



Deliverable 8
The AmbieSense Multi-Agent System,
Architecture Report

**WP6 Intelligent Agents for Context-Sensitive
Services**

Author(s)

Till C. Lech(ed.),

Marius Mikalsen, Leendert Wienhofen,

Anders Kofod-Petersen, Stuart Watt, Hans Inge Myrhaug,

Frank Sembowski, Martí Bayo Alemany,

Murat Yakici, Ralf Bierig

<h1>Report</h1>	
REPORT TITLE	
The AmbieSense Multi-Agent Framework, Architecture Report	
RESPONSIBLE WORK PACKAGE	WORK PACKAGE LEADER
WP6	CognIT
WRITTEN BY	
Till C. Lech (ed., CognIT), Marius Mikalsen (SINTEF), Leendert Wienhofen, Robert Engels (CognIT), Stuart Watt, Ralf Bierig and Murat Yakici (RGU), Hans Inge Myrhaug(SINTEF), Anders Kofod-Pedersen (NTNU), Frank Sembowski and Martí Alemany (YellowMap),	
ABSTRACT	
<p>This document is the 8th deliverable of the AmbieSense project. It describes the architecture for the AmbieSense Multi-Agent System. Software agents can be described as computer programs that behave goal-oriented in an autonomous manner. Multi-Agent systems are environments, where multiple software agents interact in order to execute complex tasks for the user.</p> <p>In the AmbieSense project, a multi-agent system is used in order to create an infrastructure for the context-aware delivery of relevant content to mobile users. This document describes the agent framework and the enabling technologies used in the multi-agent system in order to achieve this task. It describes the inner architecture of the AmbieSense agency and how it can be integrated with the AmbieSense Content Provision Services.</p>	
REPORT STATUS	KEYWORDS
[final version]	Multi-Agent Systems, context-aware information systems, mobile computing
INITIATED DATE	WRITTEN DATE
20-11-2003	20-01-2004
FILE CODE	CLASSIFICATION
[AmbieSense\deliverables]	Public
ELECTRONIC FILE CODE	
AmbieSense internal web: [Ambiesense > Project deliverables]	

This document is part of a research project funded by the IST Programme of the Commission of the European Communities as project number IST 2001-34244. The contractors in this project are: SINTEF (co-ordinator) Norway; Siemens AG, Austria; The Robert Gordon University, UK; CognIT AS, Norway; YellowMap, Germany; The Norwegian University of Science and Technology, Norway; Oslo Airport, Norway; and Seville Global, Spain.

SINTEF Telecom and Informatics

Contact person: Hans Inge Myrhaug
E-mail: hansim@sintef.no

CognIT AS

Contact person: Robert Engels
E-mail: robert.engels@cognit.no

YellowMap AG

Contact person: Jürgen Pollich
E-mail: juergen.pollich@yellowmap.de

Oslo Airport AS

Contact person: Jon Henrik Thorshaug
E-mail: jon-henrik.thorshaug@osl.no

Siemens AG Austria

Contact person: Erwin Postmann
E-mail: erwin-postmann@siemens.com

The Robert Gordon University

Contact person: Ayşe Göker
E-mail: asga@scms.rgu.ac.uk

Norwegian University of Science and Technology

Contact person: Agnar Aamodt
E-mail: agnar.aamodt@idi.ntnu.no

Sevilla Global

Contact person: Beatriz Real
E-mail: breal@sevillaglobal.es

Table of contents

EXECUTIVE SUMMARY	6
1 INTRODUCTION – ROADMAP TO THE DOCUMENT.....	7
1.1 PURPOSE OF THE DOCUMENT: COMMUNICATING THE A-MAS ARCHITECTURE	7
1.2 METHODOLOGY	7
1.3 EVALUATION.....	8
2 MULTI-AGENT SYSTEMS AND AMBIESENSE.....	9
2.1 INTRODUCTION TO AGENT TECHNOLOGY.....	9
2.1.1 <i>The Essential Characteristics of an Agency</i>	10
2.1.2 <i>Agents and Roles</i>	12
2.1.3 <i>The ‘Essential’ Characteristics of Agency Revisited</i>	13
2.2 MOTIVATION FOR USING AGENTS IN AMBIESENSE.....	15
3 AMBIESENSE MULTI-AGENT SYSTEM - FEATURES AND INNOVATION	17
3.1 THE A-MAS – AN INFRASTRUCTURE FOR THE DISTRIBUTION OF RELEVANT INFORMATION	17
3.1.1 <i>Using context middleware</i>	18
3.1.2 <i>Encapsulating case-based reasoning</i>	18
3.1.3 <i>Using the Context Ontology (Knowledge Model)</i>	19
3.1.4 <i>Enabling security and privacy</i>	19
3.1.5 <i>Encapsulating and integrating with CORPORUM™ Intelligent Agents</i>	19
3.1.6 <i>Integrating with the Content Integration Platform</i>	19
3.2 ORGANISATION OF THE A-MAS	19
4 SECURITY AND PRIVACY IN THE A-MAS.....	22
4.1 SECURITY IN MULTI-AGENT SYSTEMS	22
4.1.1 <i>Security in Message Transmission</i>	22
4.1.2 <i>Protection of Content</i>	22
4.1.3 <i>Protection of Context</i>	22
4.2 AGENT SECURITY AND EC POLICY.....	23
4.3 CONCLUSION ON AGENT SECURITY	24
5 ARCHITECTURE	26
5.1 AGENTS IN THE AMBIESENSE ARCHITECTURE	26
5.2 THE FIPA REFERENCE ARCHITECTURE	27
5.3 AGENT TYPES - FUNCTIONALITY	29
5.3.1 <i>Context Agent</i>	30
5.3.2 <i>Personalisation Agent</i>	30
5.3.3 <i>Recommender Agents</i>	30
5.3.4 <i>Content Agent</i>	30
5.3.5 <i>Interface Agent (UIAgent)</i>	30
5.3.6 <i>Search Agent</i>	30
5.3.7 <i>Integration Agents</i>	30
5.4 AGENT TYPES – IMPLEMENTATION	31
5.4.1 <i>Context Agent</i>	31
5.4.2 <i>Search Agent</i>	32
5.4.3 <i>User Interface Agent</i>	34
5.4.4 <i>Personalisation Agent</i>	35
5.4.5 <i>Content Agent</i>	36
5.4.6 <i>Recommender Agent</i>	36
5.4.7 <i>AmbieSense Ontology</i>	39

5.4.8	<i>Integration Agents</i>	41
5.4.9	<i>Corporum Integration</i>	42
5.4.10	<i>Integration of the YellowMap Service</i>	43
6	SUMMARY	45
7	APPENDICES	46
	APPENDIX 1: SURVEY – MOTIVATION FOR USING JADE/LEAP.....	46
	<i>Requirements for Agents in AmbieSense</i>	46
	<i>Agent Framework Survey Summary</i>	49
	<i>Conclusion</i>	50
	<i>Framework</i>	50
	APPENDIX 2: TEST PLAN FOR THE A-MAS	51
	APPENDIX 3: GLOSSARY.....	52
	APPENDIX 4: CLASS DIAGRAMS	53
	APPENDIX 5: CLASS DESCRIPTIONS	58
	APPENDIX 6: TABLE OF FIGURES	64
	APPENDIX 7: REFERENCES	65

Executive Summary

This document describes a scalable, reusable infrastructure for the context-aware delivery of content – the AmbieSense Multi-Agent System (A-MAS). The purpose of this document is to communicate both the background and motivation for using agent-based technology in AmbieSense, and to provide guidelines for the further design and implementation of the A-MAS. The implemented multi-agent system will be provided and documented in AmbieSense Deliverable 9, “Intelligent, personalised agents for mobile use” (to be published in April 2004).

There are many views and definitions of intelligent agents. However, in AmbieSense, we take two particular perspectives:

- a) The lower-level perspective – multi-agent systems as platform for protocol and communication
- b) The metaphor of agents and human-like behaviour.

With regards to a), the architecture uses JADE/LEAP as the agent development environment of choice for AmbieSense, which will provide a proven, well-documented platform for the intelligent agents in AmbieSense. Concerning b), the high level, intelligent functionality is provided by a number of enabling state-of-the-art technologies such as Information Retrieval mechanisms, Case-Based reasoning, personalisation modules and the AmbieSense content base components. The main objective for the design of the A-MAS was to use a multi-agent system as a platform for the combination of enabling state-of-the-art technologies in order to satisfy the needs of mobile users in a distributed setting.

Agent properties such as autonomy, reactivity, goal-orientedness and communicativeness, together with the fact that agents are well-suited for distributed systems make the agent programming paradigm a natural choice for involvement in certain tasks in AmbieSense. Among these agent-supported tasks are detection and maintenance of user contexts as well as the retrieval of content, where intelligent agents have proven to be useful in a number of systems. The AmbieSense agents can in principle be deployed and run on all three main components of any AmbieSense solution in the future:

- The wireless Context Tag
- The mobile computers
- The content service provision platform

Thus, by designing a scalable agent system that is possible to use in distributed system settings, the A-MAS will have a value of its own, making it possible to exploit the multi-agent system in other context-aware systems after the project’s end.

In addition to being scalable and reusable, the A-MAS shall contribute to agent standardisation by reusing the JADE/LEAP platform, the latter of which was developed in the EU IST project IST-1999-1021. The process of selecting this framework, along with a survey of currently available agent development platforms is also described in this document.

1 Introduction – Roadmap to the Document

1.1 PURPOSE OF THE DOCUMENT: COMMUNICATING THE A-MAS ARCHITECTURE

This document is the 8th deliverable of the AmbieSense project. It describes the Architecture of a Multi-Agent System, a Java library that will be used in the AmbieSense project as an infrastructure for the context-aware distribution of content to mobile users. The Architecture Report serves three different purposes:

- To give a detailed account of the Multi-Agent software that is going to be delivered in the AmbieSense project, consisting of the JADE framework, AmbieSense agents and enabling technologies
- To be a reference for service vendors that want to use the infrastructure the A-MAS provides for context-aware services
- To communicate the architecture of the AmbieSense Multi-Agent System (A-MAS) the ambient/context-aware computing community.

The different purposes of the document imply different target audiences for the A-MAS architecture report. On the one hand there are readers with little technical background. These readers will find a sufficient description of the system's functionality and how it can be plugged into enterprise systems and content repositories in Chapters 2-4.

Chapter 2 provides an overview of agents and agent-based technology and how they fit into the AmbieSense domain. Chapter 3 describes the AmbieSense Multi-Agent System from a computational and functional viewpoint. In Chapter 4, an analysis of security and privacy issues is given.

Chapter 5 then gives a technical, detailed overview of the A-MAS architecture. In addition to references, tables and figures lists, the appendices contain a section on the evaluation process leading to JADE/LEAP as the platform of choice. Furthermore, class diagrams of the agent classes in the A-MAS as well as an overview of the testing schedule for the software are provided.

1.2 METHODOLOGY

When documenting agent technology, there is no standard to adhere to. Shortcomings in standard UML have been identified, since UML is neither equipped to depict the roles of software agent, nor can it account for the complexity and interactivity of agent behaviours. UML-extensions such as Agent UML¹ (AUML) are being discussed in the agent community but have not yet been standardised. In this document we will take the following approach:

- The component descriptions start off with a functional view to the A-MAS and its components. Verbal descriptions of agents and enabling technologies will be given. These descriptions shall suffice for grasping the functionality of the A-MAS and how it fits into the overall AmbieSense system.
- After the verbal descriptions, the account of the system components gets more technical. In addition to the detailed descriptions of the agents and their behaviours, AUML sequence diagrams are used to visualise how the agents in the AmbieSense agency interact. Class diagrams of agents

¹ <http://auml.org>

and their behaviours will be provided in the appendix. Furthermore, wherever agents interact with system components outside the JADE/LEAP platform, detailed descriptions of classes and interfaces will be provided. This documentation will be the starting point for the API of the A-MAS which will be published in the 9th AmbieSense deliverable: Intelligent, personalised agents for mobile use” (to be published in April 2004).

1.3 EVALUATION

In addition to continuous in-house component testing, the AmbieSense Multi-Agent System is going to be tested and evaluated at AmbieSense test sites at Oslo Airport and in Seville. The A-MAS will be tested in the four AmbieSense applications:

- Airport Information Service
- Map Service
- Travel Guide Service
- News/Infotainment Service

Test results will be provided in a qualitative manner (interviews with test users) as well as quantitative; system performance, analysis of user questionnaires.

Test and Evaluation results will be published in the forthcoming AmbieSense Test and Evaluation Report (12th deliverable of the AmbieSense project). A detailed test plan for the A-MAS is provided in the Appendix 2 of the document.

2 Multi-Agent Systems and AmbieSense

2.1 INTRODUCTION TO AGENT TECHNOLOGY

Agents have for a while been an important concept in computing, but often all that the word refers to is a computational process or task with a capability for autonomous action, either alone or in an artificial society of similar agents. ‘Agent’ is a difficult word for a difficult concept; covering a range of concepts that span a gamut of different kinds of behaviour, including, for example, autonomy, learning, and social interaction. However, there is a common ground. An agent will set out to do something, and do it; therefore it has competences for intending to act, for action in an environment, and for monitoring and achieving its goals. Of course, for adequate performance of these, other competences, such as learning, negotiation, and planning, may be helpful or even necessary.

Before continuing, we should probably clarify this use of the word ‘agent’. The term ‘agent’ originated with Oliver Selfridge, with much of the concept due to joint work with John McCarthy in the 1950s. What they had in mind was an assistant that could be given a task to carry out, and which would carry out that task both intelligently and autonomously. On the one hand, this idea led to artificial intelligence, and on the other, it led to the notion of agents in the human computer interface (Kay, 1990; Maes, 1994). The difference between them is one of emphasis: in artificial intelligence, the emphasis is on how the agent works inside, in human-computer interaction, it is on how the agent is seen to behave by others.

In artificial intelligence agents are described principally in terms of the internal states, the desires, beliefs, and goals, over which the agent has control, and agents are programmed using these components. This raises the problem of what these ‘desires,’ ‘beliefs,’ and ‘goals’ really are, since they are clearly not the same as human desires, beliefs, and goals. Human-computer interaction, by contrast, uses the word ‘agent’ for any active entity that will take on a user’s goals and act on them. The human-computer interaction literature avoids describing what is going on *inside* an agent, falling back to an intuitive definition of agent as something that behaves as if it initiates and performs actions. The difference between the two is fairly simple: are these beliefs, desires, and goals explicitly represented (as in artificial intelligence) or are they metaphors, which the user applies when thinking about the behaviour of a component in a system (as in human-computer interaction).

For this reason, Kay (1990) draws a distinction between “manipulation”, which is what we do with tools, and “management”, which is what we do with agents. We *manipulate* objects, but we *manage* agents. This distinction is an important one (although the boundary between the two is often a bit fuzzy) and we’ll come back to it later. It is important because management is a very different mode of interaction to manipulation; for a start, the quality of feedback isn’t nearly so high, so building good interfaces for management is usually rather harder and requires some rather different qualities.

Let’s keep this issue open, then. For agents that interact with people, issues like permission, responsibility, and trust become far more significant. Human agents are very similar in this respect. For example, an estate agent has a task to do (helping you buy or sell a house) and has (a limited degree of) autonomy in that task. For example, they cannot change the price for you, but they can arrange for people to view your house during your absence. This means that you need to trust the agent to respect your house, and to behave responsibly towards both it and you.

So everyday human agents are responsible (but not wholly responsible), autonomous (but not completely autonomous), and have authority (but not total authority). It is this subtle balance of these qualities between the instigator and the agent — and the agent’s capacity for action within these boundaries — that makes the human agent into an agent. The question is, then, how can we design software agents that both respect these boundaries, and perhaps even more importantly, which ensures that the instigator feels confidence that these boundaries are going to be respected. As with human agents, software agents have a great potential

for freeing people from onerous and tedious tasks, but quite simply, if the instigator doesn't feel that their responsibility and authority will be respected by an agent, they simply won't use the agent.

The question is: why do we need a definition of agency at all? What purpose does it serve, either from an academic point of view or from a practical one? The only possible answer is that if there is an important distinction between agent systems and non agent systems, this distinction must be reflected in either different design rules, or different interaction principles. Either of these is important, and both have been claimed at times.

Design differences between agents and non-agents. Here the claim is that the concept of agent systems, even if not apparent to a user, offers virtues that other design approaches cannot. Wooldridge (1997) argues that multi-agent systems are likely to fundamentally change software engineering for open and distributed systems. Watt *et al.* (1995) argue that multiple agent systems allow different individual or organisational roles in a process to be more explicitly represented than in conventional software. Combined, these suggest that agent-oriented programming (Shoham, 1992) might be a next step beyond object-oriented programming – although it has to be said that so far, the promise of agents in this regard has yet to be clearly realised in practice.

Interaction differences between agents and non-agents. Here it is often claimed that agent systems are easier for people to interact with, at least for certain classes of task, than conventional software. This has been claimed by, among others, Maes (1994), Laurel (1990; 1991), and Ball *et al.* (1996). Maes' argument is typical. She claims that agents offer a way of dealing with the problem of information overload, by allowing people to delegate tasks to a program, more or less permanently. This view is not without its critics, for example, Shneiderman (1997) argues that for many tasks it is better to adopt other approaches which leave the user in control of the task, but which empower the user with tools to deal with the information overload. Central to this lack of consensus is the issue of precisely who, or what, should be in control of a given task.

So the difference between agent and non-agent systems potentially plays two very different roles. In the design process, it can (in principle) help people to design better systems. In the interaction process, it can help us to construct new forms of human-computer interaction. However, the contribution of the agent metaphor to the design process (besides the fact that it is only of subsidiary interest in this forum) does not seem to require a strong definition. As a design tool, the concept of agent can most usefully be applied with an element of flexibility. Furthermore, no element of psychological validity is required, even though (as artificial intelligence frequently borrows ideas from psychology) it might turn out to be useful to borrow agent ideas from human social theories and behaviours.

Let's assume, therefore, that a working distinction between agent and non-agent systems is most useful because it helps us to model and describe the interaction processes between agents, and between agents and people. This implies that the real need is not only for a definition of agency, but for a model of the interaction processes surrounding people and agents.

The idea that agents aren't always useful, helpful, or appropriate, should not be a surprise. The same is true of programs and physical tools as well. We need some guidelines, some theories even, which we can use to help us decide when we should be using an agent, and when a program is sufficient.

2.1.1 The Essential Characteristics of an Agency

Conventionally, agents have been described using definitions. The terms involved in these definitions vary, as does the essentiality of the definitions. Most definitions recognise at the least an element of fuzziness, but broadly speaking the following properties are some of those used more commonly in the literature to describe agent systems.

