3

Towards a Reference Architecture for Context-Aware Services

Axel Bürkle, Wilmuth Müller and Uwe Pfirrmann
Fraunhofer Institute for Information and Data Processing
Germany

1. Introduction

This Chapter describes an infrastructure for multi-modal perceptual systems which aims at developing and realizing computer services that are delivered to humans in an implicit and unobtrusive way. The framework presented here supports the implementation of human-centric context-aware applications providing non-obtrusive assistance to participants in events such as meetings, lectures, conferences and presentations taking place in indoor "smart spaces". We emphasize on the design and implementation of an agent-based framework that supports "pluggable" service logic in the sense that the service developer can concentrate on the service logic independently of the underlying middleware. Furthermore, we give an example of the architecture's ability to support the cooperation of multiple services in a meeting scenario using an intelligent connector service and a semantic web oriented travel service.

The framework was developed as part of the project CHIL (Computers in the Human Interaction Loop). The vision of CHIL was to be able to provide context-aware human centric services which will operate in the background, provide assistance to the participants in the CHIL spaces and undertake tedious tasks in an unobtrusive way. To achieve this, significant effort had to be put in designing efficient context extraction components so that the CHIL system can acquire an accurate perspective of the current state of the CHIL space. However, the CHIL services required a much more sophisticated modelling of the actual event, rather than simple and fluctuating impressions of it. Furthermore, by nature the CHIL spaces are highly dynamic and heterogeneous; people join or leave, sensors fail or are restarted, user devices connect to the network, etc. To manage this diverse infrastructure, sophisticated techniques were necessary that can map all entities present in the CHIL system and provide information to all components which may require it.

From these facts, one can easily understand that in addition to highly sophisticated components at an individual level, another mechanism (or a combination of mechanisms) should be present which can handle this infrastructure. The CHIL Reference Architecture for Multi Modal Systems lies in the background, and provides the solid, high performance and robust backbone for the CHIL services. Each individual need is assigned to a specially designed and integrated layer which is docked to the individual component, and provides all the necessary actions to enable the component to be plugged in the CHIL framework.

2. The CHIL Project

The architecture presented in this chapter is a result of the work carried out in the FP6 research project CHIL (IP 506909) partially funded by the European Commission under the Information Society Technology (IST) program.

2.1 Project Goals

The main idea behind the CHIL project is to put Computers in the Human Interaction Loop (CHIL), rather than the other way round. Computers should proactively and implicitly attempt to take care of human needs without the necessity of explicit human specification. This implies that a service is aware of the current context of its users in order to react adequately to a user's situation and environment. Computers are repositioned in the background, discretely observe the humans and attempt to anticipate and serve their needs like electronic butlers.

To realize this goal and to overcome the lack of recognition and understanding of human activities, needs and desires as well as the absence of learning, proactive, context-aware computing services, research in the following areas was carried out within the CHIL project:

- Perceptual Technologies: Proactive, implicit services require a good description of human interaction and activities. This implies a robust description of the perceptual context as it applies to human interaction: Who is doing What, to Whom, Where and How, and Why.
- Software Infrastructure: A common and versatile software architecture serves to improve interoperability among partners and offers a market driven exchange of modules for faster integration.
- Services: Context-aware CHIL services assisting humans interacting with each other are designed, assembled and evaluated. Prototypes are continuously developed and their effectiveness explored in user studies.

2.2 CHIL Services

CHIL focuses on two situations, in which people interact with people and exchange information to realize common objectives: meetings and lecture rooms. In order to support the interaction between humans in these scenarios, the following context-aware services have been implemented:

- The **Connector** attempts to connect people at the best time and by the best media possible, whenever it is most opportune to connect them. In lieu of leaving streams of voice messages and playing phone tag, the Connector tracks and knows its masters' activities, preoccupations and their relative social relationships and mediates a proper connection at the right time between them.
- The **Memory Jog** is a personal assistant that helps its human user remember and retrieve needed facts about the world and people around him/her. By recognizing people, spaces and activities around its master, the Memory Jog can retrieve names and affiliations of other members in a group. It provides past records of previous encounters and interactions, and retrieves information relevant to the meeting.
- The **Travel Service** provides assistance with planning and rearranging itineraries. It detects if a user is going to miss a scheduled travel connection e.g. due to the delay of a meeting, and automatically searches for alternative travel possibilities. Based on user

preferences, it provides a selection of the "best" travel possibilities found. The Travel Service can either be evoked directly by the user through one of his personal devices or triggered by another CHIL service.

