minOccurs="0" ref="position" />
      <xs:element minOccurs="0" ref="orientation" />
      <xs:element minOccurs="0" ref="contains" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required" />
    <xs:attribute name="class" type="xs:string" />
  </xs:complexType>
</xs:element>
<xs:element name="condition">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" name="air_quality" type="xs:positiveInteger" />
      <xs:element minOccurs="0" name="color" type="xs:string" />
      <xs:element minOccurs="0" name="light" type="xs:positiveInteger" />
      <xs:element minOccurs="0" name="music" type="xs:string" />
      <xs:element minOccurs="0" name="sound" type="xs:positiveInteger" />
      <xs:element minOccurs="0" name="temperature" type="xs:integer" />
      <xs:element minOccurs="0" name="rain_chance" type="xs:positiveInteger" />
      <xs:element minOccurs="0" name="water" type="xs:string" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
<xs:element name="performance">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="activity" type="xs:string" />
      <xs:element minOccurs="0" name="start" type="xs:time" />
      <xs:element minOccurs="0" name="speed" type="xs:integer" />
      <xs:element minOccurs="0" name="acceleration" type="xs:integer" />
      <xs:element minOccurs="0" name="angle" type="xs:integer" />
      <xs:element minOccurs="0" maxOccurs="unbounded" name="words" type="xs:string" />
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="actor" />
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="prop" />
    </xs:sequence>
  </xs:complexType>

```

```

</xs:element>
<xs:element name="relation">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="object" type="xs:string" />
      <xs:element name="subject" type="xs:string" />
    </xs:sequence>
    <xs:attribute name="class" type="xs:string" use="required" />
  </xs:complexType>
</xs:element>
<xs:element name="stage">
  <xs:complexType>
    <xs:sequence>
      <xs:element minOccurs="0" ref="condition" />
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="performance" />
      <xs:element minOccurs="0" maxOccurs="unbounded" ref="relation" />
    </xs:sequence>
    <xs:attribute name="id" type="xs:string" use="required" />
    <xs:attribute name="indoor_outdoor" type="xs:string" />
  </xs:complexType>
</xs:element>
<xs:element name="context">
  <xs:complexType>
    <xs:sequence>
      <xs:element maxOccurs="unbounded" ref="stage" />
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:schema>

```

D.6 User test scenarios

Scenario I

“After dinner, a smart home senses that there are many people (guests) in the living room. It thinks it is a party, so it plays nice music and gives visualizations on the windows of the living room.”

```
<beat id="afterDinnerParty">

  <!-- preconditions that must hold for this beat to be selected -->
  <pre>
    <stage id="LivingRoom" min_time="17:00:00">
      <performance>
        <actor min_count="5">
          <health class="emotion">happy</health>
        </actor>
      </performance>
    </stage>

    <script>
      <present />
      <history>
        <active>
          <beat id="Dinner" min_period="-02:00:00" />
        </active>
      </history>
    </script>
  </pre>

  <!-- what happens when the beat is selected -->
  <action>
    :
    :
  </action>

</beat>
```

Scenario X

“A shop window senses a person walking in the cold and the rain, not close to the shop window, with a dog and an umbrella and looking straight ahead. It responds to this in a very specific way to attract the person's attention.”

```
<beat id="badWeatherMan">

  <!-- preconditions that must hold for this beat to be selected -->
  <pre>
    <stage id="outside">
      <condition>
        <max_temperature>5</max_temperature>
        <water>yes</water>
      </condition>

      <performance>
        <prop id="ShopWindow"/>
      </performance>

      <performance>
        <prop id="$umbrella" class="umbrella"/>
      </performance>
    </stage>
  </pre>

  <!-- what happens when the beat is selected -->
  <action>
    :
    :
  </action>

</beat>
```

```

<performance>
  <activity>walking</activity>

  <actor id="$actorX">
    <role>customer</role>
    <position class="relative">
      <anchor>ShopWindow</anchor>
      <width>800</width>
      <length>300</length>
      <offset_angle>90</offset_angle>
      <offset_distance>400</offset_distance>
    </position>
    <orientation>
      <min_angle>20</min_angle>
      <max_angle>-20</max_angle>
    </orientation>
  </actor>

  <actor>
    <role>animal</role>
    <position class="relative">
      <anchor>$actorX</anchor>
      <max_distance>200</max_distance>
    </position>
  </actor>
</performance>

  <relation class="touch">
    <object>$actorX</object>
    <subject>$umbrella</subject>
  </relation>
</stage>

  <script>
    <present />
    <history />
  </script>
</pre>

  <!-- what happens when the beat is selected -->
  <action>
    :
    :
  </action>
</beat>

```


D.7 Software test scenarios

Below the beat “interactionPointPrice” is shown. This beat will get activated when a person, standing in front of the intelligent shop window, points at a certain product and says the word “price”.

```
<beat id="interactionPointPrice" class="interactionMode">

  <!-- preconditions that must hold for this beat to be selected -->
  <pre>
    <stage id="outside">
      <performance>
        <activity>speaking</activity>
        <words>price</words>

        <actor id="$actor">
          <position class="absolute">
            <min_latitude>0</min_latitude>
            <max_latitude>200</max_latitude>
            <min_longitude>0</min_longitude>
            <max_longitude>200</max_longitude>
          </position>
        </actor>
      </performance>

      <performance>
        <prop id="$product" />
      </performance>

      <relation class="point">
        <object>$actor</object>
        <subject>$product</subject>
      </relation>
    </stage>

    <script>
      <present />
      <history />
    </script>
  </pre>

  <!-- what happens when the beat is selected -->
  <action>
    :
    :
  </action>
</beat>
```

In the beat above, the position of the actor is defined absolutely. We will show the same beat once again, but now with relative positioning.

```
<beat id="interactionPointPrice" class="interactionMode">

  <!-- preconditions that must hold for this beat to be selected -->
  <pre>
    <stage id="outside">
      <performance>
        <activity>speaking</activity>
        <words>price</words>
```

```

    <actor id="$actor">
      <position class="relative">
        <anchor>ShopWindow1</anchor>
        <width>150</width>
        <length>150</length>
        <offset_angle>90</offset_angle>
        <offset_distance>75</offset_distance>
      </position>
    </actor>
  </performance>

  <performance>
    <prop id="$product" />
  </performance>

  <relation class="point">
    <object>$actor</object>
    <subject>$product</subject>
  </relation>
</stage>

<script>
  <present />
  <history />
</script>
</pre>

  <!-- what happens when the beat is selected -->
  <action>
    :
    :
  </action>
</beat>

```