- Adaptiveness

- Authority
- Autonomy
- Persistence
- Reactivity
- Responsibility
- Sociability

Franklin and Graesser (1996) see the common basis for all these properties in two common uses of the word agent, both of which pre-date computer technology: “1) one who acts, or who can act, and 2) one who acts in place of another with permission”. For them, the second is subsumed by the first, permission is not an issue, and (intended) action within an environment is both necessary and sufficient for being an agent. Unfortunately, their abandonment of the second definition (and its implicit dependence on permission and social relations) is both casual and unjustified. This is particularly the case when looking at the user’s eye view of agent interaction. Ignoring the issues of the agent as *locum tenens*, permission, and responsibility, simply doesn’t make sense when you’re trying to look at the (social) relationship between agents and people.

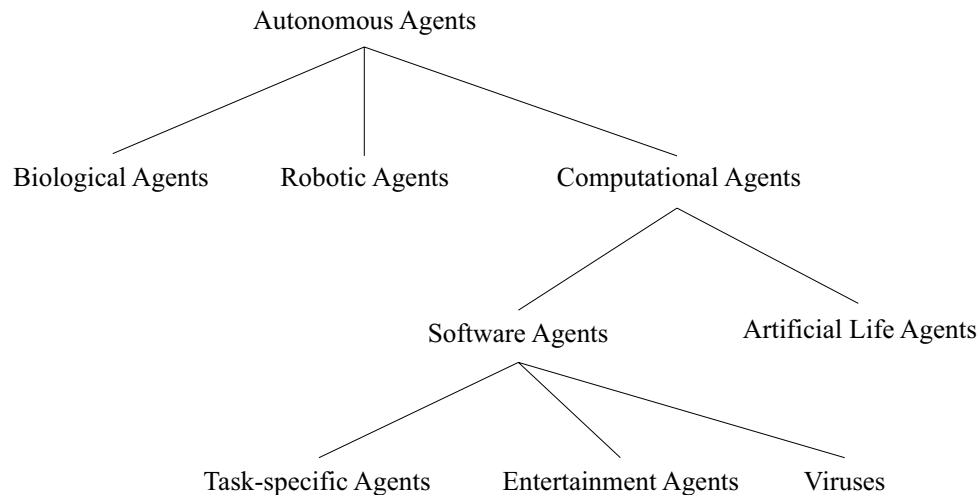


Figure 1: Natural kinds of agency, after Franklin and Graesser (1996)

In effect, the categorisation proposed by Franklin and Graesser (shown in Figure 1) depends on something like natural categories; yet although social categorisation behaves in many ways like natural categorisation, social categories are very different in structure, and the classifications, such as they are, depend on very different features. So although a diagram like this might be useful to system designers (who have a rather functional and architectural view of agency), it is probably less than helpful to interface designers (who have to have a social view of agency). On this basis, we will not follow Franklin and Graesser’s line on agency.

A second point is that while Franklin and Graesser recognise an element of fuzziness that might surround the essence of agency, this is missing the point. It is not simply that there are a few awkward cases that fall around the boundary of what an agent is. Quite simply, we will argue that agents are not defined by an essence at all, but by social and cognitive psychological theories that describe how people see things as agents. Just as a triangle’s definition only makes sense against a background theory of geometry, we argue that a theory of agent interaction is needed before the characteristics of different kinds of agents make any sense. Only with this background theory is a valid taxonomy possible. In order to map out this background theory, though, we first need to study what people tend to mean when they talk about computer agents.

Because defining what an agent is doesn't help very much, so these properties should not be read as definitive of "the essence of agency" (Franklin & Graesser, 1996). In fact, the only reason they are worth writing down is that as common properties and behaviours of actual agent systems, they shape the kinds of design principles that need to be applied to achieve those behaviours. Instead, a better way of understanding what agents are may be by reference to prototypical agents. Among systems which have explicitly been described as prototypical agents is Julia (Foner, 1993).

Following the method of Rosch and Mervis (1975), one of us² asked a group of researchers in the agent field to write down the properties that they felt were most typical of agent systems – the results are shown in Figure 2. Because a rather definitional approach to the concept of agency is prevalent in this field, we asked the participants to give those properties that they personally felt were definitive, without reference to published definitions or sources. It is remarkable that the majority of characteristics mentioned are characteristics shared by people. It is also worth noting that this is an analogue of the approaches from information retrieval as this one being used by RGU in AmbieSense, which use patterns of word frequency to identify which concepts are more or less significant within a field.

2.1.2 Agents and Roles

Agents come in many different guises—the word has a rather technical meaning that is actually used rather rarely, compared to when we're describing these systems to a potential user. (And again, we can see a parallel with 'intelligence' here; systems exploiting artificial intelligence technology are only rarely described as 'intelligent'.) Agents are, instead, often described using a variety of 'role' words that say more about what they *do* than about what they *are*. Role words used to describe agents include: assistant, butler, broker, matchmaker, guide, tutor, buyer, seller, player, opponent, secretary, critic, and facilitator. It is no coincidence that many of these roles are the names of jobs, kinds of employment.

It would be a surprise if the requirements of one role would be the same as those of another. For example, the appropriate presentation of a broker would inevitably be rather different to the presentation of an opponent. Studies in one context (e.g. Koda & Maes, 1996) do not necessarily apply in another. The same level of anthropomorphism may make an opponent seem engaging, but make a broker seem overpowering. Similarly, where trust is of central importance to a broker, it can be a positive disadvantage if you want a realistic opponent. Before choosing the characteristics you need, therefore, it is best to be fairly clear about the task that the agent will face, and the role that it will adopt in relation to the people it will interact with. With this in mind, it is much easier to decide on the appropriate psychological characteristics for the agent to present.

² Research conducted by Stuart Watt, currently at RGU

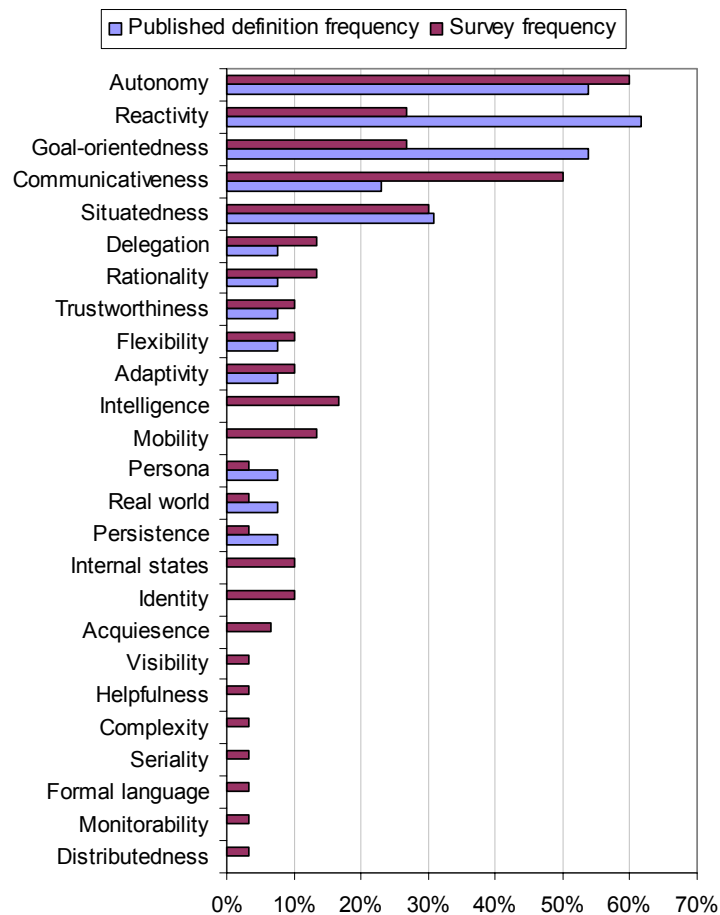


Figure 2: Results of a survey of agent definitions

Surprisingly frequently, agents have more than one role, and then each role will typically have rather different characteristics, and the agent may need to present itself differently to different people according to those roles. A good example of this is Masterton's (1998) "Virtual Teaching Assistant". A teaching assistant has a different role with respect to a tutor (where it must be seen as subordinate) compared to the role it has with respect to a student (where it is typically seen as an advisor). It would be a surprise, then, if the interfaces that represented these very different roles were the same. When this happens, it is worth considering having several different interfaces, one for each role.

Interestingly, although many (e.g. Maes, 1994) talk about people "delegating" tasks to agents, it is far from clear that this is actually what happens. In many ways, the designer of the agent, not the user, is actually delegating the task. Some (e.g. Dickinson, 1998; Shneiderman, 1997) have even argued that delegation might not be the right thing to do in the first place.

2.1.3 The 'Essential' Characteristics of Agency Revisited

We can now return to the apparently essential characteristics that we identified earlier in section 2. Far from being definitive of an essence of agency, the theories that we have reviewed show that these

characteristics follow as consequences of psychological theories. This is important: because we can ground these characteristics in theories rather than in definitions, we can predict more precisely how people will react to these characteristics. This is the foundation that we need for a practical approach to the design of interactive agents.

Perhaps not surprisingly, none of the theories that we have investigated is commonly used in the field of human-computer interaction. However, in the shift from manipulation to management, we are shifting from the physical realm to a psychological one. It should not be a surprise, then, that the theories we need to support this transition are theories from social psychology or social cognition. For now, let's go through some of the characteristics that we viewed as being important to agency, and show how they fit with the theories that we have discussed.

Autonomy. As our study found, autonomy is the most widely asserted characteristic of agency, although a few of our participants were careful to point out that this autonomy was limited. In fact, in his classic study of obedience to authority, Milgram (1974) even draws an opposition between autonomy and agency, and describes the "agentic state" as that in which there is no autonomy. What autonomy is in the context of agents, then, and how it treads the line between freedom and obedience, is far from clear.

Character. A character-based representation was especially important in the field of human-computer interaction. Although this is probably the most-widely criticised aspect of agent interfaces, its role in agency is now slightly clearer. Character-based representations encourage us to take the intentional stance to a system, and therefore to view it as having mental states. As the operating system paradox and attribution theory showed, systems must be visible, and their actions must be clearly attributable, for them to be seen as agents. But over and above this, the characteristics associated with agent systems are overwhelmingly human-like, so anthropomorphism in a psychological sense is required, even if anthropomorphism in a physical sense is not.

Trust. Trust has been cited many times (e.g. Laurel, 1991) as an important part of the design of a good agent. A good agent should be trustworthy. Of course, the nature of trustworthiness is itself a significant problem in its own right. Trustworthiness includes elements of respect, responsibility, privacy, cooperation, predictability, conscientiousness, and even conscience. Attempting to define this seems likely to be both counterproductive and unsuccessful. Having said that, it should not be too surprising that many of these qualities describe the way that people handle different social roles, and particularly on the 'contract' between an agent's role and a person's role. For example, an agent acting as a broker needs to respect a person's privacy.

Tolerance. Tolerance has been included in the list of qualities important to an agent, even though it is not one that has been considered a significant feature of agents anywhere else in the literature. Adding it, therefore, needs some justification. Put simply, tolerance is part of an agent's handling of errors and exceptions, when they happen, without having to resort to an external controller. For the most part, it corresponds to Collins' (1990) notion of 'repair'.

Tolerance is an important quality for an agent because it needs to be able to cope without interference of the user. This is the sense in which agents actually need to be 'autonomous'. It might seem that we have been going on about this one particular quality rather longer than is appropriate, but when we come to look at the problems of managing agent systems, its significance will become a little clearer.

2.2 MOTIVATION FOR USING AGENTS IN AMBIESENSE

The characteristics of Multi-Agent Systems are that

- (1) *each agent has incomplete information or capabilities for solving the problem and, thus, has a limited viewpoint;*
- (2) *there is no system global control;*
- (3) *data are decentralized; and*
- (4) *computation is asynchronous.*

- Katia Sycara, 2003³

After having found that autonomy, character, trust and tolerance are among the "essential characteristics" of any agent system, the natural question to proceed with is indeed: "do these characteristics allow for inclusion of agents within the AmbieSense domain"?

In order to answer this question let us have a look at the following quote, taken from The Economist (October 9,2003):

"In this context, an agent is a program that acts in a self-interested manner in its dealings with numerous other agents inside a computer. This arrangement can mimic almost any interactive system: a stock market; a habitat; even a business supply-chain."

This analysis exactly states what the point is: in order to model a complex, interactive world, we need a system being capable of mimicking this world. Agent Systems and, more specifically, Multi-agent systems are found to be able to do so. Currently such systems have been applied in a variety of worlds.

Within AmbieSense, we have to deal with a domain including users who want to manage tasks in the real world. These tasks include and/or require techniques from the field of Artificial Intelligence: planning, brokering, interacting and understanding. Currently, several initiatives in the MAS world are directed towards integrating and using such technologies within specific domains and much has been written about theory and background of such applications.

However, integrating MAS systems with a user's perspective in which the system is "managed" by a user (as opposed to manipulating, cf. section 2), is something rather unique. A rather novel initiative in the MAS world undertaken within AmbieSense is to model users' wishes into a system that is integrated in a user's everyday life in a non-intrusive manner. The AmbieSense scenario is complex and consists of numerous depending tasks. None of these tasks possess all knowledge needed to perform all other tasks and there is no clear boundary between the users and software within the AmbieSense MAS system. Such kind of so-called "mixed-initiative systems", where users can be represented by agents (and indeed, why not represent agents by users as well?) are not common (although excellent theoretical research on MAS basics can be found, e.g. in [Lyback 2003]).

Within AmbieSense therefore, part of the motivation for using agent systems are based upon non-functional requirements and lie in the fact that the complexity of the interaction of the system's users with their environment, including the sheer diversity of usage domains and contexts, require a modular, component based approach. Various devices for system interaction at rather diverse places posing context sensitive requests to a distributed system made the choice of technology easy. Only Multi-Agent System approaches are capable of delivering the flexibility, which is needed in such a case. Additionally, the non-functional requirements specified for the AmbieSense system, including the requirement on scalability and extendibility into new domains with new users and new contexts, made obvious the need for a MAS-like solution.

³ [Sycara 2003]

Multi Agent Systems have been suggested for Context-Aware Mobile Services, such as proposed by Sadeh et al. [Sadeh 2002], but none of them make use of a holistic approach to the concept of context, as is the case in the AmbieSense project.

Another part of the motivation for using MAS in AmbieSense is caused by the need that is identified for a “mixed-initiative system”⁴, using MAS technology in a non-intrusive, ambient manner and thereby embedding human beings as entities within the A-MAS architecture. In addition, the system should be aware of its surroundings⁵. Therefore, a new mobile and agent-enabling infrastructure has to be developed using a variety of end-user devices that can help agents reach the user in previously “unreachable” situations, examples often being taken from smart phones and the like. Also new is the integration of ontological knowledge within such hardware in order to allow both the devices as well as the connection points a touch of “intelligence” about its own whereabouts.

Summarised, one can say that AmbieSense's goal to provide an innovative environment to mobile, tacit users has led to a system architecture which makes use of innovative and state-of-the-art technologies, and develops a new hardware and human interface infrastructure in order to deliver the defined functionality to its users.

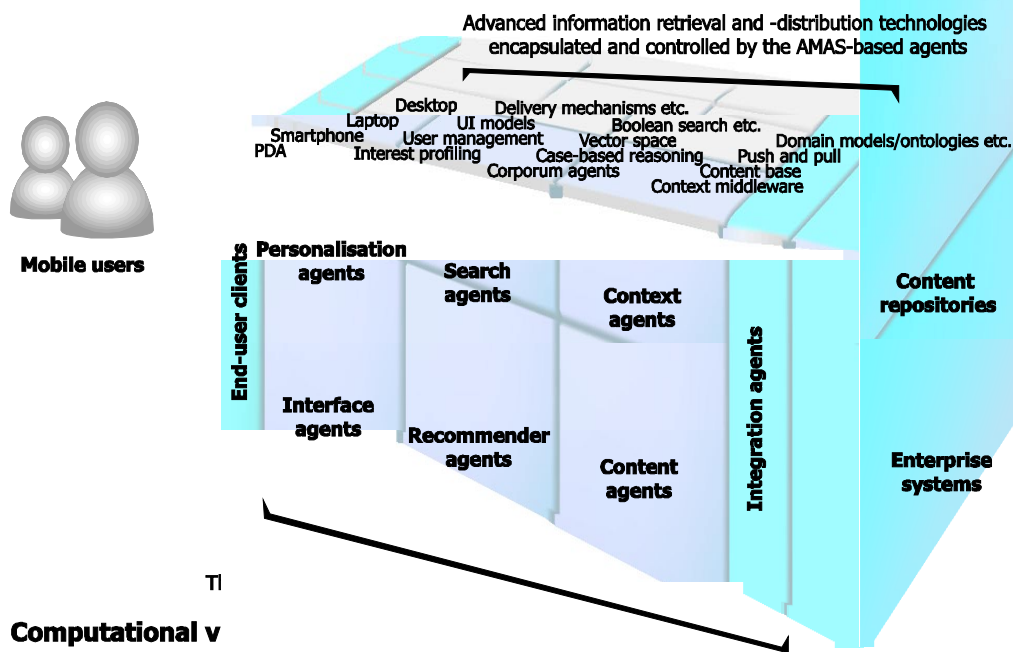
⁴ the consortium had not found evidence for any real-world applications of “Mixed-Initiative Systems” when defining AmbieSense in 2001

⁵ i.e. the system should have knowledge about its surroundings and context, at the same time being able to reason with that knowledge. For this, AmbieSense integrates ontologies as knowledge modelling technique to be used by all agents that need to know about their surroundings and peers.

3 AmbieSense Multi-Agent System - Features and Innovation

3.1 THE A-MAS – AN INFRASTRUCTURE FOR THE DISTRIBUTION OF RELEVANT INFORMATION

AmbieSense is an information system designed to serve the mobile user. While communication and mobility had always been major factors in human society, the synergy of both has gained major success within the last decade: mobile communications. AmbieSense seeks to address the needs of the mobile user by providing better tailored information. As shown in the previous sections, a technology has proven useful in Information Retrieval (IE) and recommender systems. AmbieSense exploits the advantages of State-of-the-Art multi-agent systems, by providing a platform-independent structure for detecting user contexts and delivery of information that suits the needs of users in various contexts.



Computational v AmbieSense Multi-Agent System (A-MAS) architecture

The diagram should be regarded as a proposed reference architecture/ framework (i.e. a blueprint) for the development of general agent-based applications and services, some of which can be context-aware and/or personalised.

Purpose: It is intended to help the system developer to structure and encapsulate various advanced/intelligent methods for information retrieval, storage, and distribution of relevant content to mobile and stationary end-users.

In a specific agent-based application or service, the developer should choose which of the components to re-use/ implement/ not use in order to: 1) meet the requirements and information needs of the end-user, and 2) the content provision needs of the content owner/ content service provider.