- The **Meeting Manager** handles all meeting related issues. For example, it keeps track of organisational meeting data, e.g. the list of planned participants and the meeting location. It can display a personalized welcome message, when a participant enters the meeting room. Furthermore, the Meeting Manager Service provides support during a meeting. One of its functionalities is to automatically start the corresponding presentation when a presenter approaches the presentation area.

Other services such as a Socially Supportive Workspace were implemented by the various CHIL partners. In addition, a series of basic services for common and elementary domain tasks are available. CHIL, however, was not intended as a project to develop specific application systems. It was intended to explore them as exemplary placeholders for the more general class of systems and services that put human-human interaction at the centre and computing services in a supporting function on the periphery, rather than the other way round.

2.3 Project Results

The CHIL consortium (15 institutes and companies in 9 countries) was one of the largest consortia tackling the above mentioned problems to-date. It has developed a research and development infrastructure by which different CHIL services can be quickly assembled, proposed and evaluated, exploiting a User Centred Design approach to ensure that the developed services answer real users' needs and demands. The CHIL project has pushed the state-of-the-art in various perceptual technologies (Stiefelhagen & Garofolo, 2007; Stiefelhagen et al., 2008). Furthermore, we have developed a highly flexible architecture employing collaborative context-aware agents, which enable the implementation and provision of the CHIL services. The effectiveness of the developed components and technologies has been demonstrated by implementing a number of CHIL services.

3. Related Work

Developing architectural frameworks supporting ubiquitous computing projects is not a recent trend. Major pervasive and ubiquitous computing projects have placed significant effort in designing such platforms that can facilitate integration, debugging, development, management and eventually deployment of the end-services.

The m4 project (multimodal meeting manager) (Wellner et al., 2005) presents a client-server architecture using JSP-generated frames in a meeting browser to produce the output of audio and video streams and services. The AMI Project (Augmented Multi-Party Interaction) (AMI, 2008), probably the project most closely related to CHIL, focuses on technologies which are integrated by a plug-in mechanism of a relatively simple, browser-based framework that allows indirect communication between the modules. Both of them concentrate on technologies and adapt existing software for integration purposes. CHIL in contrast developed an architecture that is particularly designed for the integration of and direct communication between context-aware services.

Other previous research focused on distributed middleware infrastructures (Roman et al., 2002), on architecture frameworks for developers and administrators (Grimm et al., 2000),

on the design process and the development of frameworks and toolkits (Dey, 2000), on context-aware broker agents (Chen et al., 2004), on client-server architectures with a central server and multiple clients to support the management of multiple sensor input for different services (Johanson, 2002), on flexible, decentralized networks connecting dynamically changing configurations of self-identifying mobile and stationary devices (Coen et al., 1999), on architectures for coordination of I/O-devices and the exploitation of contextual information (Shafer et al., 1998), and on systems that span wearable, handheld, desktop and infrastructure computers (Garlan et al., 2002).
All these efforts justify the importance of ubiquitous computing architectures. At the same time they manifest that there is no global unified framework addressing all needs. Rather, the majority of these architectures concentrate on one or more application specific goals. A fundamental limitation of these architectures is that they assume that all context-aware components (e.g., perceptual components, situation modeling middleware) are contributed by the same technology provider, which is not always the case. CHIL as well as other emerging pervasive computing environments are composed by a variety of distributed heterogeneous components from multiple vendors. Another drawback of these architectures is that they were built upon legacy technologies and therefore do not benefit from emerging technologies (e.g., the semantic web). The CHIL architecture presented in the following sections is proposed as a reference model architecture for multi-modal perceptual systems.

4. The CHIL Reference Architecture

Due to the scale of the CHIL project with its large number of partners contributing with a diversity of technical components such as services, event and situation modelling modules, context extraction components and sensors, as well as the complexity of these components, a flexible architecture that facilitates their integration at different levels is essential. A layered architecture model was found to best meet these requirements and allow a structured method for interfacing with sensors, integrating technology components, processing sensorial input and composing services as collections of basic services.
In order to realize context-aware, proactive, and intelligent services, both context-delivering and collaborating components have to be integrated. Context is any information that can be used to characterize the situation of a person or computing entity that is considered relevant to the interaction between a user and an application (Dey, 2000). In CHIL, context is delivered both by perceptual components and learning modules. Perceptual components continuously track human activities, using all perception modalities available, and build static and dynamic models of the scene. Learning modules within the agents model the concepts and relations within the ontology. Collaboration is enabled by a set of intelligent software agents communicating on a semantic level with each other.
The result of the architecture design work is the CHIL Reference Architecture, which is an architectural framework along with a set of middleware elements facilitating the integration of services, perceptual components, sensors, actuators, and context and situation modelling scripts. The framework and the associated middleware elements facilitate integration and assembly of ubiquitous computing applications in smart spaces. Specifically, they mitigate the integration issues arising from the distributed and heterogeneous nature of pervasive, ubiquitous and context-aware computing environments.