Figure 3: The AmbieSense Multi-Agent System Reference Architecture

The A-MAS uses JADE/LEAP⁶, an existing multi-agent framework, for the protocol and communication. The AmbieSense agents that are implemented within the JADE/LEAP framework provide the generic functionality for handling user contexts and triggering content queries. The JADE/LEAP platform was chosen as a result of a State-of-the-Art Survey and evaluation process of agent programming frameworks as

⁶ [Bellifemine 2000], <http://jade.cselt.it>

described in Appendix 7.1. On top of the JADE/LEAP framework the A-MAS integrates intelligent components for context-based content retrieval. The components comprise state-of-the-art technologies such as case-based reasoning for situation recognition, IE and personalisation modules as well as semantic web technology for content classification.

As Figure 3 shows, the infrastructure imposed by the A-MAS consists of three main layers:

- **Core technology agents** that constitute the core of the architecture. There are agents for context handling, Recommender Agents, Search Agents, Personalisation Agents, Content Agents and User Interface Agents. These Agents will be described thoroughly in Chapter 5 of this document.
- **Enabling technologies** that provide the “intelligence” for the core technology agents. This includes technologies for content, context and user management, information extraction and retrieval, as well as reasoning engines. The open architecture of the A-MAS allows for plugging in various systems for handling these tasks. For the AmbieSense project the following technologies will be used: the AmbieSense Context Middleware, a Case-Based Reasoning engine for situation recognition, intelligent CORPORUM agents, a personalisation module as well as the AmbieSense Content Base. Chapter 5 will provide further information on these.
- **Surrounding Components** like User Interfaces on the end-user clients, content repositories, or enterprise systems (e.g. SAP). Content may be accessed through the AmbieSense Content Base or other components of the AmbieSense Content Integration Platform⁷, or directly from the Content Provider’s platform as shown in the Yellow Map integration described in Chapter 5.

The AmbieSense Multi-Agent System combines the agent-based computing paradigm and state-of-the-art technology in order to create a flexible infrastructure that can be re-used by any context-aware information system. The following sections will direct more attention towards the innovative features of the A-MAS that are going to be implemented in the prototype.

3.1.1 Using context middleware

The AmbieSense Context Middleware is a means to overcome the challenges of context management. It plays an integral part of applications that aim to provide context sensitive services and information to mobile users. The Context Middleware deals with the following tasks: context storing, context retrieval, context matching, context merging, context security, context-content linking, context storage synchronization, ease of use and clarity surrounding the notion of context. The A-MAS encapsulates the AmbieSense Context Middleware as one of the enabling technologies for context-aware content provision.

3.1.2 Encapsulating case-based reasoning

As one of the core technology agents the recommender agent utilises Case Based Reasoning (CBR) to assess the user’s current situation. Based on that assessment it tries to recommend the best possible response to the user’s need.

Case Based Reasoning [Aamodt 1994] is concerned with adapting to new situations by remembering similar earlier experienced situations (cases). CBR has historically been used in large monolithic systems. AmbieSense applies CBR as a lightweight reasoning mechanism that is capable of running on a small mobile device.

AmbieSense employs the Creek [Aamodt 1991] CBR environment developed at the Norwegian University of Science and Technology.

⁷ A complete account of the Content Integration Platform will be provided in the AmbieSense deliverable No.6.

3.1.3 Using the Context Ontology (Knowledge Model)

One of the main issues when exchanging information is the ability to share a common understanding of concepts. Without this shared understanding it would be impossible to communicate in any meaningful way. E.g. one entity should not define the concept of a horse as *Equus Caballus* (the mammal), while another entity defines it as: “a mass of the same geological character as the wall rock occurring within a vein” [Gramercy 1996].

To handle this potential Babylonia problem ontologies are used. An ontology is most often defined in the words of Tom Gruber: “An ontology is an explicit specification of a conceptualization.” [2].

Since contextual information comes in many different types, it is of great importance that AmbieSense entities share a common understanding. The Context Ontology defines the concepts and relations used in to communicate and reason about context.

3.1.4 Enabling security and privacy

Security and Privacy are crucial issues when developing information systems. On the one hand the users’ personal data must be protected and processed in a safe manner, on the other hand, the security and integrity of the content must be ensured. The AmbieSense agency adheres to security requirements imposed by European legislation (such as Directive 2002/58) as well as to general AmbieSense principles on Security and privacy. AmbieSense agents will use the security mechanisms implemented in the Context Middleware when accessing user context.

3.1.5 Encapsulating and integrating with CORPORUM™ Intelligent Agents

CORPORUM is the name of a set of tools developed by CognIT that enables high-precision content classification and retrieval based on a linguistic analysis and neuro-fuzzy methods, as described in [Engels 2001]. A major part of the CORPORUM functionality was developed within the OnToKnowledge project (IST1999-10132). The A-MAS integrates with CORPORUM, thus making the functionality available on mobile clients.

3.1.6 Integrating with the Content Integration Platform

The Content Integration Platform (CIP) is a set of functional packages, many of which are developed within the AmbieSense project in order to ease the interaction between Content Providers and the delivery mechanisms in AmbieSense. The A-MAS will facilitate packages for personalisation, content storage and information retrieval.

3.2 ORGANISATION OF THE A-MAS

The AmbieSense Multi Agent System can be seen as an electronic institution such as presented by e.g. [Rocha 2001]. The norms and rules of the institution are clearly defined by the concept of context and context templates. Whereas context templates define the valid attribute-value pairs that describe a situation, a context is the instantiation of such a structure. Interaction in the AmbieSense agency relies on the mutual understanding of context among the agents and a pre-defined protocol for communication: the FIPA Query protocol (more details in Chapter 5).

The following section visualises the tasks and co-operation of agents in the A-MAS; a detailed description of the interaction of A-MAS agents is provided in Chapter 5.

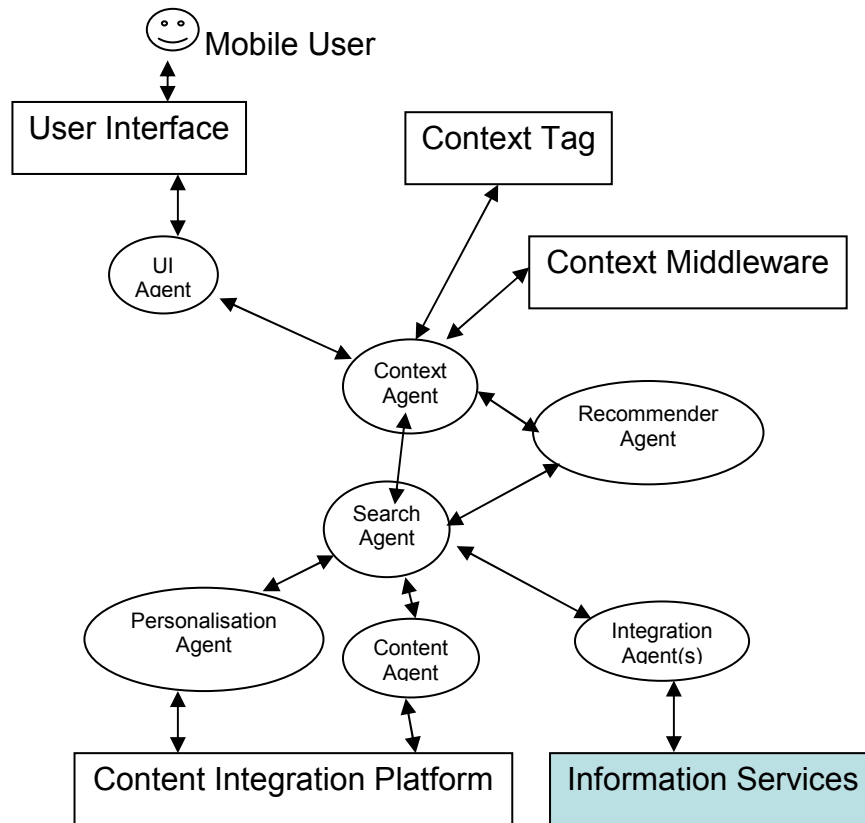


Figure 4: Agents in the A-MAS

The diagram above shows the organisation of the A-MAS. The context agent is the central entity in the agency. It gets contextual information from both Context Tag and User Interface (via the UI Agent) and encapsulates the Context Middleware with the User's context space. Upon context updates, it requests services from the search agent that uses the available retrieval agents in order to get content. Some of the retrieval agents (e.g. the content agent) may reside on the mobile device; others reside on remote content servers.

The detailed interaction of the agents will be described in Chapter 5. However, an example scenario is given in order to visualise the agent tasks:

Ole W. is on his way to the Oslo Airport from where he will leave heading to Rome in order to visit a conference on mobile computing. When arriving at the Oslo Airport (OSL), he turns on his AmbieSense-equipped PDA and chooses the Airport Information Service. The interface now suggests for Ole to provide his flight number in order to get access to value-added information services, to which Ole complies.

After having checked in, Ole has some spare time in the departure lounge. Not being very fond of Airline catering, he decides to get something to eat before he has to board his plane. His PDA already shows the location of the next salad bar. While eating, he checks the standings and highlights of the ongoing Cricket match between England and Pakistan that his PDA already has downloaded. His PDA also notifies him on some Whisky offers in the Tax-Free Shop that might be of interest to him. By the time, he finished his shopping, his PDA notifies Ole that it is time to move towards his flight gate.

The little scenario will be used in order to describe the tasks that are executed by agents. When Ole turns on the PDA and starts the Airport Service Application, a JADE container is launched, containing the A-MAS agents. As soon as Ole inserts his flight number, the User Interface Agent passes this information on to the Context Agent. Furthermore, the context agent receives information on Ole's current location from the Context Tags nearby. All the information about Ole's situation is merged with Ole's context. With the updated context, the Context Agent now requests the Search Agent to get appropriate content.

The Search agent uses the yellow-pages service in the agency in order to get an overview about what agents are available to support him. He then passes the query to the Personalisation and content agent that access the AmbieSense Content Integration Platform. Here an Info package (containing the sports content) is bundled and sent (pushed) to Ole's PDA.

In addition, the search agent requests the Integration Agents to search the OSL content for other information that might be interesting for Ole. The whisky offers in the Tax-Free shop match with Ole's interests, specified in his personal context. A URI to this information is being returned to the interface agent that prompts the Interface to notify Ole that this – potentially interesting – information is ready for download (pull). Other information coming from the OSL information database, such as the "Go to gate" notification, is being pushed directly.

Meanwhile, the recommender agent uses the case-based reasoning engine in order to provide advanced services. It finds out that it's been a while since Ole had been in a "food"-location and requests the search agent to find a place serving salad.

As described, AmbieSense aims at providing context-aware information to mobile users, with tourists as the main target group for the project. The application domains include personalised airport information services, news/infotainment services, travel guide services and a map service.

4 Security and Privacy in the A-MAS

AmbieSense aims at providing content to mobile users in a context-aware manner in order to provide user-friendly, better-tailored services. Capturing user contexts naturally involves the processing of personal data such as location data, personal preferences, travel information (flight number) etc. However, the processing of highly sensitive user data requires security measures to ensure that the user's fundamental rights to privacy will not be infringed upon.

The AmbieSense project pays great attention to security and privacy issues. The project adheres to both the according EC directives as well as to internal security policies. A thorough investigation of the security and privacy measures in AmbieSense is carried out in work package 5 of the project and will be documented in the forthcoming 6th AmbieSense deliverable: "System Architecture and Information Standards Report" (to be published in January 2004).

There are, however, issues concerning security and privacy that are specific to the A-MAS. The following sections will discuss some general security aspects in agent-based technology, followed by an analysis of the Directive 2002/58/EC of 12 July 2002, concerning the processing of personal data and the protection of privacy in the electronic communications sector (Directive on privacy and electronic communications).

4.1 SECURITY IN MULTI-AGENT SYSTEMS

4.1.1 Security in Message Transmission

When agents wish to exchange messages security must be handled on three levels: the physical, transport, and session layer. For any system not directly communicating with the physical layer, it must be assumed that the security for that layer is handled in a reasonable manner by the hardware implementing it.

For the transport layer the agency has two possibilities for communication, either internally or externally. Communication within an agency is hidden from the outside. JADE utilises either Java events or RMI. External communication across platforms and agencies in Jade, works over TCP/IP, either via the IIOP or HTTP. This implements the usual security mechanisms for IIOP and HTTP. There is further more a possibility for direct socket communication between platforms on different IP locations.

When dealing with the session layer it is entirely up to the particular agent or agency how security is handled. However, some basic functionality is implemented into Jade. JADE allows for serialisation and transportation of Java objects. This supports transmission of complex data structures between agents; thus allowing the programmer to implement any security mechanism for interpretation of serialised objects. The project will not further extend this, but instead focus on the core issues of the agent platform.

4.1.2 Protection of Content

When dealing with content from either the users or content providers, it is obvious that the agent containing or communicating the content has the responsibility of complying with the guidelines given in the *AmbieSense Security and Privacy Report*.

Since most of the external security is defined by the protection mechanisms applied on the transmission and transport layer, the major focus for the agents will be on internal security. JADE handles private messaging between agents, by storing messages in private cues for each agent.

4.1.3 Protection of Context

Context is the kind of information that has a high degree of confidentiality and a high degree of public interest. The agents handling the contextual information have to deal with this paradox in a flexible - yet private manner. The contextual information can primarily come from three suppliers: the user, the environment, and information suppliers. In the A-MAS, only the Context Agent has the

Username/password-protected access to the users context space. Communication of context will employ the Secure Socket Layer (SSL) protocol.

4.2 AGENT SECURITY AND EC POLICY

As part of the European Commission's 1999 Review of the communications framework, a draft proposal to update the existing [Telecoms Data Protection Directive \(97/66/EC\)](http://europa.eu.int/ISPO/infosoc/telecompolicy/en/9766en.pdf)⁸ was adopted on 12 July 2000, formerly known as the Communications Data Protection Directive (CDPD) but now known as the Directive on Privacy and Electronic Communications (DPEC). The overriding aim of the new Directive is to take account of technological changes and to make the provisions as technology-neutral as possible.

The DPEC, concerning the processing of personal data and the protection of privacy in the electronic communications sector sets the legislative framework for this issue in the AmbieSense project. In the following, we give a brief resume of the addressed issues relevant for the AmbieSense system. It should be noted that not all of the issues relevant to the AmbieSense System are relevant to the A-MAS. It should be made clear up front that the AmbieSense agency only processes data that either is in the user's context space already (i.e. the user has given consent for using the data) or contextual data that belongs to context-aware applications which the user has subscribed. The security settings for these applications are managed in the design of the content provision mechanisms, which belong to another work package in the AmbieSense project. Furthermore, the AmbieSense agency will facilitate the security mechanisms implemented in the other system components such as the context middleware.

However, in the following, we will account for how each of the issues may or may not affect the AmbieSense agency.

1. The provider of a public network must take appropriate technical and organisational measures to safeguard security of its services (exception: lawful interception) (Art. 4/1, Art. 5/1); the user must be informed of any special risks of a breach of the security of the network (Preamble /20).
2. The storage of personal data on the terminal equipment is restricted. The user must be informed on the purpose of storage and has the right to refuse (Art. 5/3).
3. Traffic data relating to subscribers and users processed and stored by the provider must be physically deleted after usage. (Art. 6/1 with restrictions of Art. 6/2-3, i.e. billing and commercial services).
4. Subscribers must have the right to receive non-itemized bills (Art. 7/1)
5. Calling Line Identification presentation may be restricted by the user (Art. 8)
6. Location data must be made anonymous unless accorded by the subscriber (Art.9.)
7. Subscriber shall be informed whether they are listed in a public directory (Art.12)
8. Unsolicited messages (e.g. direct marketing, etc.) should only be sent in accordance with the system subscriber. (Art. 13.)

⁸ <http://europa.eu.int/ISPO/infosoc/telecompolicy/en/9766en.pdf>

Issue 4 deals with billing, which is not within the AmbieSense agency's domain and therefore not applicable.

Issues 2,3, 5 and 6 deal with the processing and storage of user data. As AmbieSense aims at providing context-aware services (or so-called value added services as in Preamble/35) it is necessary to store and process user specific data. The user's desired level of privacy however, is being set at UI level for each application and just forwarded to the agency.

The AmbieSense agency is a specific instance for each user, there are no public user directories in the agent scenario and thus issue 7 is not applicable.

However, issue 1 must be paid attention to. AmbieSense agents reason on and communicate context and thus user specific data – even though it is anonymised. That means that measures must be taken to ensure the security of the content of agent messaging.

4.3 CONCLUSION ON AGENT SECURITY

In FIPA compliant agent systems according to [Houmb 2002] there are the following security issues concerning message integrity:

- Modification of message contents
- Masquerade
- Fabrication
- Replay

Modification of message contents is an attack where the attacker alters some or all of the content in a message when being transported over the network, or stored on a storage device. Masquerade is an attack where the attacker acts on the behalf of the original sender or receiver. Fabrication is an attack where the attacker forges a message. Replay means sending the same transmission dialogue at a later time in an attempt to disrupt the synchronization or integrity of the agent framework.

In order to meet the requirements stated in Art. 1 of the DPEC, the AmbieSense System Architecture and Information Standards Report suggests that the following security measures will be implemented in the A-MAS:

- Each agent has an ID
- User and user agents will be handled separately in access right (although the JADE platform is not used for inference mechanisms)
- Application agents will get an X.509 certificate
- Application agents belong to a share group.

With regards to issue 8, agents will use filtering mechanisms described in Chapter 5 when interacting with content provision components in AmbieSense and thus filter out unsolicited messages unless the user has given her acceptance.

In addition, the AmbieSense Project will adopt the Security mechanisms provided by the JADE-S security add-on. The JADE-S platform, which is constantly developed further, provides mechanisms for authentication by using certificates, and authorization. Furthermore, the JADE-S add-on uses Java support for principal-based authentication. Permissions are assigned not only to pieces of code but also to who executes that code.

Finally, JADE-S enables secure communication through usage of the Secure Socket Layer (SSL) protocol, providing privacy and integrity for all intra-platform communication.

5 Architecture

5.1 AGENTS IN THE AMBIESENSE ARCHITECTURE

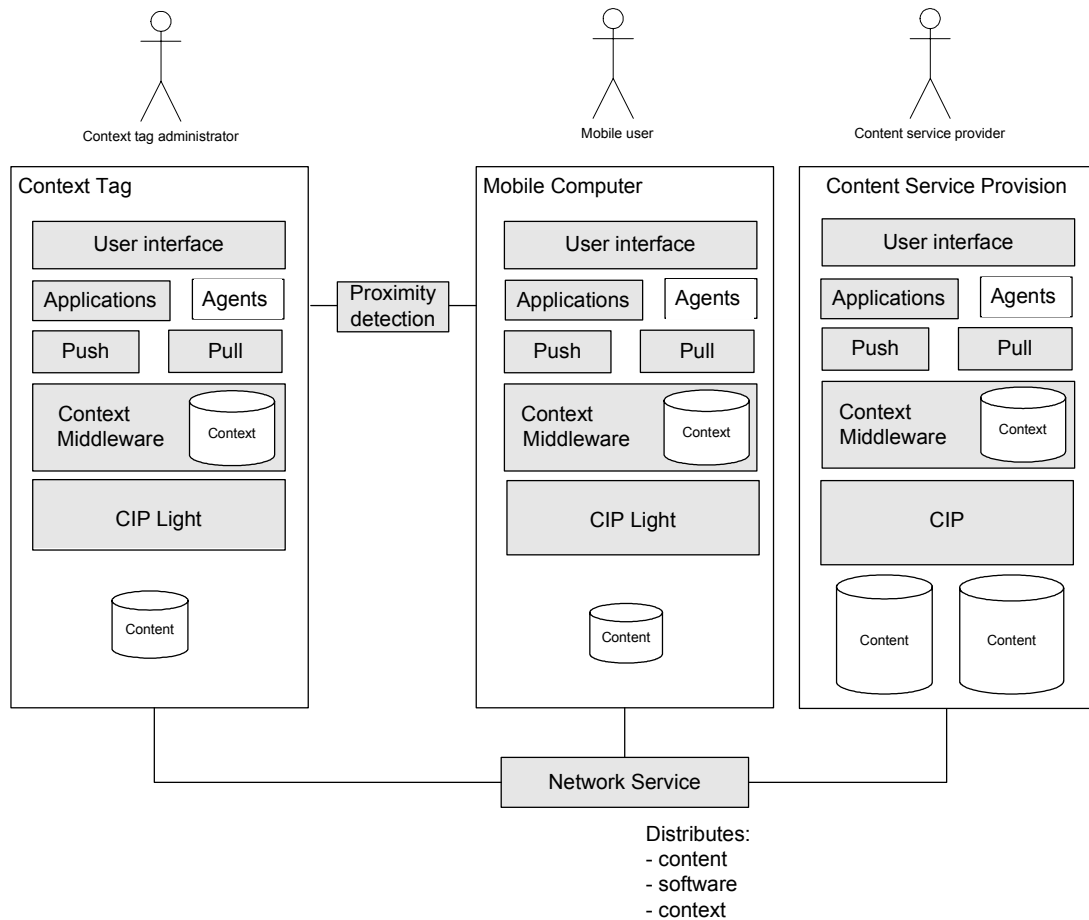


Figure 5: AmbieSense Logical Software Architecture

Figure 5 depicts the logical software architecture as presented in the AmbieSense Reference Information Model. The A-MAS is designed to be able to reside on all three cornerstone platforms of the AmbieSense systems:

- JADE/LEAP agents can run on the mobile device as well as in containers on the Content Service Provision.
- Containers with Context Agents run on the context tags.
- The parts of the A-MAS that require heavier processing (e.g. CORPORUM agents, or the CREEK CBR engine) will run on the servers of the Content Service Provision.

5.2 THE FIPA REFERENCE ARCHITECTURE

The Foundation for Intelligent Physical Agents (FIPA) works on agent interoperability via standardised agent communication and content language. Beside the generic communication framework, FIPA is also specifying ontology and negotiation protocols to support interoperability in specific areas such as travel assistance, manufacturing, etc. Figure 6 presents an overview of the architecture of the FIPA platform.

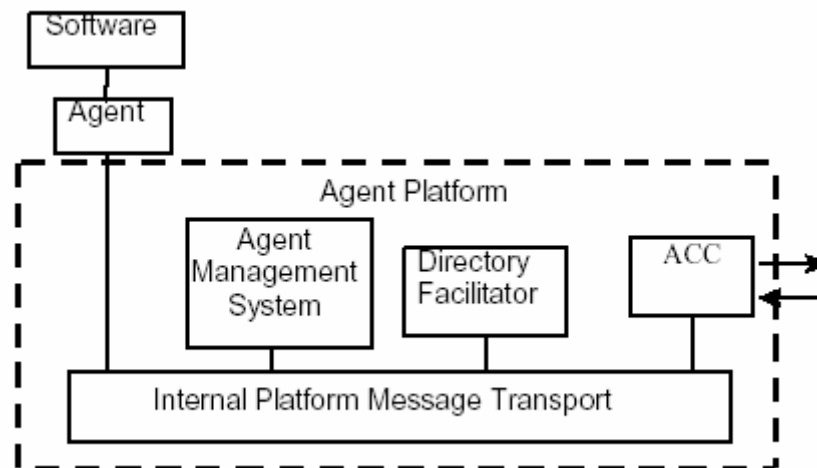


Figure 6: FIPA reference architecture

This reference model identifies roles of some key agents necessary to manage an agent platform. The Agent Management System (AMS) is the agent that exerts control over access to and use of the agent platform. This includes: maintaining a directory of agents and handling their life cycles. The agent communication channel (ACC) ensures basic contact between agents inside and outside the platform. The directory facilitator (DF) is the agent that provides yellow pages services to the agent platform.

As JADE/LEAP is FIPA compliant, it implements the AMS, ACC and DF. This will allow us to set up agencies on the mobile computer, context tag and content platform.

It seems like FIPA is having some difficulties after the burst of the “Internet bubble”; yet the FIPA compliant JADE/LEAP platform gains more and more support. Recently Motorola became a member of the JADE board, showing that large corporations are interested in this technology. We are basing our technology on JADE/LEAP, which is Open Source meaning that, in case something goes terribly wrong with either the FIPA standardisation or the JADE/LEAP platform, we always have to possibility to continue our work.

In the AUML sequence diagrams we use the FIPA Query Interaction Protocol “query_ref”, which has four different answer possibilities “not_understood”, “refuse”, “failure” and “inform”. In the sequence diagrams however, we only visualise the “inform” answer, as this is the wished behaviour in the described context. If malformed messages arrive, the receiving agent will not reply with an “inform”, yet with the appropriate behaviour that suits the malformed message. See the figure below for an AUML overview of the FIPA Query Interaction Protocol.

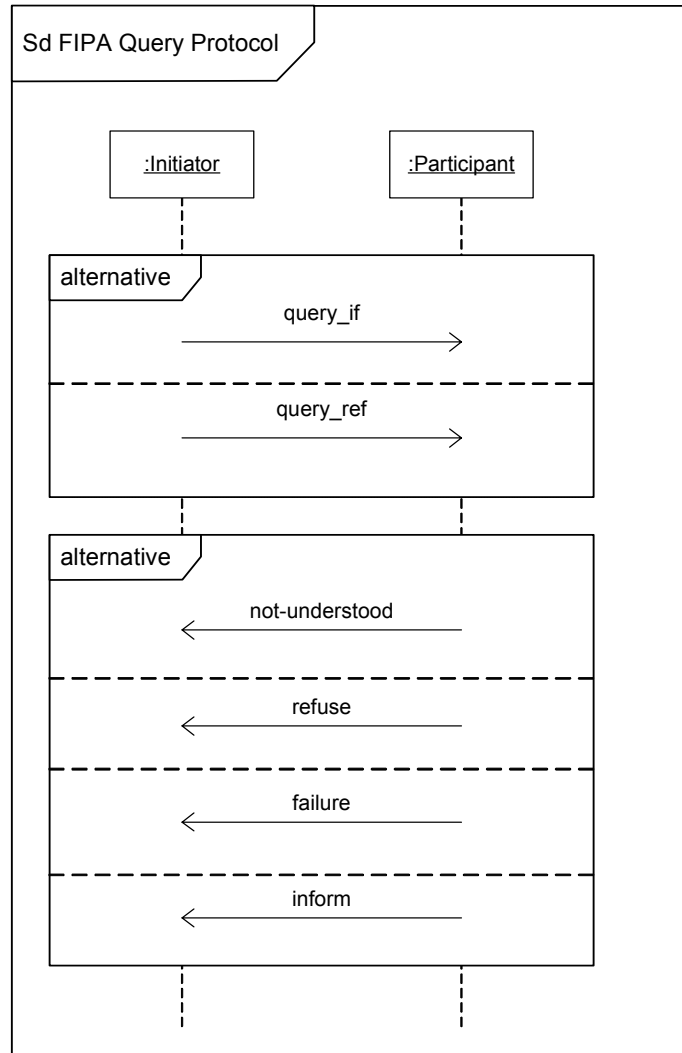


Figure 7: AUML representation of the FIPA Query Interaction Protocol

5.3 AGENT TYPES - FUNCTIONALITY

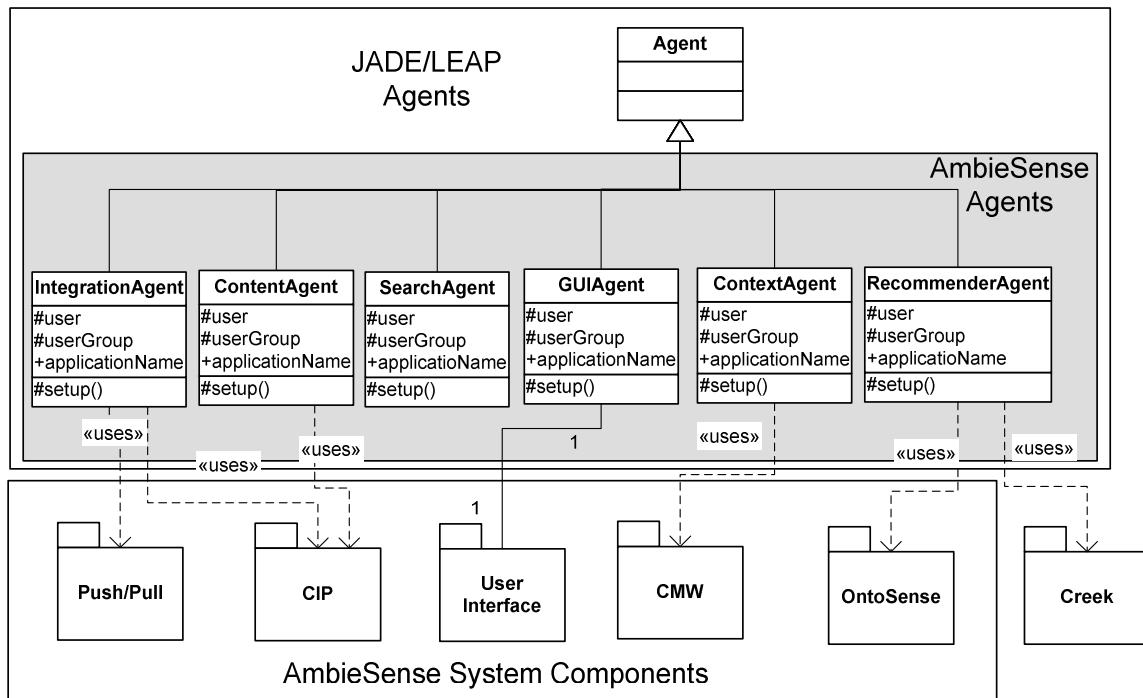


Figure 8: The A-MAS architecture (class diagram)

Figure 8 depicts the components of the A-MAS. As shown in the diagram, the JADE/LEAP environment is the framework for the AmbieSense specific agents, thus providing the basic infrastructure according to FIPA. The entities necessary in the FIPA architecture such as the Directory Facilitator, Agent Management System etc. are not explicitly depicted in the diagram but can be assumed as they are part of the JADE framework.

The AmbieSense-specific architecture consists of two layers: AmbieSense Components, some of which reside on the mobile device, others reside on servers. The other layer consists of agent classes – extensions of the JADE/LEAP agent class. These agents reside on the mobile device and the Context Tag in order to enable the quick and secure processing of the user context. All AmbieSense Agents are extensions of the jade.core.agent class and inherit JADE's FIPA compliance, thus communicating with each other in ACL via the different message transmission protocols (IIOP, HTTP). In addition they interact with other AmbieSense components such as the CIP or the Context Middleware.

Agents in the A-MAS fulfil the description of agent properties in various degrees. Some of them e.g. the Context Agent act autonomously, whereas others (Integration Agents) work in a rather reactive manner.

The following paragraphs provide a brief account of the different agents, before each component in the A-MAS will be described in technical detail. However, the entirety of the AmbieSense agency fulfils the criteria of situatedness and goal-orientedness as they have to achieve their goal (the provision of relevant content) in whatever context they are put into.

5.3.1 Context Agent

The Context Agent is the heart of the A-MAS. It encapsulates the Context Middleware and administers the access to the user's context space, ensuring privacy and security. The Context Agent updates and maintains the user context and triggers the queries for content conducted by the search agent. Furthermore, it informs the Interface Agent about available content and provides the Recommender Agent with the current context if necessary. In requesting services from the search agents and making the user context available to other agents, the context agents will be the most autonomous entity in the A-MAS.

5.3.2 Personalisation Agent

The Personalisation Agent ensures that the content presented to the user is not only relevant to the user's context, but also personalised according to the user's or to a user group's needs and preferences. The Personalisation Agent incorporates the personalisation modules of the CIP. The Personalisation agent needs an integration with the Personalisation Module of the CIP.

5.3.3 Recommender Agents

The Recommender Agent uses contextual information and employs reasoning techniques like e.g. case-based reasoning for an analysis of the users' situation in order to provide appropriate content.

5.3.4 Content Agent

The Content Agent provides access to the AmbieSense Content Integration Platform with advanced content storage and retrieval capabilities.

5.3.5 Interface Agent (UIAgent)

Interface Agents interact with the UI components. They react to user input that influences the context, and they notify the UI if other agents find new available content.

5.3.6 Search Agent

The Search Agent receives an information request from the Context Agent in the form of a user context. The Search Agent will forward this context to one or more specific Integration Agents that will convert the context to a proprietary query fit for the respective IE and content modules. The Search Agent keeps track of the available search types/modules and will always forward the context to each of these modules.

5.3.7 Integration Agents

Whereas the Content Agent can communicate directly with AmbieSense specific components, other applications or content providers will need a smooth integration with the A-MAS without having to program JADE agents. The Integration Agents are wrapper classes that handle the interaction between JADE/LEAP agents and any generic JAVA environment by providing the API to the AmbieSense agency. Two examples for the Integration with non AmbieSense applications are given: The Corporum integration as well as the integration with the YellowMap map service.

5.4 AGENT TYPES – IMPLEMENTATION

5.4.1 Context Agent

5.4.1.1 Overall Functionality

The Context Agent is responsible for updating the user context, maintaining the context space and request services from other available agents. In order to clarify these non-trivial tasks, it is important to revisit the concept of context in the AmbieSense framework.

The AmbieSense Reference Information Model [AmbieSense 2003] defines context as “a description of aspects of a situation”. In AmbieSense, this description consists of five context categories that capture the situations of mobile users:

- Social Context
- Task Context
- Personal Context
- Environmental Context
- Spatio-Temporal Context

Quoting the AmbieSense Reference Information Model:

“A context describes aspects of a situation seen from a particular actor's point of view. An actor can in the widest sense for instance be a person, thing, matter, or organism. In this way context is actually defined as something separate from the situation it self. A context in AmbieSense is a representation inside the computer. It represents aspects of a situation in the real world.(...) AmbieSense chose to implement this structure by developing Java-classes integrated with XML technology. “

In summary, context in AmbieSense is represented as a structured set of attribute-value pairs that capture relevant aspects of a user situation, where the term relevance depends on the service application domain (to a map service, the user location will be more important than to a news service).

The Context Agent’s task is to update and maintain this structure for its user. That includes the contact with the system components that can be the source for context changes; mainly the context tag and the user interface. The former can provide contextual information regarding the environmental or the spatio-temporal context; the latter can provide information on user preferences that will be stored as part of the personal context.

Furthermore, the Context Agent is the gatekeeper to the user’s context space, i.e. the Context Agent is the only entity in the AmbieSense agency that has the username/password to a specific user’s instance of the context space. The updated user context is stored in the user’s context space.

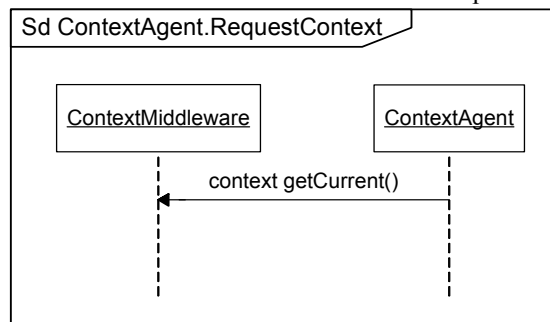


Figure 9: AUML Sequence Diagram Context Agent Request Behaviour

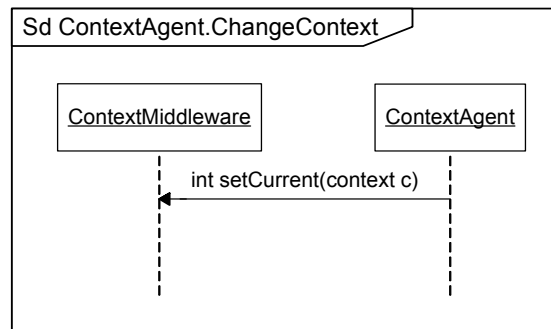


Figure 10: AUML Sequence Diagram Context Agent Change Behaviour

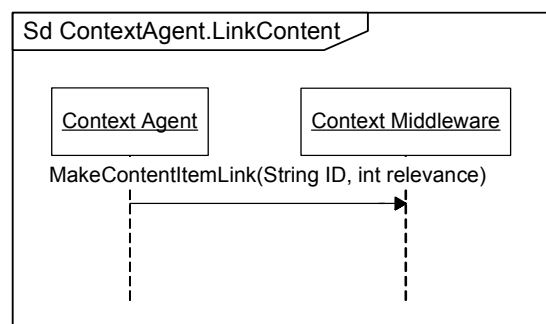


Figure 11: AUML Sequence Diagram Context Agent Link Content Behaviour

5.4.2 Search Agent

5.4.2.1 Overall Functionality

The Search Agent has the ability to search different kinds of content bases by controlling several different Agents which each has its specific task. When an information request arises (in the Context Agent), the Search Agent creates the specific Agents and forwards the information request. Each specific Agent will then distribute this information to proprietary search mechanisms that lay behind. Each of the proprietary systems gives its result to the specific Agent, which in its turn forwards this to the Search Agent. The Search Agent contacts the Context Agent to inform new information has been gathered.

5.4.2.2 Detailed Description

The Search Agent (SA) is defined to only respond to requests from Context Agent (CA). The CA sends a context that reflects an information need. SA will check an internal list that defines the available Integration Agents (IA). In case one or more of these IA are unavailable, the SA will launch a new IA, otherwise it will forward the context to the IA. See paragraph 5.3.7 for a description on how the IAs handle these requests.