4.1 Layer Model

The layered system architecture that facilitates the integration of these various components is presented in Fig. 1. According to the components' different functional levels such as audio and video streaming, perceptual components, sensor control, situation modelling, services and user interfaces, the architecture consists of 8 horizontal layers. They are completed by two vertical layers, the ontology and "CHIL utilities", which are accessible by all components and provide elementary functions relevant to all layers.

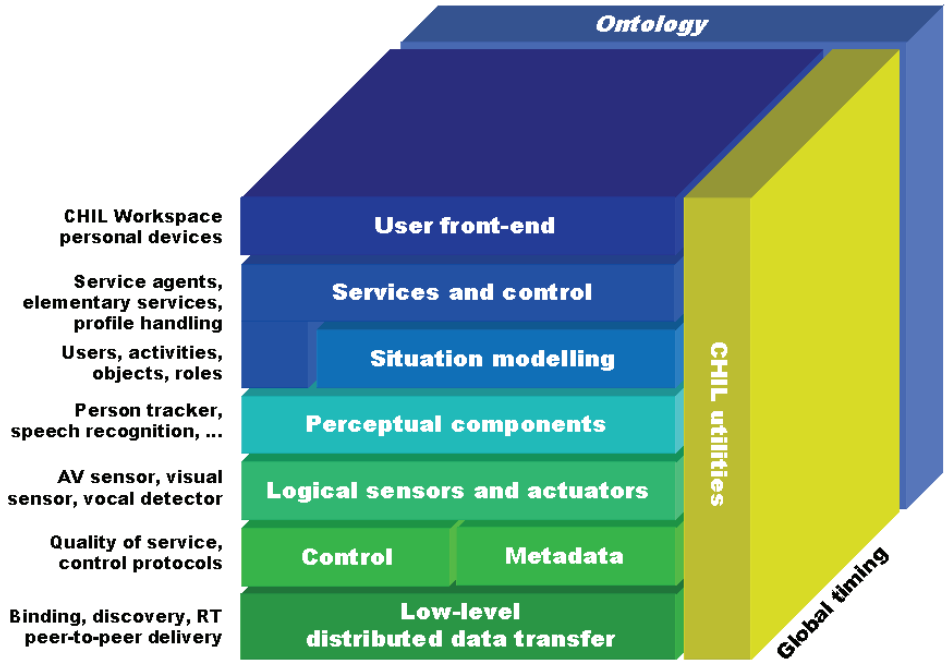


Figure 1. The CHIL architecture layer model

The lower layers deal with the management and the interpretation of continuous streams of video and audio signals in terms of event detection and situation adaptation and contain the perceptual components. The upper layers of the infrastructure enable reasoning and management of a variety of services and user interfacing devices. "User-front-end" and "Services and control" are implemented as software agents, thus the service and user interface management is based on agent communication. All layers use a common ontology as a backbone. A detailed description of the layers is given in the following section, the agent infrastructure of the CHIL system is described in detail in section 6.

4.2 Description of the Layers

User Front-End: The User Front-End contains all user related components such as the Personal Agents, Device Agents and the User Profile of a CHIL user. The Personal Agent acts as the user's personal assistant taking care of his demands. It interacts with its master through personal devices (e.g. notebook, PDA, smart phone) which are represented by

corresponding Device Agents. The Device Agents are the controlling interfaces to the personal devices. The Personal Agent also provides and controls access to its master's profile and preferences, thus ensuring user data privacy. The User Profile stores personal data and service-related preferences of a user.

Services and control: The Services and Control layer contains the components which implement the different services as well as service-to-service information exchange and user-service communication. The services are implemented through software agents. The Service and Control layer comprises both the service agents and their management. The interaction with other agents within this layer and the User front-end layer follows well-defined standards and the communication mechanisms of the agent platform, while communication with the other layers follows internal mechanisms. Services are both reusable basic services as well as complex higher-level services composed of suitable basic services.