At the moment, three (or more as this part is expandable) specific wrapper agents exist that will convert the context to a proprietary query fit for the search module.

Upon receiving the request from the CA, a timer starts, if the timeout value is reached before the search of one of the IA is completed, this agent (or these agents) will be terminated. In case no timeout occurs, the IA will send an URL to the found content to the SA, which forwards it to the CA. Note that the URL may point to an empty resource, meaning there were no search results.

The implementation platform and underlying operating system is transparent. The modules exchange messages and data through a set of predefined JADE services that are provided by the main Search Agent. All components can run on different machines and on different sites.

5.4.2.3 Sequence Diagram

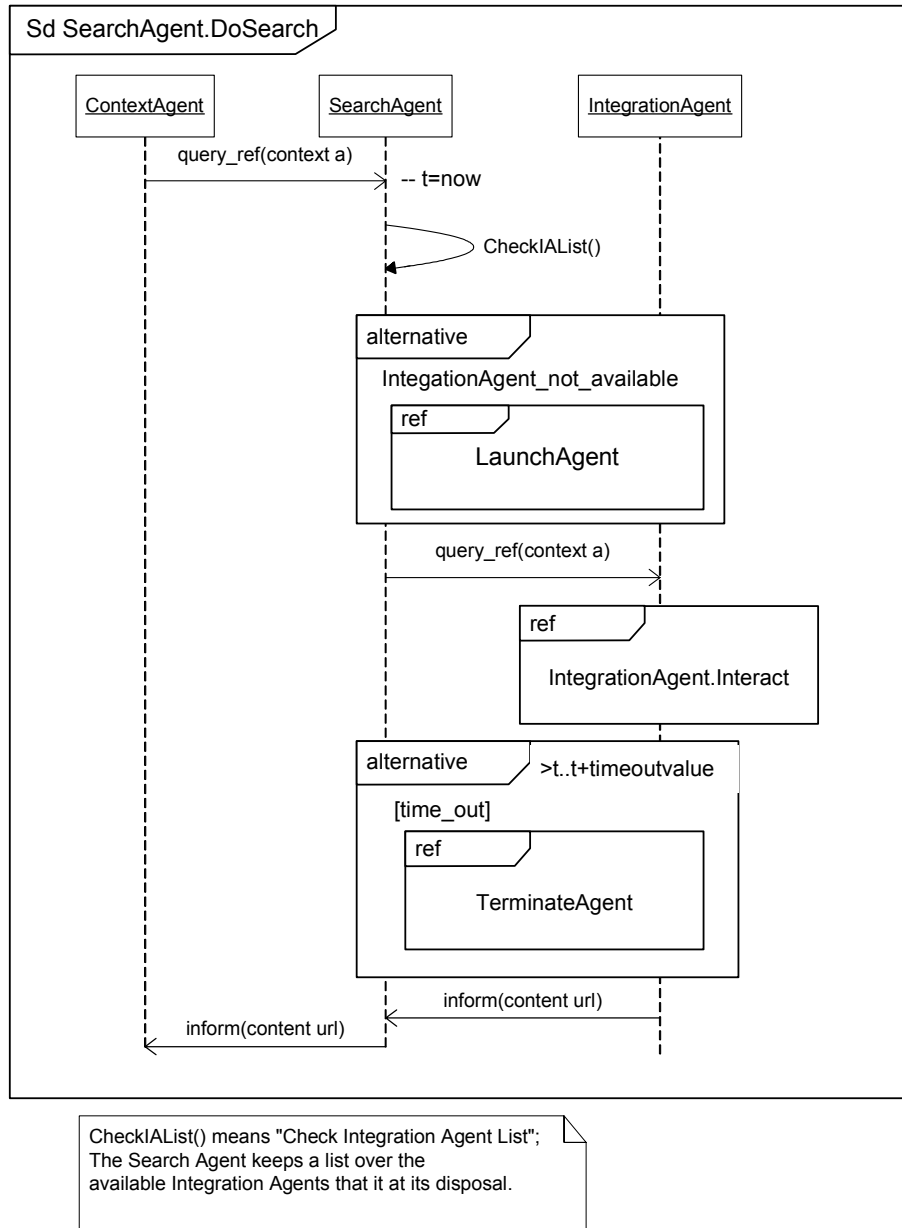


FIGURE 12: SEQUENCE DIAGRAM SEARCHAGENT.DOSEARCH

5.4.3 User Interface Agent

5.4.3.1 Overall Functionality

The User Interface (UI) Agent can receive information from both the UI on the mobile device as well as from the Context Agent. When it receives an information request or a change from the UI, it will forward this request to the Context Agent, which in turn has the task to decide what to do with this request or change. The same is valid the other way around: when the Context Agent has found information that could be of interest for the user, or if some information is to be updated, it will notify the UI agent which then can update the UI on the mobile device. Depending on the user's feedback (or lack of it for that matter), the UI Agent can take further action. However, when the user is subscribed to a certain service, he/she will automatically receive the information without further approval.

5.4.3.2 Detailed Description

If the Context Agent finds something that may be of interest for the user, the UI Agent is notified by a message containing a description and an URI to this information. The UI Agent will forward the description to the UI on the mobile device and if the user accepts to receive this information, a confirmation message is sent to the UI Agent, which in its turn will fetch the information from the URI specified earlier and send this to the UI.

This functionality also works the other way around: when a user wishes to receive information about a certain topic, it can send a request to the UI Agent, which will forward this to the Context Agent, which in its turn decides what to do.

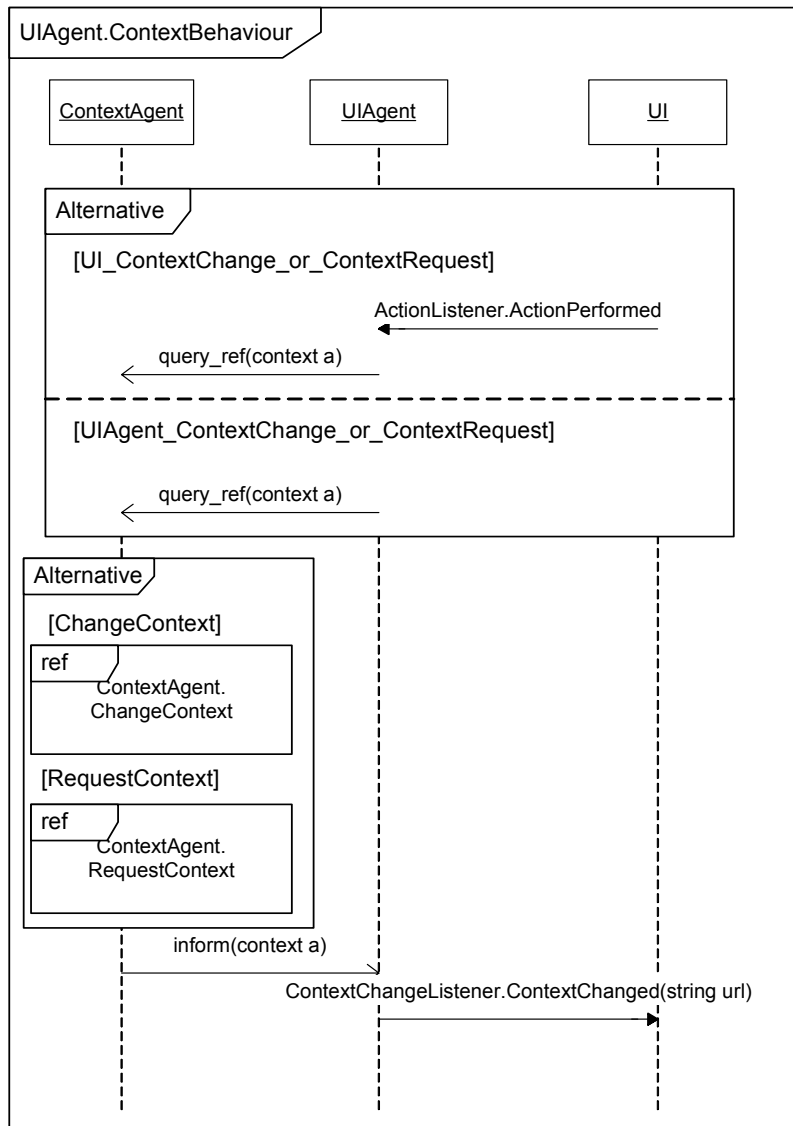


Figure 13: AUML Sequence Diagram User Interface Agent Behaviour

5.4.4 Personalisation Agent

5.4.4.1 Overall Functionality

The special (high level) tasks of the personaliation agent are:

- Holding user data and managing it with the help of the Usermanagement from the CIP API
- Building of personalisation models and associating it dynamically to users
- Gathering of user context and interfacing it with the personalisation models
- Triggering the personalisation process and communicating its results to other agents (i.e. User Interface Agent).

Innovation

The innovation clearly lies within the agents and not in the the surrounding agent framework. This is because the agent framework is instantiated and used but not extendend.

The inner-agent innovation has two angles:

- 1) Firstly, innovation is delivered by the way the agent uses the analysis package. This means mainly the way how the personalisation workflow models are composed. Beside that, it also includes the way the agent put these models together with other sources of information to make them work. This includes the connection of personalisation models with users, execution of personalisation models for users, gathering of user context and connection with the personalisation model.
- 2) Secondly, the personalisation agent is able to introduce additional, innovative own information processing techniques (preferably from the field of information retrieval or collaborative filtering) to its own personalisation models. This allows to extend the existing analysis package with own techniques and methods.

5.4.4.2 Detailed Description

The personalisation agent uses the personalisation framework from the CIP analysis package. The API provides a generic infrastructure which is instantiated by this agent. The connection with the CIP Usermanagement allows to handle users and usergroups. The agent holds an instance of the PersonalisationModelManager allowing it to connect users with personalisation models. Beside the generic infrastructure, the CIP Analysis package provides some standard implementations, which can directly be used by this agent.

5.4.5 Content Agent

The Content Agent delivers content to mobile users in AmbieSense linked to the user's context in a meaningful way. When the context of a user changes, be it location, be it preferences, the need for content changes as well. The Content Agent receives an update message from the Context Agent each time a change occurs.

At the airport, for example, tax-free offers have no value if you are no travelling abroad since you will not be allowed to buy these items. At the airport, the current location of a user plays a very important role, as there are two 'points-of-no-return', the security check and the document control that has to be passed when entering the area for international departures. Once one of these point is passed, the traveller is not allowed back to the previous area, meaning that all services before that point are now irrelevant as the user is not able to access these areas.

The Content Agent uses CORPORUM technology, which is the name of a set of tools developed by CognIT that enables high-precision content classification and retrieval based on a linguistic analysis and neuro-fuzzy methods, as described in [Engels 2001]. A major part of the CORPORUM functionality was developed within the OnToKnowledge project (IST1999-10132). The Content Agent uses this technology to retrieve relevant information, based on the users' preferences, from a pre-indexed source such as the shopping area web site of OSL.

5.4.6 Recommender Agent

5.4.6.1 Overall Functionality

The recommender agent is one of the core technology agents. It assess the user's situation, and uses Case-Based reasoning (CBR) to reason about which actions to take, based on know situations (cases) and general domain knowledge.

5.4.6.2 Detailed Description

The reasoning mechanism is split into two different parts: The on-line part that resides on the user's mobile device, which maintains an on-line case-base, and communicates with the other agents in the agency to facilitate the recommendation through task-decomposition. The off-line reasoning resides on the user's backbone system, and maintains a more complex history of the users' situations.

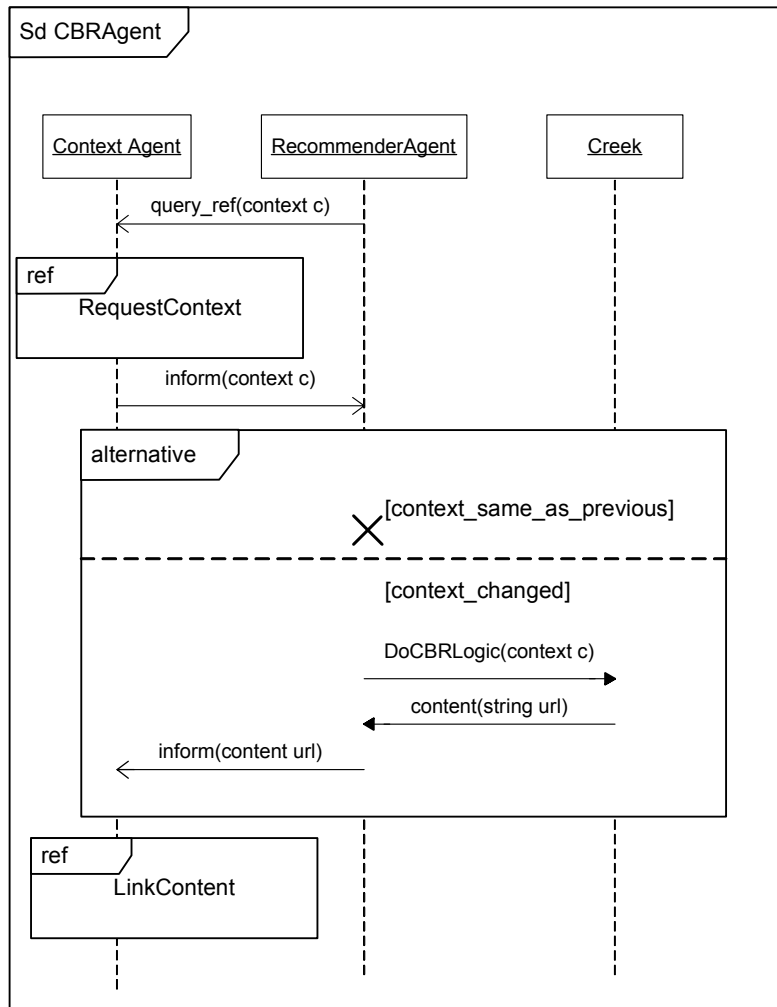


Figure 14: AUML Interaction Diagram Recommender Agent

5.4.6.2.1 ON-LINE REASONING

Through contact with the Context agent, the recommender agent maintains a dynamic structure of the contextual information available.

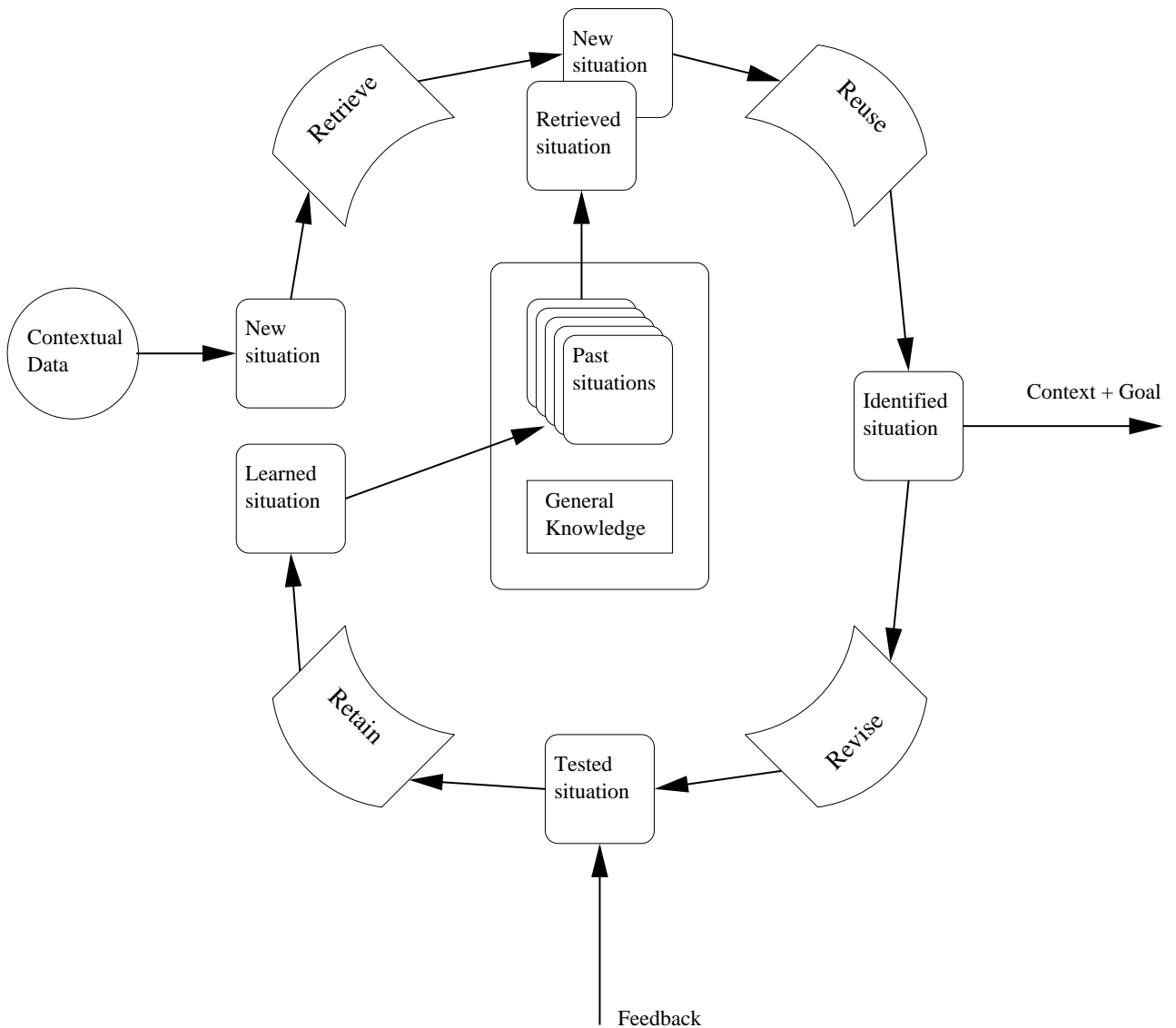


Figure 15: Identifying Situations with Case Based Reasoning

Different types of contextual information can arrive in a very diffuse fashion, e.g. time is continually flowing into the system, whereas location might be pseudo static. Since CBR works on discreet cases the continuously values flowing into the system must be made discreet. The recommender agent handles this.

Following the suggestion of [Zimmermann 2003], the agent takes snapshots of the contextual information at certain time intervals, i.e. the state of the context structure, and stores them as cases. Once a snapshot has been taken, the CBR cycle is activated (see Figure 15). The system will try to retrieve a known context or case, and classify the current situation based on the retrieved one.

When the situation has been classified, the associated goal is decomposed into tasks manageable by the other agents and services accessible to the agency. When the goal has been reached the case will be stored in the case base in a triplet consisting of: i) the contextual information describing the situation, ii) the problem associated with the situation, and iii) the solution constructed by the application agents.

Since the user is expected to experience at least a few different situations a day, the storage of the cases will quickly fill up the mobile device with cases. This potential vast amount of cases will also severely hamper the searching process for the CBR mechanism. To remedy this, some of the reasoning process is moved into the user's backbone servers.

5.4.6.2.2 OFF-LINE REASONING

There are potentially two main problems with the use of Case-Based Reasoning for identifying situations; the storage problem, and the problem of indexing and searching.

First and foremost is the problem of storing the, potentially vast, amount of cases constructed during run time. To solve this, the user's personal persistence storage is available on the user's home network. This storage will be used for storing the cases, and will be synchronised when the user has an up-link.

This large amount of data does not only affect the amount of storage space needed, it will also severely affect the indexing and matching algorithm used. To remedy this a generalisation process will occur on the home net. Similar cases will be grouped into prototypical ones, e.g. everything that the 600 business meetings have in common will constitute the prototypical business meeting situation. These prototypical cases will be part of the on-line case base, and be used in the every day reasoning process. Generalisation of cases is a well-known research area within CBR, for an overview, refer to e.g. [Zimmermann 2003]. This case base is structured by prototypes, which are generated on the basis of the amount of similar parameters in the point-cases.

5.4.7 AmbieSense Ontology

5.4.7.1 Overall Functionality

To allow for any reasoning within the AmbieSense framework a common agreement on the semiotics used are required. The most prominent reasoning agent in the AmbieSense agency is the Recommender Agent. This agent utilises both CBR and the general domain knowledge to recommend which actions should be taken in particular situations.

As mentioned in the beginning of the chapter, FIPA provides some ontologies for interoperability of applications. However, no useful ontology representing user context could be found. Therefore, AmbieSense introduces a user context ontology based on the context model proposed by [Myrhaug 2003]: The general knowledge and contextual model in AmbieSense is encapsulated in OntoSense, the ontology for the AmbieSense domain.

The ontology is structured as a tree, with three layers, where a common frame of reference constitutes the root, the AmbieSense specific concepts are defined in the middle layer, and the top, or leafs, is specific for each of the specific sub-domains. As an example: the restaurants at Oslo Airport could each build their own ontology concerning what they serve. This ontology could be attached to the AmbieSense ontology, which primarily defines the contextual concepts [see Figure 16].

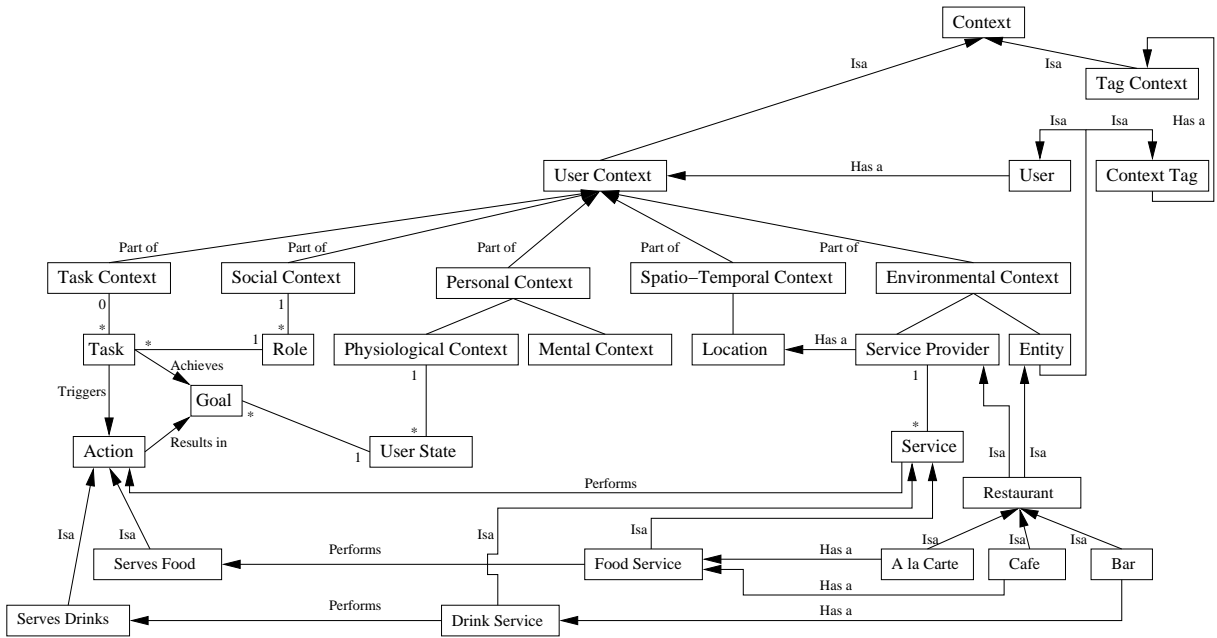


Figure 16: The AmbieSense Ontology, OSL scenario

5.4.8 Integration Agents

Integration Agents are the bridge between the A-MAS and proprietary systems. The Integration Agent handles the two following tasks:

- They provide the interface between the JADE/LEAP agent framework and generic JAVA environments. They come with a clearly defined API, enabling application developers to have their systems interact smoothly with the A-MAS
- They encapsulate the functionality of the ambiesense.context package in order to parse context represented in XML into byte efficient context objects.

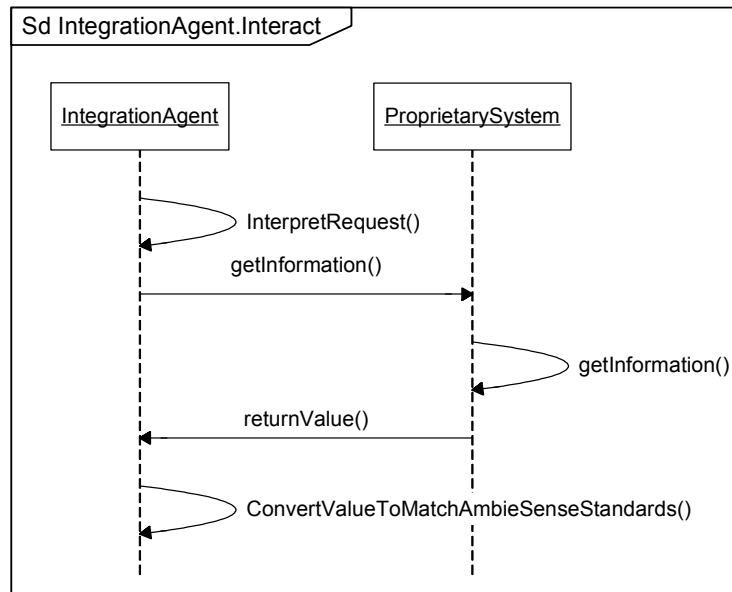


Figure 17: Interaction of the Integration Agent with a proprietary system

5.4.9 Corporum Integration

5.4.9.1 Overall Functionality

One of these wrapper agents is the CORPORAUMWrapperAgent, whose task it is to create the bridge between CORPORAUM technology and AmbieSense Multi Agent Platform.

The Content Wrapper Agent interprets the Context provided by the Search Agent and converts it to a query or multiple queries that suit the underlying proprietary interface. It then awaits the results and sends this to the Search Agent.

5.4.9.1.1 DETAILED DESCRIPTION

CORPORAUM™ is a generic name for a group of products supplied by CognIT a.s, helping to increase access to and the sorting of relevant information. CORPORAUM™ uses advanced language and knowledge technology developed in Norway. This technology is based on the semantic analysis of text and cognitive models. For supplying the content to the AmbieSense platform, CORPORAUM™ Knowledge Server (KS) is used. KS can be seen as a “librarian” system. This metaphor alludes to the fact that the system knows what users need and keeps track of what is stored in various archives and directory structures. This information can be indexed with reference to the specific interests of a group or an individual. The system sets up a personal or group-specific archive of what is most topical or relevant to the users in question. The content of these is displayed in line with specified user, role or job profiles.

5.4.10 Integration of the YellowMap Service

In order to make the integration of the YellowMap's service possible, a bridge between the .NET world and the JADE agent framework needed to be made, which was not yet been achieved sufficiently by the JADE community. That means programming web services easily, the possibility to optimise the speed of the application, and the efficiency of the .NET framework could be incorporated in AmbieSense. On the other hand, since .NET is important in commercial environments, it would allow also other Content Service Providers with .NET technology to be integrated easily in AmbieSense.

Five options have been considered in order to make that possible.

	Deployment Complexity	Java Objects and Java Threading	Licences are needed	Source Code is needed
A -Microsoft Java # compiler	Low	Yes (2)	No	Yes
B - Microsoft Convert for binary classes	Medium (3)	Yes(2)	No	No
C -Webservices with Apache Tomcat and Axis plugin.	Medium	Yes	No	No
D -CORBA with Janeva from Borland	High	Yes	Yes (4)	No
E - Microsoft JLCA 2.0	Very High	No (1)	No	Yes

- (1) **Java Language Conversion Assistance** We would have C# source code for the Jade platform. However that implies a relative long development time.
- (2) **Over .NET framework** That means the threading and the serialization are capabilities from the Microsoft Java implementation.
- (3) **It is not always possible**
- (4) It is possible the implementation with another ORB for example OmniORB (<http://www.uk.research.att.com/omniORB/omniORB.html>). That will avoid the need for a license.

Any of the 5 options can be used in order to use the Java API's in the .NET Server. Therefore it would be possible to use the Context Middleware, the Content Integration Platform (CIP) and the AMAS system in .NET technology. In regard to A-MAS and Jade, Jade is actually the only agent platform that works with .NET using the option four, the Microsoft Java # compiler. It is possible to use Jade/Leap 2.5 over .NET since July of 2002. That possibility was provided by Steffen Rusitchka from Siemens, more information on link 1. However it was not possible to use the latest version of Jade/Leap 3.0b1 with this add-on. We found out that one patch from Microsoft is needed (link 3) in order to run the new version of Jade in the .NET framework. A detailed tutorial about how to compile Jade over .NET has been written (link 2) and included on the Jade Web page. Actually it is possible to use Jade over .NET framework. It is also possible since December 2 2003 (link 4) to run Jade/Leap on the .NET compact framework which fits good in one of the mobile target platforms of AmbieSense, Pocket PC.

Finally the possibility to run Jade over .NET opens a new option to integrate Java API's and .NET API's. It is now possible to design two interactive agents one written in Java and the other in .NET.

Links:

- (1) <http://sharon.cselt.it/pipermail/jade-news/2002/000006.html>

- (2) Yellowmap Tutorial (<http://jade.cselt.it/FAQ.htm#q21>)
- (3) Microsoft Patch (<http://www.microsoft.com/downloads/details.aspx?FamilyID=789dac0d-2ccf-4021-a943-a1c4802e1568&DisplayLang=en>)
- (4) <http://jade.cselt.it/jade-develop-archive/1122.html>

5.4.10.1 Overall Functionality

The functionality of the YellowMap integration platform is divided into two parts. First the Context Middleware functionality (Context, ContextTemplate, ContextValidator, ContextSpace) and second the different services available MapRequest, GeocoderRequest and RouteRequest. These two functionalities will be programmed using agents. That means we have one agent, the IntegrationAgentYellowMap which will receive the context from the AMAS system using Context Middleware functionalities to store and check a user context. At the same time the agent will decide which services should be called. The IntegrationAgentYellowMap will call the ProviderAdapterYellowMap with the right parameters in regard to the context. The IntegrationAgentYellowMap will get content items from the provider adapter and then build an InfoPackage.

6 Summary

In this report, we have documented the architecture of the AmbieSense Multi-Agent System. The system creates an innovative infrastructure that supports the AmbieSense overall system in distributing relevant content to mobile users.

After giving a critical overview of agent-based technology in general, we have shown the reasons for using a multi-agent system in AmbieSense. We pointed out that in the domain of information systems, agent-based technology is well-proven and widely accepted. The distributed architecture of the AmbieSense system in connection with the complexity of providing context-aware information services made agent-based technology a natural choice in AmbieSense.

We displayed the computational view to the A-MAS architecture by describing the enabling technologies that are being combined on top of the JADE/LEAP agent framework. In the following we gave a detailed functional and technical account of the components implemented in the AmbieSense Multi-Agent System.

The proposed infrastructure is designed for use in the AmbieSense system, together with other AmbieSense system components but can also easily be integrated with any enterprise system or content provision platform by means of integration agents. The provided architecture report gives examples for interaction with other AmbieSense System Components as well as with non-AmbieSense applications:

The Content Agent and the Personalisation Agent are examples for accessing innovative user management and information retrieval components through a multi-agent platform. Integration agents are illustrated by YellowMap and CORPORUM integration examples.

7 Appendices

APPENDIX 1: SURVEY – MOTIVATION FOR USING JADE/LEAP

In chapter 3 we described the high-level properties and functionalities of agencies. However, to make agents interact smoothly with each other, one needs a common platform for communication and protocol. This protocol for interaction can be provided by agent frameworks. Agent frameworks are programmable platforms that provide the functionality to handle the low-level (i.e. less human-like) tasks of the agency. Among these functionalities are the administration of agent life cycles, the handling of inter- and intra platform communication and the management of the services in the agency.

Implementing such a framework from scratch would be rather uneconomical since a number of programmable agent frameworks are readily available. Therefore, it is a reasonable and risk-reducing measure to select and adopt a suitable framework for AmbieSense. The following sections describe the evaluation process of suitable frameworks leading to the choice of JADE/LEAP for the A-MAS.

Requirements for Agents in AmbieSense

In order to evaluate agent frameworks, a list of requirements was set up. Some of the requirements were derived from the AmbieSense Requirements report (AmbieSense 2002), other requirements – those specific to agent functionality – were added before the evaluation process. These agent-specific requirements will be added to or, if possible, merged with the former ones.

Requirements from the AmbieSense Requirements Report

The AmbieSense Requirements Report (AmbieSense 2002) lists up 131 functional and non-functional requirements for the AmbieSense system. Obviously, not all of them are relevant for the agent-based parts of the system. Table 1 gives an overview which system requirements could be derived into relevant agent requirements (AR) for the A-MAS.

Agent Framework Requirements		
Req. Nr	Requirement description	Trace
AR1	Multi-agent functionality	R6, R54, R81, R98, R42
AR2	Programmable agent development platform	R6, R32, R17, R36, R42, R54, R56, R59, R76, R80, R81, R98
AR3	Integration with AmbieSense components	R24, R54, R98
AR4	Infrastructure	
AR4.1	Service discovery	R4, R17, R40, R54, R81
AR4.2	Service management	R8, R42, R54, R76, R98
AR4.3	Security/privacy	R34,

Table 1: Agent Framework Requirements (from Requirements Report)

In Table 1 the agent requirements are derived and generalised requirements from the actual system requirements (R) from the AmbieSense Requirements Report. The actual requirements are available in AmbieSense (2002). In the following, we just give a summary of them.

AR1: Multi-agent functionality: The requirements listed here demand that the agent-based components in AmbieSense should support multiple agents that implement a variety of different behaviours.

AR2: Programmable agent development platform: These requirements involve the various intelligent behaviours that the agents in the AmbieSense agency have to be capable of. The framework must therefore provide a platform that can be extended without any functional limitations.

AR3: Integration with AmbieSense Components: The requirements listed here concern the interaction of the AmbieSense agency and other AmbieSense system components.

AR4: Infrastructure: These requirements concern the discovery and managements of the agents within the A-MAS. These functions should be handled smoothly by the framework. In addition, the framework must allow for implementing security/privacy functionality.

Agent-specific Requirements

In addition to the ones derived from the Requirements Report, there are more requirements to add that are agent-specific.

FIPA Compliance

AmbieSense seeks to exploit standards where available. In the agent developers' community the Foundation for Intelligent Physical Agents (FIPA) recently adopted the FIPA specifications as standard for academic and industrial exploitation. Therefore, AmbieSense seeks to adopt an agent framework compliant to the FIPA specifications.

Adhering to existing standards is not the only reason why FIPA compliance is regarded as an important requirement. Moreover, complying with the FIPA specifications will fulfil all infrastructural requirements (AR4):

- Multi Agent Communication – FIPA compliant frameworks provide functionality that handles agent communication across different platforms
- Service discovery – a FIPA compliant framework discovers automatically which services are being offered by what agent
- Service management – FIPA compliant frameworks handle the offered services automatically within the agencies
- Support of multiple platforms

So, all the infrastructural requirements can be subsumed by a new requirement: FIPA compliance.

Runs on mobile devices

This requirement is more or less self-explanatory. In a system that distributes content to mobile users, with PDAs and smartphones as intended end user devices, it is natural that the agent framework must run on these gadgets.

Open platform

The A-MAS is to interact with other system components in AmbieSense. Therefore, it is of utmost importance that the technology used in the A-MAS is as interoperable as possible. As JAVA is selected to

be the common technology platform in AmbieSense, and with JAVA being the most widespread programming language for platform-independent applications, in fact, the framework has to be implemented in JAVA in order to fulfil the requirement.

The “Open platform” can be merged with AR3 (Integration with AmbieSense Components).

Open Source

In order to adopt the framework for the AmbieSense-specific tasks, it may be necessary to make changes in the framework code. If the framework source is publicly available making these changes will be much easier and can be done without involvement of third parties.

Representation languages

The AmbieSense agents may communicate contextual information as well as information on available content. This task will be simplified if the agent framework supports the use of representation languages for context and content.

Agent Framework Requirements – Revisited

After merging the original requirements with the agent-specific ones, the final set of requirements was divided into two priority groups: *must* and *should*, where failure to fulfil a *must*- requirement rules out the framework in question for use in AmbieSense. Table 2 provides an overview of the revised agent framework requirements for AmbieSense:

Requirement Nr.	Requirement Description	Priority
AR1	Runs on mobile devices	must
AR2	Open Platform	must
AR3	FIPA Compliant	must
AR4	Programmable agent development platform	must
AR5	Open Source	should
AR6	Representation languages	should

Table 2: Agent Framework Requirements

Requirements AR1 to AR4 belong to the *must* category. That means if a framework does not run on mobile devices, fails to provide an open, integratable platform, if is not FIPA compliant or fully programmable, it will not be considered. However, also the *should*-requirements should be met. They can be compensated by a strong performance in every other requirement only.

Agent Framework Survey Summary

This section sums up the evaluation of the selected frameworks against the set of requirements. The selected frameworks were evaluated because they all claim to be FIPA compliant⁹ and thus fulfil requirement AR3. The complete evaluation is available in appendix 8.6

- Tryllian
- April
- Comtec
- FIPA-OS
- Grasshopper
- Jack
- JADE/LEAP
- JAS
- ZEUS

There is no documentation for the use of either **April**, **Comtec**, **JACK** or **Zeus** on mobile devices. In addition, April is not implemented in Java, nor is JACK available as open source. Furthermore, ZEUS does not seem to have been maintained since May 2001. None of these frameworks will therefore be further considered.