Situation Modelling: The Situation Modelling layer is the place where the situation context received from perceptual components is processed and modelled. The context information acquired by the components at this layer helps services to respond better to varying user activities and environmental changes. For example, the situating modelling answers questions such as: Is there a meeting going on in the smart room? Is there a person speaking at the whiteboard? Who is the person speaking at the whiteboard? This layer is also a collection of abstractions representing the environmental context in which users act and interact. An ontological knowledge-base maintains up-to-date state of objects (people, artefacts, situations) and their relationships. Additionally it serves as an "inference engine" that regularly deduces and generalizes facts during the process of updating the context models as a result of events coming from the underlying layer of Perceptual Components. The searched-for situations are dictated by the needs of active services, as for the detection of the level of interruptability of a particular person in a room for whom a service has to monitor incoming calls. In the CHIL Reference Architecture, the situation model takes its information from the perceptual components and produces a set of situation events that are provided to services.

Perceptual Components: Perceptual Components are software components which operate on data streams coming from various sensors such as audio, video and RFID-based position trackers, process it, interpret it, and extract information relating to people's actions. Such information may be the people's locations, IDs, hand gestures, pose recognition etc.

The CHIL Project brought together perceptual component developers from different partners; all of them were using customized tools, different operating systems, a wide variety of signal processing (and other) libraries, and in general a plenitude of equipment which was not agreed upon from the early stages of the project. However, the CHIL Project had the vision to be able to combine the developed technologies and algorithms and deliver state-of-the-art services which would exploit the whole underlying sensor and actuator infrastructure and, ideally, would be independent of the technology provider. These circumstances had to be considered when designing the Perceptual Components tier.

The design of the Perceptual Components tier does not define the parameters of the core signal processing algorithm, but pertain to the input and output data modelling aspects of it; it considers the Perceptual Component as a "black box". The design specifies and gives guidelines, how perceptual components shall operate, "advertise" themselves, subscribe to receiving a specific sensor data stream and how they shall forward their extracted context to

the higher layer of the CHIL Reference Architecture. This specification incorporates all the interfaces for communicating with the sensors, the ontology and the services.

Logical Sensors and Actuators: Sensors and actuators are keys in the design and implementation of multi-modal perceptual services. They act as the "eyes and ears" of the system and provide a continuous flow of output data to the processing components which extract pertinent information by applying algorithms able to extract elementary context. This layer comprises several abstractions which wrap the sensor control and transmission components for each one of the sensors in the smart space. Several APIs for initializing the component, capturing the sensor data, for configuring the component, and for starting and stopping it are provided. Each sensor is controlled by a specified sensor controller, which provides the low-level commands to the sensor. The sensors, and therewith the logical sensor components produce a continuous flow of information which is transmitted using a particularly designed interface, the ChilFlow middleware (see below), to any consuming component in the CHIL system. Finally, each logical sensor is able to communicate with the framework's knowledge base where it can register and "advertise" itself to the rest of the framework.

Control/Metadata: Control and Metadata provide mechanisms for data annotation, synchronous and asynchronous system control, synchronizing data flows, effective storing and searching multi-media content and metadata generated by data sources.

Low-level Distributed Data Transfer: The Low-level Distributed Data Transfer layer of the CHIL Architecture is responsible for transferring high-volume and/or high-frequency data from sensors to perceptual components or between perceptual components. This layer is implemented by the ChilFlow data-transfer middleware. This middleware is heavily used by developers of perceptual components to distribute their components over several networked computers in order to exploit more computational power. In order to free developers from handling networking issues and managing the connections between components, ChilFlow offers an easy to master, yet powerful object-oriented programming interface, which provides type-safe network transparent one-to-many communication channels for data exchange between components. These communication channels are called flows. Flows provide higher abstraction level over ordinary network connections (e.g. TCP sockets) that fit the communication characteristics of components and simplify the work of the components' developers significantly.

CHIL utilities: This layer provides basic functionality that is needed by components in all layers of the framework. One particular example is global timing, an important issue in distributed, event driven systems like CHIL where time-stamped messages and streams are sent through the infrastructure.

Ontology: In order to enable the intended cognitive capabilities of the CHIL software environment, it is necessary to conceptualize entities and to formally describe relations among them. Through the ontology, CHIL software components both know the meaning of the data they are operating on and expose their functionality according to a common classification scheme.

The CHIL ontology is defined using the Web Ontology Language OWL (OWL, 2008). It comprises several modules that are physically represented by separate Web resources with distinct URLs, among them the fully integrated CHIL agent communication ontology. The idea of modularization is that software developers only need to reference those parts of the ontology that are relevant for them. Additionally, modularization increases performance, for

example when deploying the agent communication subset of the ontology in order to generate software agent code.
For managing the ontological data, as well as for reasoning and detecting inconsistencies, access to an ontology is typically backed by a central knowledge base management system. The CHIL Knowledge Base Server exposes the functionality of arbitrary off-the-shelf ontology management systems by a well defined API based on OWL. As such, it provides unified access to the central ontological knowledge base for heterogeneous platforms, programming languages and communication protocols. Together with this API and the CHIL ontology, the knowledge base server constitutes the Ontology layer of the CHIL architecture.

5. Sensor Infrastructure & Context-Awareness

Sensing infrastructures are a key prerequisite for realizing context-aware applications. Within CHIL several sites have constructed in-door environments comprising multi-sensor infrastructures called "smart rooms". They include:

- Microphones and microphone arrays (Brandstein & Ward, 2001)
- Cameras (fixed, active with pan, tilt and zoom (PTZ) or panoramic (fish-eye))
- RFID-based location trackers (Ubisense, 2007)

Based on these sensor infrastructures, a variety of perceptual components have been built and evaluated such as 2D and 3D-visual perceptual components, acoustic components, audio-visual components, RFID-based location tracking, as well as output components such as multimodal speech synthesis and targeted audio.
Perceptual components derive elementary contextual information; however in general they lack information about the overall current status of the people's interactions and environmental conditions. To be able to "fuse" many of these perceptual components together in order to determine more sophisticated and meaningful states, additional middleware interfaces have been developed to facilitate the intercommunication of these components. The middleware acts as a receptor of the whole range of the elementary context cues and processes them in order to map the resulting contextual information into a complex situation. This process is called situation recognition and is a major part of every human-centric ubiquitous application. Situation recognition in our implementation follows the "network of situations" approach. This scheme allows the interconnection of distinct cases (situations), which are connected with edges, forming a directed graph. A transition from one situation to another occurs if given constraints are satisfied. As soon as this transition is feasible, the service logic is applied, and the active state is reflected by the newly reached one. Context-awareness is hence modelled by this network. Fig. 2 illustrates such a network of situations which can be used to track situations during meetings in a "smart room".
A meeting is a sequence of three main situations: "MeetingSetup", "OngoingMeeting" and "MeetingFinished". Starting with the initial "MeetingSetup" and its sub state "Arrival", the situation changes to "WelcomeParticipant" when the person identification perceptual component signals that a person is entering the room; the new person will be welcomed by displaying a message. Based on location and activity information of all attendees, the "Meeting Detector" perceptual component recognizes the start of the meeting and triggers the transition to the next main situation "OngoingMeeting". During the meeting, the body

tracker perceptual component tracks the locations of the participants, detects when a person approaches or leaves the presentation area and switches the situation accordingly between the "Discussion" and "Presentation" state. The Meeting Detector again determines the end of the meeting and triggers the transition to the final "MeetingFinished" situation.

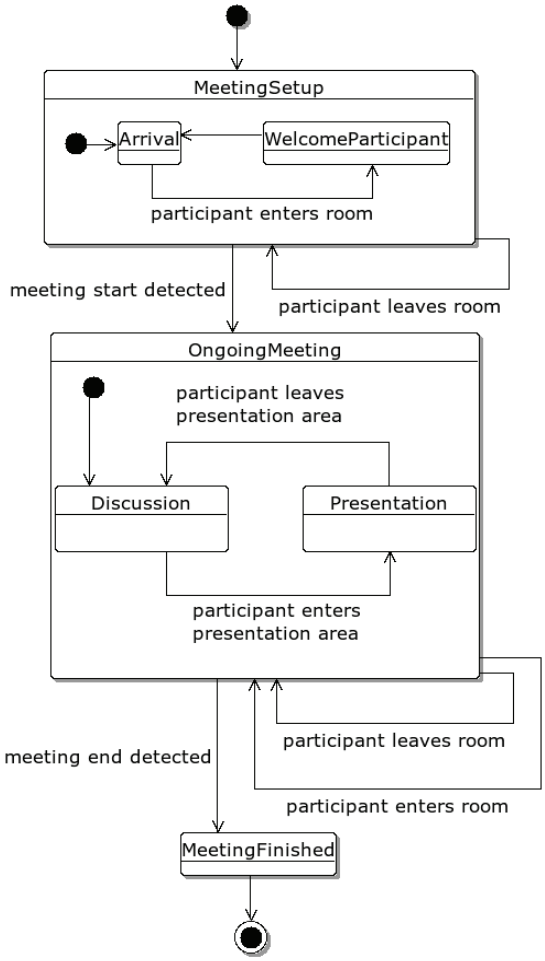


Figure 2. A "network of situations" model for tracking meetings in a smart room

6. Agent Infrastructure for Context-Aware Services

Context-aware services are usually based on complex heterogeneous distributed systems comprising sensors, actuators, perceptual components, as well as information fusion middleware. In such sophisticated systems agents are preferred (Huhns & Singh, 1999), since they are equipped with a large set of capabilities. An agent is "*...any entity that can be viewed as perceiving its environment through sensors and acting upon its environment through*

effectors" (Russell & Norvig, 2003). Incorporating that "*An agent is a computer system, situated in some environment, that is capable of flexible autonomous action in order to meet its design objectives.*" (Jennings et al., 1998) and an intelligent inter-agent communication, a multi-agent system seems to perfectly meet the challenges of such complex human-centric systems.