JAS seems to be an interface specification rather than a fully programmable agent framework, so it fails to meet requirement AR4.

Tryllian and **Grasshopper** meet all *must*-requirements but do not come as Open-Source software. Even if not a *must*-requirement, lack of source code might slow down the development in case adjustments of the framework kernel are needed.

The table below provides an overview over the fulfilment of requirements of each framework.

Framework	AR1	AR2	AR3	AR4	AR5	AR6
	Mobile Platform	Open Platform	FIPA	Programmable	Open Source	Representation languages
Tryllian	yes	yes	yes	yes	no	yes
April ¹⁰	no	no	yes	yes	yes	yes
Comtec	no	yes	yes	yes	yes	yes
FIPA-OS	yes	yes	yes	yes	yes	yes
Grasshopper	yes	yes	yes	yes	no	yes
JACK	no	yes	yes	yes	no	yes
JADE/LEAP	yes	yes	yes	yes	yes	yes
JAS ¹¹	yes	yes	yes	no	yes	no
ZEUS ¹²	n/a	yes	yes	yes	yes	yes

Table 3: Agent Framework Survey

⁹ The list of agent platforms is adopted from <http://www.fipa.org/resources/livesystems.html>

¹⁰ Implements the April language, not JAVA

¹¹ Appears to be an interface specification rather than a fully-featured agent development framework.

¹² ZEUS has not been maintained since May, 23rd, 2001

The table sums up the evaluation of the frameworks against the final set of requirements. It shows which framework does or does not fulfil each requirement. Failure to meet a *must*-requirement is stressed by bold characters.

Conclusion

The following section sums up the evaluation process and the arguments put forward when selecting JADE/LEAP as the framework of choice.

Framework	Fulfil all <i>must</i> -requirements	Fulfil all <i>should</i> -requirements
Tryllian	NO	NO
April	NO	YES
Comtec	NO	YES
FIPA-OS	YES	YES
Grasshopper	YES	NO
JACK	NO	NO
JADE/LEAP	YES	YES
JAS	NO	NO
ZEUS	NO	YES

Table 4: Framework Evaluation Summary

As Table 4 shows, only FIPA-OS and JADE/LEAP manage to meet all the requirements in the evaluation process. Since JADE/LEAP and FIPA-OS both fulfilled all requirements, they had to be evaluated against each other. Even if Grasshopper and Tryllian fulfil all *must*-requirements, the fact that they are not available as open source software rules them out against the other two.

In an internal evaluation of the two remaining frameworks within the AmbieSense Consortium, the following arguments were posted:

- There has been former positive experience with the JADE platform in the consortium.
- LEAP was developed in another IST project¹³. Using this framework in AmbieSense contributes to a consistent way of utilising EU-funded research.
- With the release of JADE version 3 (19.03.2003), the JADE developer team manages LEAP; LEAP is now integrated in JADE as an add-on, thus ensuring closer integration of these frameworks as well as support and further development.
- JADE/LEAP has a very active developer community, thus ensuring support and further development even after the project's end.

Since there were no according arguments in favour of FIPA-OS put forward, the AmbieSense consortium decided to run the first iteration of the A-MAS design within the JADE/LEAP framework.

¹³ The LEAP project: IST-1999-10211 (<http://leap.crm-paris.com/>)

APPENDIX 2: TEST PLAN FOR THE A-MAS

This appendix provides an overview of the scheduled testing activities connected to the implementation of the A-MAS. Test results, especially of tests conducted at the test sites Oslo Airport and Seville will be provided in the 12th deliverable: AmbieSense Testing and Evaluation Report. A general evaluation of the A-MAS will also be delivered within the 9th deliverable of the AmbieSense project (AmbieSense Multi-Agent System and Report).

June 03-November 03:

- Agent testing, functionality testing

November 03 – January 04:

- Integration of Agents and context middleware + testing

February 04:

- Integration of Agents and User Interface, preliminary system tests
- Integration with CORPORUM search and indexing tools
- Integration with Java Creek

April 04:

- Factory Acceptance Test
- Usability test at Oslo Airport
- In this test the following core entities will be integrated:
 - AmbieSense GUI
 - AmbieSense Context Middleware
 - AmbieSense Agents
 - Oslo Airport content database

March 05 – May 04

- Integration with CIP components + testing

APPENDIX 3: GLOSSARY

Agent	A computer program that acts autonomously in order to achieve a certain state/goal
A-MAS	The AmbieSense Multi-Agent System
Case-based Reasoning	A problem-solving system that relies on stored representations of previously solved problems and their solutions.
Content	Any kind of information downloaded to the end user's client in AmbieSense
Content Integration Platform	A set of functional packages for the storage and distribution of content in AmbieSense
Context	Description of aspects of a situation
Context-awareness	Ability of applications to consider a user's context
Context Middleware	Software for context storing, context retrieval, context matching, context merging, context security, context-content linking, context storage synchronization
Context Space	The complete set of contexts that is stored together in an electronic repository. The context space can be viewed as a database of past context, future contexts, and the current context.
Context Tag	The Context Tag is an entity that enables the binding of location to a context (or a set of contexts). A context tag may be a concrete computer running software or just the software itself.
DPEC	EC Directive on Privacy and Electronic Communications
FIPA	Foundation for Intelligent Physical Agents
HTTP	Hypertext Transfer Protocol. The protocol that enables the World Wide Web.
IIOB	Internet Inter-ORB Protocol. A protocol used for communication between CORBA object request brokers
Information Retrieval	Actions, methods and procedures for recovering stored data to provide information on a given subject.
JADE/LEAP	The Java Agent Development Environment and its Lightweight Extension
Multi-Agent System	An Environment for multiple Software Agents that interact in order to execute certain tasks
Ontology	An ontology is a formal, explicit specification of a shared conceptualization.
RMI	Remote Method Invocation
TCP/IP	(Transmission Control Protocol / Internet Protocol) - The protocols, or conventions, that computers use to communicate over the Internet.
UML	Unified Modelling Language