Autonomous agents are computational systems that inhabit some complex dynamic environment, sense and act autonomously in this environment, and by doing so, realize a set of goals or tasks for which they are designed (Maes, 1994). Hence, they meet the major requirements for the CHIL architecture: to support the integration and cooperation of autonomous, context-aware services in a heterogeneous environment. Summarized in this short term, the major goals for the infrastructure span discovery, involvement and collaboration of services as well as competition between services in order to perform a certain task the best way possible. Standardized communication mechanisms and protocols have to be considered to raise information exchange onto a semantic level and to ensure location transparency.

To this end, we have devised a multi-agent framework that meets the following target objectives:

- Facilitate the integration of diverse context-aware services developed by numerous different service providers.
- Facilitate services in leveraging basic services (e.g. sensor and actuator control) available within the smart rooms.
- Allow augmentation and evolution of the underlying infrastructure independent of the services installed in the room.
- Control user access to services.
- Support service personalization through maintaining appropriate profiles.
- Enable discovery, involvement and collaboration of services.
- Provide the human-oriented services to the end user the best way possible.

The JADE (Java Agent DEvelopment Framework) platform (JADE, 2008) was selected to be the backbone of our system as it is equipped with many features, which make it a highly flexible and secure platform. As seen in (Altmann et al., 2001), JADE performs adequately in many of the evaluation criteria which include agent mobility capabilities, administration, network traffic issues, stability, security, debugging capabilities etc. Finally, JADE is compliant to the standards for agent communication, agent management and message transfer of the IEEE Computer Society standards organization FIPA (Foundation for Intelligent Physical Agents) (FIPA, 2008).

The following sections describe the CHIL agent infrastructure and how we achieved our objectives. Moreover, they demonstrate how we realized a multi-agent system that is capable to "*solve problems that are beyond the individual capabilities or knowledge of each problem solver*" (Jennings et al., 1998).

6.1 Agent Description

The CHIL software agent infrastructure, shown in Fig. 3, is composed of three levels of agents: personal and device agents that are close to the user and offer the functionality and results of the human-centric services to the end user, basic agents providing elementary tasks like service discovery, collaboration and agent communication to the service developers, and service agents, which implement or cover the functionality of the various context-aware services.