APPENDIX 4: CLASS DIAGRAMS

Class Diagram

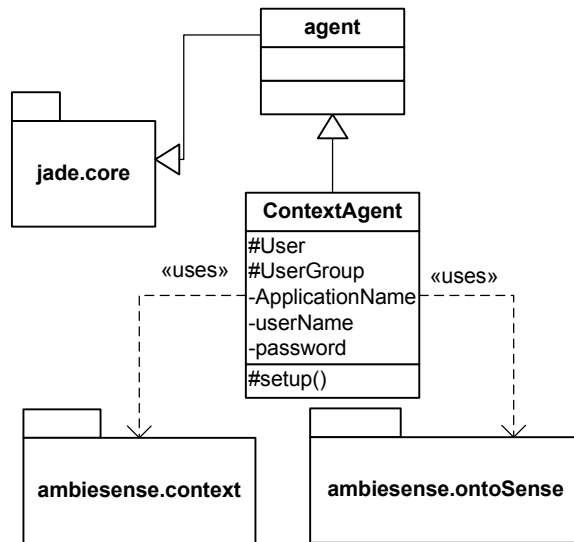


Figure 18: Context Agent, class diagram

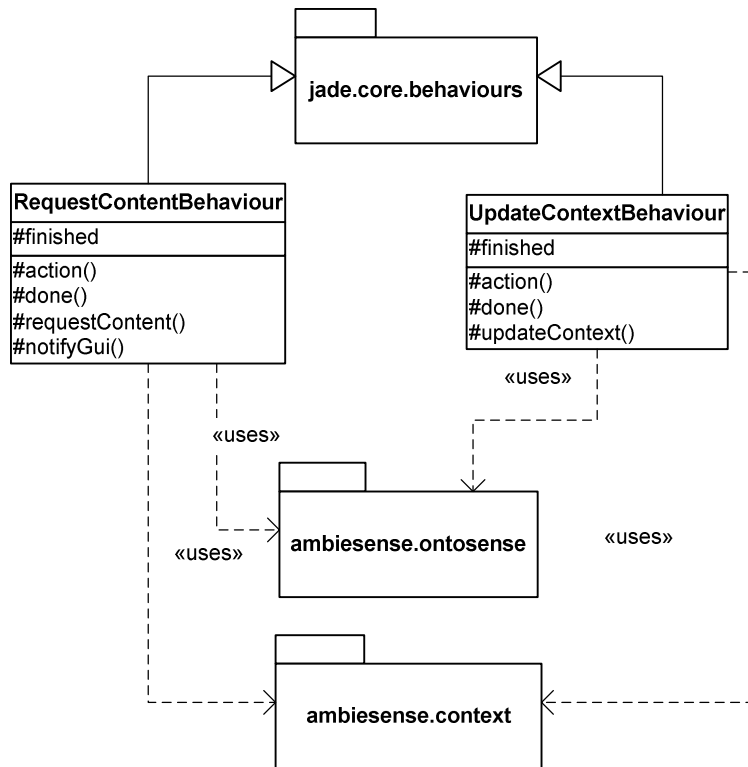


Figure 19: Context Agent behaviours, class diagram

CLASS DIAGRAM

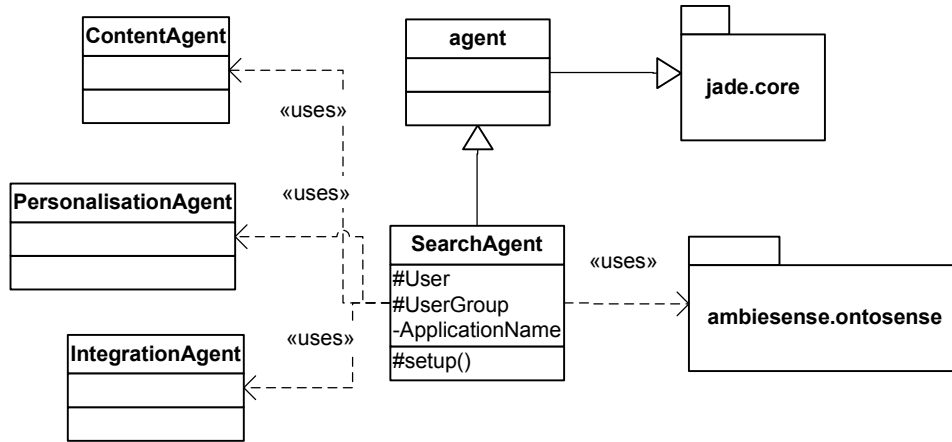


Figure 20: Search Agent, class diagram

CLASS DIAGRAM

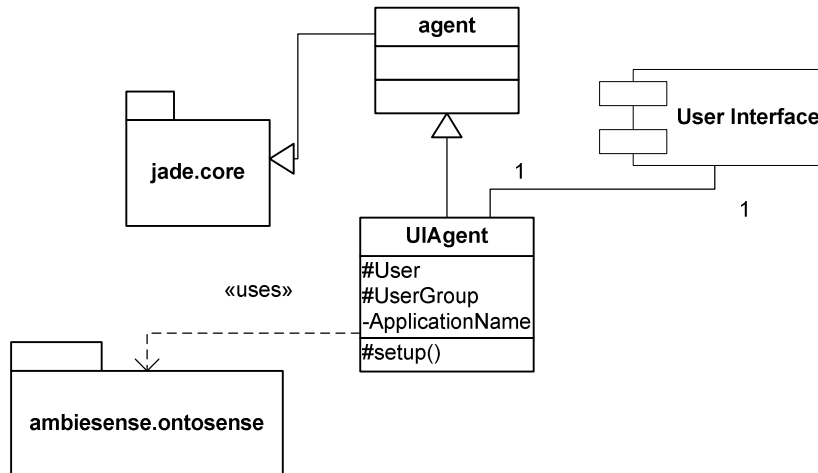


Figure 21: User Interface Agent, class diagram

CLASS DIAGRAM

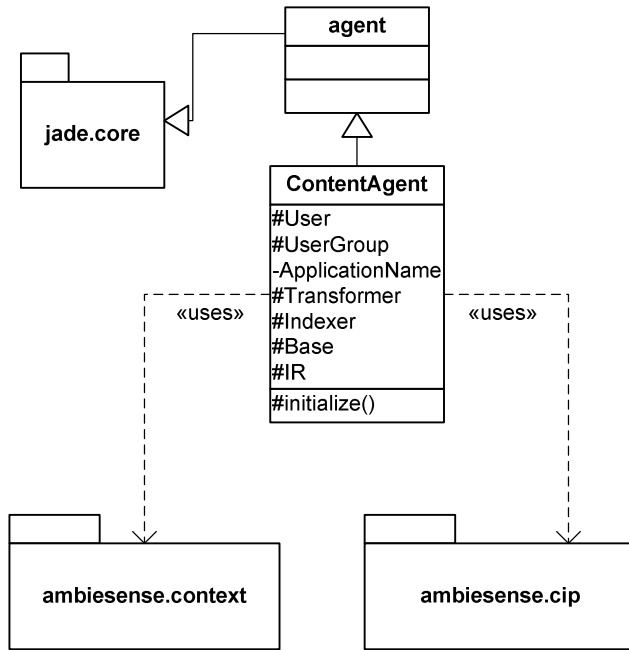


Figure 22: Content Agent, class diagram

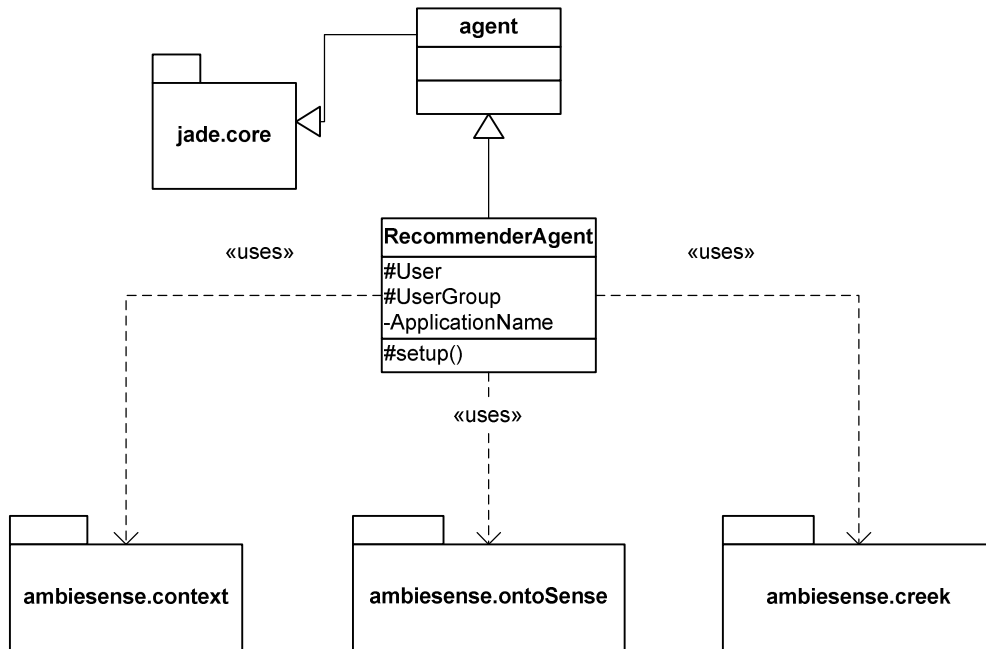


Figure 23: The Recommender Agent, class diagram

7.1.1.1 Class Diagram

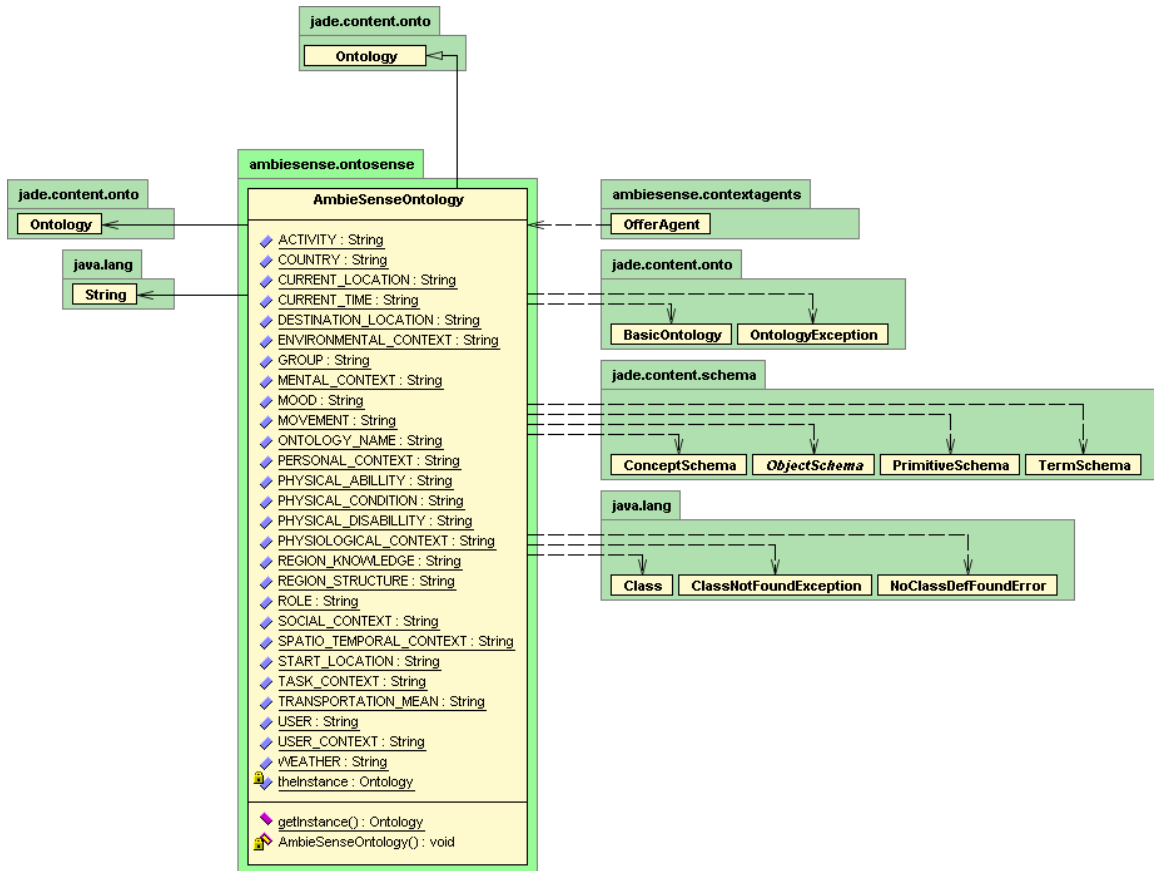


Figure 24: The AmbieSense ontology, class diagram

7.1.1.2 Class Diagram

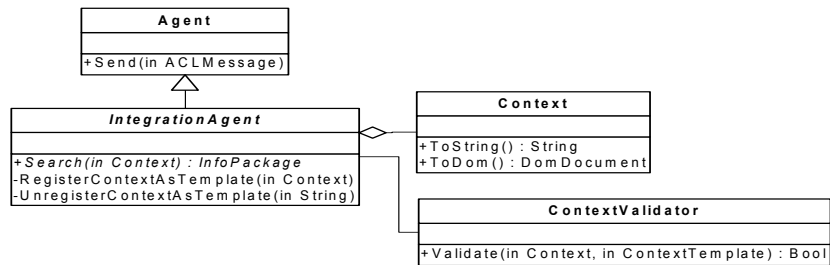


Figure 25: Integration Agent, class diagram

CLASS DIAGRAM

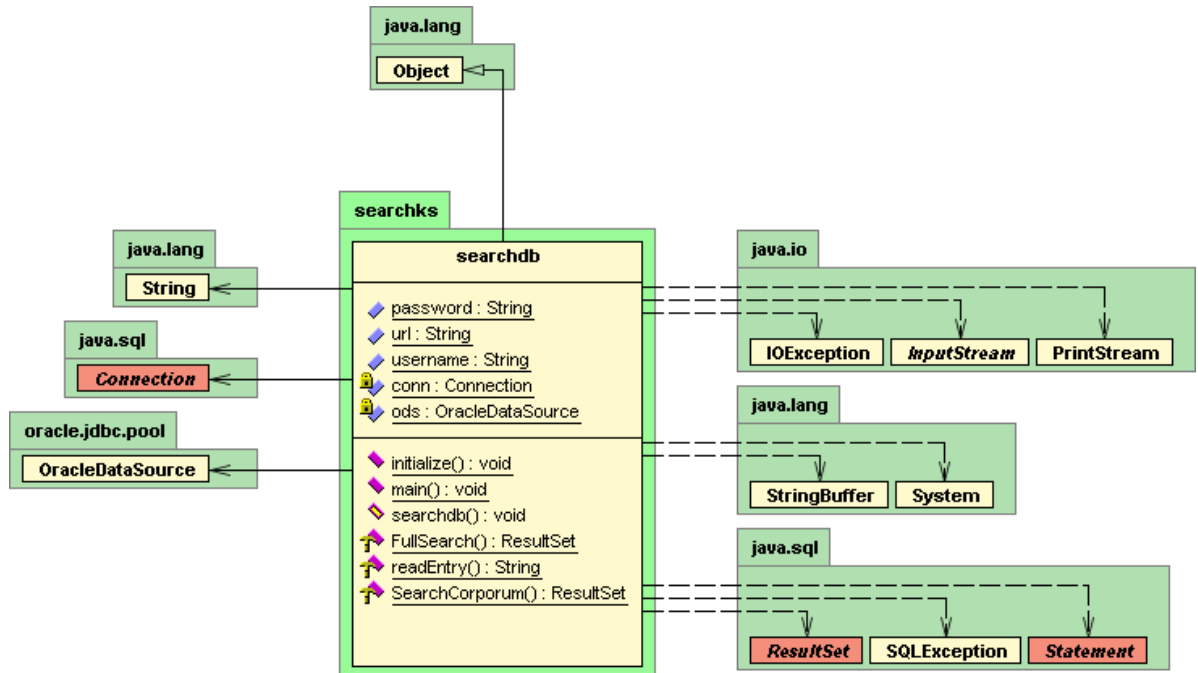
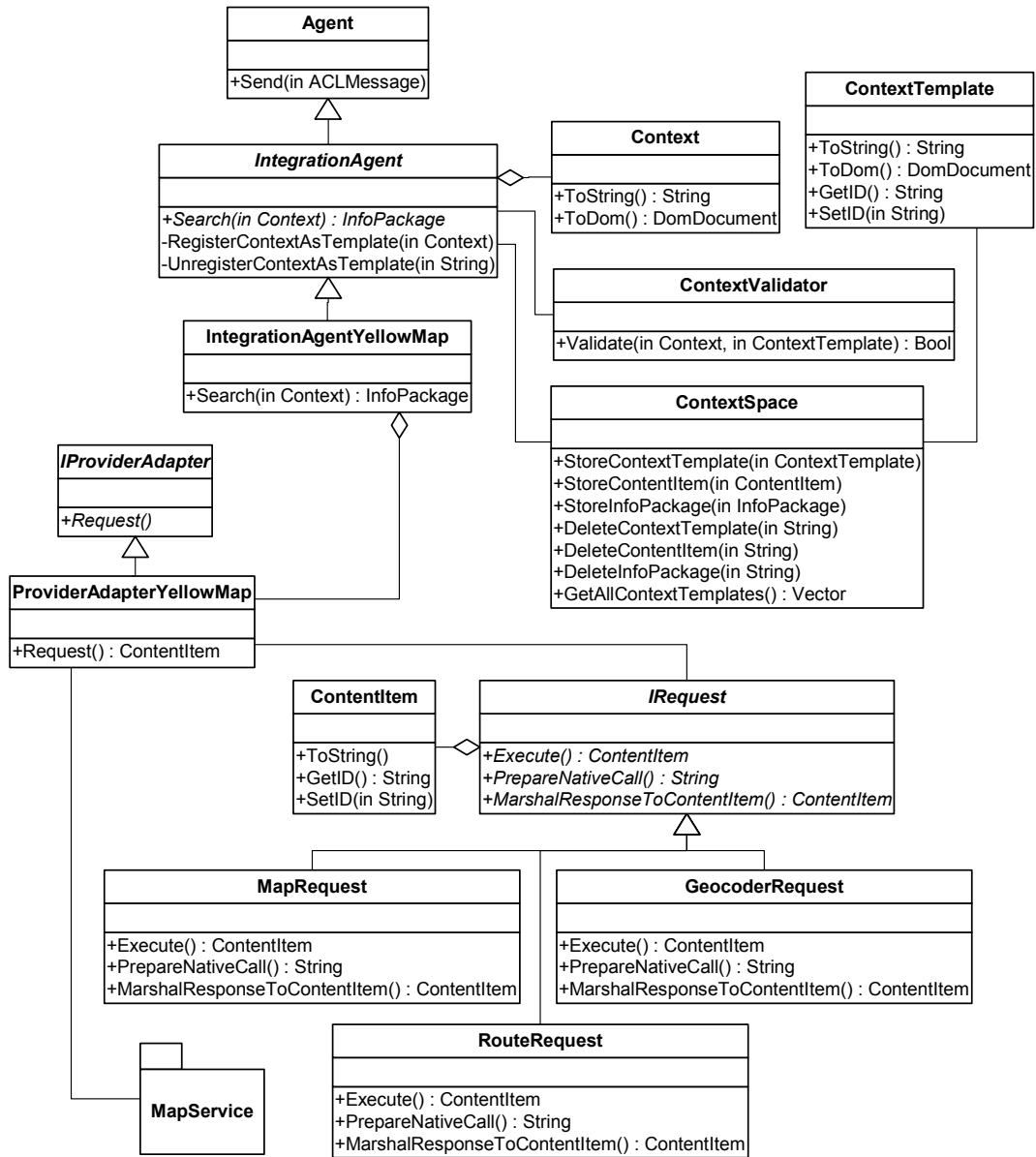


Figure 26: Corporum Wrapper, class diagram

CLASS DIAGRAM



APPENDIX 5: CLASS DESCRIPTIONS

Class: SearchAgent

Stereotype(s):

Definition

The Search Agent class is derived from the jade.core.agent class. It reacts to certain messages that are sent to it by other agents, rather than having public classes. Upon receiving a message, the Search Agent will check if it has a properly defined behaviour for this message, if not, it will reply with a “not understood” message.

The Search Agent will only reply to query_ref messages containing a context object, sent by the Context Agent. This message will be forwarded to three (or more as this part is generic) specific wrapper agents that will convert the context to a proprietary query fit for the search module. The Search Agent keeps track of the available search types/modules and will always forward the context to each of these modules.

Attributes

Name	Definition
user : User	The agent’s user
userGroup : UserGroup	The agent’s user group
applicationName : ApplicationName	Application domain of the agent

Associations

No.	Role	Target	Cardinality
1.	A SearchAgent can use zero or more IntegrationAgents	IntegrationAgent	0..*
2.	A SearchAgent can use zero or more ContentAgents	ContentAgent	0..*
3.	A SearchAgent can use zero or one PersonlisationAgent	PersonlisationAgent	0..1
4.	A SearchAgent uses one instance of the ontology	AmbieSenseOntology	1

Generalisation

Super-class

ambiesense.agents.AmbieSenseAgent

Sub-classes

Implemented interfaces

Notify(string: sender; integer contextID)

TEXTUAL DESCRIPTION

Class: UIAgent

Stereotype(s):

Definition

The UI Agent class is of the type Agent and therefore it reacts to certain messages that are sent to it by other agents, rather than having public classes for this purpose. However, since this Agent also maintains contact with non-agent enabled applications (the GUI on the mobile device), it does have some public classes.

The agent part of this class only reacts to query_ref messages containing a context object, sent by the Context Agent. This message will interpreted and will then be notified to the GUI.

From the UI changes can be made, this will result in the UI calling the UI Agent public class. The UI Agent will forward this change to the Context Agent.

Associations

No.	Role	Target	Cardinality
-----	------	--------	-------------

1.	A GUI Agent one UI	GUI	0..1
2.	A GUI can use one UI Agent	GUI Agent	1

Generalisation

Super-class
AmbieSenseAgent

Sub-classes**Implemented interfaces**

Notify(string: sender; integer contextID)
+update(Context c)

Interface: UIAgent**Stereotype(s):****Definition**

Update is the interface to the UIAgent class, which can be accessed by the GUI on the mobile device to notify context changes.

Methods

Name, attributes and return type	Definition
Update, context c, void	

Generalisation

Super-class
Jade.core.agent

Implemented by classes

UIAgent

Interface: Prompter**Stereotype(s):****Definition****Methods**

Name, attributes and return type	Definition
<i>public void informUser(java.lang.String informationText, boolean requireAcknowledgment)</i>	<p>This method should be used if the user should get an information text.</p> <p>Parameters: informationText: a string containing the text to be shown to the user requireAcknowledgment: a boolean telling whether the caller requires acknowledgment or not. It's up to the implementer to decide by which means the user should acknowledge.</p>

Generalisation

Super-class

Sub-classes

N/A

Implemented by classes

GUI

Interface: ContentChangeListener**Stereotype(s):****Definition****Methods**

Name, attributes and return type	Definition
<i>public long newContentItemAdded()</i>	<p>Callback function that tells the listener that a new relevant content item record is added.</p> <p>Returns: a long value with the identifier of the ambiesense.context.ContentItem object that is added.</p>
<i>public long contentItemChanged()</i>	<p>Callback function that tells the listener that a relevant content item record is changed.</p>

	<p>Returns: a long value with the identifier of the ambiesense.context.ContentItem object that is changed.</p> <p><i>public long contentItemRemoved()</i></p> <p>Callback function that tells the listener that a relevant content item record is removed.</p> <p>Returns: a long value with the identifier of the ambiesense.context.ContentItem object that is removed.</p>
--	--

APPENDIX 6: TABLE OF FIGURES

Figure 1: Natural kinds of agency, after Franklin and Graesser (1996)	11
Figure 2: Results of a survey of agent definitions	13
Figure 3: The AmbieSense Multi-Agent System Reference Architecture	17
Figure 4: Agents in the A-MAS	20
Figure 5: AmbieSense Logical Software Architecture	26
Figure 6: FIPA reference architecture	27
Figure 7: AUML representation of the FIPA Query Interaction Protocol	28
Figure 8: The A-MAS architecture (class diagram)	29
Figure 9: AUML Sequence Diagram Context Agent Request Behaviour	31
Figure 10: AUML Sequence Diagram Context Agent Change Behaviour	32
Figure 11: AUML Sequence Diagram Context Agent Link Content Behaviour	32
Figure 12: Sequence Diagram SearchAgent.DoSearch	33
Figure 13: AUML Sequence Diagram User Interface Agent Behaviour	35
Figure 14: AUML Interaction Diagram Recommender Agent	37
Figure 15: Identifying Situations with Case Based Reasoning	38
Figure 16: The AmbieSense Ontology, OSL scenario	40
Figure 17: Interaction of the Integration Agent with a proprietary system	41
Figure 18: Context Agent, class diagram	53
Figure 19: Context Agent behaviours, class diagram	53
Figure 20: Search Agent, class diagram	54
Figure 21: User Interface Agent, class diagram	54
Figure 22: Content Agent, class diagram	55
Figure 23: The Recommender Agent, class diagram	55
Figure 24: The AmbieSense ontology, class diagram	56
Figure 25: Integration Agent, class diagram	56
Figure 26: Corporum Wrapper, class diagram	57

APPENDIX 7: REFERENCES

- Aamodt, Agnar and Enric Plaza (1994): Case-based reasoning: Foundational issues, methodological variations, and system approaches. *AI Communications*, 7(1):39-59, 1994.
- AmbieSense (2002). Requirements and Overall System Architecture Report.
- AmbieSense (2003). The AmbieSense Reference Information Model
- Ball, G., Ling, D., Kurlander, D., Miller, J., Pugh, D., Skelly, T., et al. (1996). Lifelike Computer Characters: The Persona project at Microsoft Research. In J. M. Bradshaw (Ed.), *Software Agents*. Cambridge, Massachusetts: AAAI Press/MIT Press.
- Bassoli, E. Intelligent Agents & Privacy. *Lea 2002 Workshop On The Law Of Electronic Agents in connection with AAMAS*. University of Bologna, Italy, 2002.
- Bellifemine, F. Poggi, A. and Rimassa, G. (2000). Developing multi-agent systems with a FIPA-compliant agent framework. In: *Software - Practice And Experience*, 2001 no. 31, p. 103-128
- Collin Bennet. An International Standard for Protection: Objections to the Objections. 10th Conference on Computers, Freedom and Privacy, Toronto, 2000
- Borking, J. J., Eck, B. M. A. V., and Siepel, P. Intelligent Software Agents and Privacy: Registratiekamer-Dutch Data Protection Authority, 13, 1999.
- Collins, H. M. (1990). *Artificial Experts: Social Knowledge and Intelligent Machines*. Cambridge, Massachusetts: MIT Press.
- Dickinson, I. (1998). *Human-Agent Communication* (No. HPL-98-130). Bristol, UK: Hewlett-Packard Laboratories.
- Foner, L. N. (1993). What's An Agent, Anyway? A Sociological Case Study (Agents Memo No. 93-01). Cambridge, MA: MIT Media Laboratory.
- Franklin, S., & Graesser, A. (1996). Is it an Agent, or just a Program?: A Taxonomy for Autonomous Agents. Paper presented at the Third International Workshop on Agent Theories, Architectures, and Languages.
- Gruber, T. R. (1993): A translation approach to portable ontologies. *Knowledge Acquisition*, 5(2):199-220, 1993.
- Houmb, Siv Hilde (2002): Security Issues in Fipa Agents. The Norwegian University of Science and Technology: <http://www.idi.ntnu.no/emner/dif8914/essays/Houmb-essay2002.pdf>
- Kay, A. (1990). User Interface: A Personal View. In B. Laurel & S. J. Mountford (Eds.), *The Art of Human-Computer Interface Design* (pp. 191-207): Addison-Wesley.
- Koda, T., & Maes, P. (1996). Agents with Faces: The Effects of Personification of Agents. Paper presented at the HCI'96, London, UK.
- Kofod, A. & Aamodt, A. (2003) Case-Based Situation Assessment in a Mobile Context-Aware System AIMS, 2003.
- Laurel, B. (1990). Interface Agents: Metaphors With Character. In B. Laurel & S. J. Mountford (Eds.), *The Art of Human-Computer Interface Design* (pp. 355-365): Addison-Wesley.
- Laurel, B. (1991). *Computers as Theatre*. Reading, Massachusetts: Addison-Wesley.
- Maes, P. (1994). Agents that Reduce Work and Information Overload. *Communications of the ACM*, 37(7), 31-40.
- Masterton, S. J. (1998). Computer support for learners using intelligent educational agents: the way forward. Paper presented at the Sixth International Conference on Computers in Education (ICCE'98), Beijing, China.
- Maximini, Kerstin, Rainer Maximini, and Ralph Bergmann (2003): An investigation of generalized cases. In Kevin D. Ashley and Derek G. Bridge, editors, *ICCBR 2003, Case-Based Reasoning Research and Development*, LNAI 2689, pages 261-276. Springer-Verlag, 2003.
- Milgram, S. (1974). *Obedience to Authority*: Tavistock Publications.

- Myrhaug, Hans Inge and Ayse Göker (2003). Ambiesense – interactive information channels in the surroundings of the mobile user. In Constantine Stephanidis, editor, Universal Access in HCI, 10th International Conference on Human-Computer Interaction, volume 4, pages 1158–1162. Lawrence Erlbaum Associates, 2003.
- Rocha, Ana Paula and Eugenio Oliveira: Electronic institutions as a framework for agents' communication and mutual commitment. In: Portuguese Conference on Artificial Intelligence 2001.
- Rosch, E., & Mervis, C. (1975). Family Resemblances: Studies in the Internal Structure of Categories. *Cognitive Psychology*, 7, 573-605.
- Sadeh, Norman, Enoch Chan & Linh Van (2002): MyCampus: An Agent-Based Environment for Context-Aware Mobile Services. Workshop on Ubiquitous Agents in embedded wearable, and mobile devices. Bologna, 2002.
- Shneiderman, B. (1997). Direct Manipulation versus Agents: Paths to Predictable, Controllable, and Comprehensible Interfaces. In J. Bradshaw (Ed.), *Software Agents* (pp. 79-96). Cambridge, Massachusetts: AAAI Press / MIT Press.
- Shoham, Y. (1992). Agent Oriented Programming. *Artificial Intelligence*, 60(1), 51-92.
- Watt, S. N. K., Zdrahal, Z., & Brayshaw, M. (1995). Multiple agent systems for configuration design. In J. Hallam (Ed.), *Frontiers in artificial intelligence and applications* (pp. 217-228): IOS Press.
- Wooldridge, M. (1997). Agent-based Software Engineering. *IEE Proceedings on Software Engineering*, 144(1), 26-37.
- Wooldridge, M. (2002). *An Introduction to Multiagent Systems*. New York: Wiley
- W3C Consortium. Privacy Preferences Project (P3P), 2003. <http://www.w3c.org/p3p>
- Zimmermann, Andreas (2003): *Context-awareness in user modelling: Requirements analysis for a case-based reasoning application*. In Kevin D. Ashley and Derek G. Bridge, editors, IC-CBR 2003, Case-Based Reasoning Research and Development, LNAI 2689, pages 718 732. Springer-Verlag, 2003.