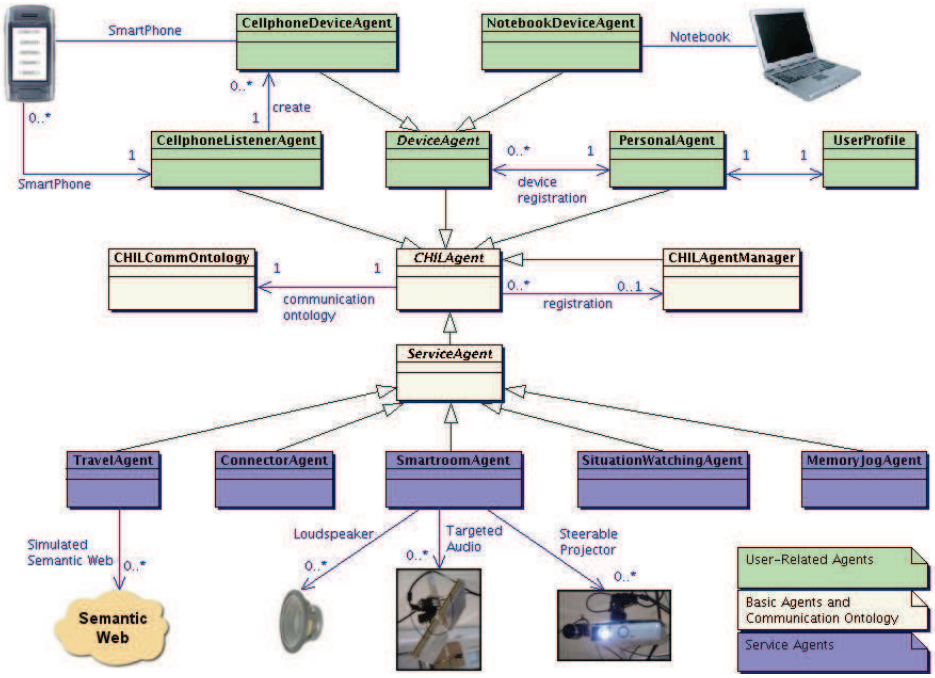


Figure 3. The CHIL software agent infrastructure

User-related Agents

Every person in the CHIL environment has a dedicated Personal Agent that acts as a personal secretary for him. Personal Agents are assigned during a login procedure and manage the complete interaction between its user and the CHIL system, supported by specialized device agents, which are bound to specific input and output devices. The Personal Agent knows what front-end devices its master has access to, how it can best receive or provide information and what input and notification types he prefers (notebook, cell phone call, SMS, targeted audio, etc.). Moreover, the Personal Agent is permanently aware of its master's context (location, environment, activity) by communicating with the Situation Watching Agent, in order to act or react in an appropriate way. Furthermore, the Personal Agent provides and controls access to its master's profile and preferences, thus ensuring user data privacy.

Similar to a user and his Personal Agent, each device has its own Device Agent that manages the complete communication between the device and a user's Personal Agent. The Notebook Device Agent handles the interaction between the graphical user front-end on the user's notebook and the user's Personal Agent, the Cell Phone Device Agent controls the communication between the Personal Agent and the user's cell phone. The Cell Phone Listener Agent supervises all incoming cell phone based requests: it watches a specific port and in case of an incoming call, it creates a Cell Phone Device Agent and passes over the socket connection to it. The newly created agent will then handle all further communication between the cell phone and the user's Personal Agent.

Basic Agents

The major basic agents in the CHIL environment are the CHIL Agent and the CHIL Agent Manager. The CHIL Agent is a subclass of the JADE Agent, which itself is the elementary agent in the JADE agent platform, and acts as the fundamental abstract class for all agents in the CHIL agent framework. It provides methods for essential agent administrative functionality (agent setup and takedown), for ontology-based messaging (create, send, receive messages, extract the message content), utility jobs like logging, and, in cooperation with the CHIL Agent Manager, for directory facilitator service (DF service) tasks such as register and deregister agents, modify agent and service descriptions and search service-providing agents based on a semantic service description. Special importance is attached to keep the agent communication conform to the FIPA (FIPA, 2008) specifications: the complete message transfer is compliant to the FIPA Interaction Protocols and the FIPA Communicative Acts and is based on a well-defined communication ontology, as recommended in the FIPA Abstract Architecture Specification.

The CHIL Agent Manager is a central instance encapsulating and adding functionality to the JADE Directory Facilitator (DF) and coordinating the selection and use of agents. Each CHIL agent registers its services with the CHIL Agent Manager, so the agent manager is, at any time, aware of all available agents, their abilities and the required resources. The CHIL Agent Manager can act both as a matchmaker and a broker. In case of an incoming service request, it knows which agents can solve the problem and which resources are needed. If the requested service can be provided by a single agent, it returns – serving as a matchmaker - the agent identifier of the capable agent to the requester, which in turn may contact the service providing agent. As a broker and in the case of a more complex problem, the CHIL Agent Manager decomposes the overall problem into sub problems, computes a problem solving strategy and invokes the subtasks on appropriate service agents. Having received all necessary partial results, it computes the overall solution and returns the final result to the initial requester.

Service Agents

Each Service Agent is associated with a specific service and provides access to the service functionality both to users and other agents. Service agents register the service functions with the CHIL Agent Manager using ontology based service descriptions, thus mapping the syntactical level of services to the semantic level of the agent community. These service descriptions can be queried both by Personal Agents in order to provide the service to human users and by other service agents, which may compose various functions from other services to supply their own one.

The basic Service Agent is an abstract ancestor class for all specific service agents; it provides the methods for the registration of service features including the necessary resources. Specialized service agents comprise the core functionality of the associated service whereas the service itself may be implemented as agent or may be covered by an agent, which then provides a suitable interface to the service. This applies to the Connector Agent, the Memory Jog Agent, the Travel Agent and the Meeting Manager Agent; the associated services have already been described in the section "CHIL Services".

Two special agents provide common abilities, which are useful for the whole agent society: the Smart Room Agent and the Situation Watching Agent. The Smart Room Agent controls the various optic and acoustic output devices in the smart room. It may communicate general messages to all attendees in the smart room by displaying them on the whiteboard

or transmitting them via loudspeaker. Furthermore, it is able to notify single participants without affecting other attendees, using the steerable video projector or targeted audio, dependant on the user's preferences.
The Situation Watching Agent wraps the Situation Model of the smart room. It monitors the smart room and tracks situation specific user information such as the current location, movement and activity on different semantic levels (simple coordinates as well as hints like "attendee X approaches whiteboard"). Moreover, it can provide location information of important artefacts like notebooks, whiteboards, tables etc. Other agents may query the current situation at the Situation Watching Agent as well as subscribe to events; both information retrieving methods are supplied by well-defined access and subscription APIs to the Situation Model.

6.2 Intelligent Messaging

As proposed in the FIPA Abstract Architecture Specification (FIPA, 2008), the complete information exchange between agents is based upon a well-defined communication ontology. Thus, the semantic content of messages is preserved across agents, a concept that becomes highly important by the fact that various services are implemented by a great number of developers in different places and the necessity of these developers to understand each other correctly.
The CHIL Communication Ontology is completely defined using the Web Ontology Language OWL (OWL, 2008) and fully integrated in the overall CHIL domain ontology. It is based upon the "Simple JADE Abstract Ontology", an elementary ontology provided by JADE, which must be used for every ontology-based message exchange within the JADE agent management system. The communication ontology extends the CHIL domain ontology by tokens which are indispensable for agent communication, particularly agent actions for requesting services and response classes to hold the results of the services and return them to the requesters. The ontology classes are used in the Java environment by means of the JADE abs package, found in *jade.core.abs*. Their handling is implemented by the basic CHIL Agent, providing it to the agent community by methods for ontology-based encoding, sending, receiving and decoding ACL (Agent Communication Language) messages.
A second level of intelligent messaging is achieved by the implementation of advanced agent coordination. Coordination of agents can be performed using well-defined conversation protocols (Cost et al., 2001). As a first approach, we use the interaction protocols and communicative acts specified by FIPA (FIPA, 2008). The communication support methods of the basic CHIL Agent, supplemented by initiator and responder classes for submitting and receiving messages, ensure that the agent communication is strictly compliant to the FIPA specification. Together with the central CHIL Agent Manger as the agent coordinating instance, the FIPA compliance and the ontology-based message transfer form a highly sophisticated approach of Intelligent Messaging.

6.3 Multiple Scalable Services

In the CHIL project, several complex context-aware services have been implemented in different places and had to be integrated in one system. To minimize the integration effort and to allow the service developers to concentrate on the core service functionality, a framework for the integration of services has been highly important. Hence, one of the

major goals of the CHIL agent infrastructure was to provide a mechanism that allows the distributed development of services, an easy integration and configuration and the cooperation and communication of multiple services in the CHIL system.
A simple service can easily be integrated by creating an agent which handles the framework tasks and the service control, and integrate the agent in the CHIL system. In this way the agent acts as a wrapper for the service; the agent could also embed the service logic itself.
But usually a service is more complex and requires particular functionality from other agents. For example, the Connector Service needs, in order to establish a connection the best way possible, knowledge about the user's connection preferences (phone call, SMS, notebook, audio) and the social relationship between two ore more participants (VIP, business or personal contact). For the sake of privacy, these personal information tokens must be held by the Personal Agent, although they are only used by the Connector Service. Thus, a service may require exclusive service specific functionality that is or must be realized by another agent than the service agent itself. Implementing such functionality in the agent itself implicates that all service providers which use this agent would have to synchronize the agent's code. This technique would quickly raise significant problems coordinating the implementation and configuration of software components.

6.3.1 Pluggable Behaviours

To this end, a plug-in mechanism has been designed that allows an easy integration of service specific functionality in other agents without modifying the agents' code: service specific agent behaviours are moved to pluggable handlers, which are agent independent and plugged into appropriate agents during their start-up phase. Using this mechanism, the agents themselves remain service independent, contain only the common methods and attributes all partners have agreed on, and thus become stable modules. Three types of pluggable handlers have been considered to be necessary, namely:

1. **Setup handler:** are responsible for service specific initialization and will be executed in the setup phase of an agent.
2. **Event handler**: are triggered by events from outside the agent world, e.g. the user's GUI, a perceptual component, the situation model or a web service.
3. **Responder**: are added to the agent's behaviour queue and react on incoming ontology-based ACL (Agent Communication Language) messages sent by other agents.

At start-up time each agent parses the service configuration files, determines which behaviours have to be instantiated and adds them to its behaviour queue. Since the code for this mechanism is concentrated in the basic CHIL Agent, the plug-in mechanism is available for all agents without additional implementation work for the agent developer. Moreover, the source of the configuration data could easily be switched; instead of using XML-based files, the service configuration could similarly be imported from a knowledge base.
Table 1 shows a sample implementation of a pluggable handler; the example is about a responder, which informs a user (i.e. the user's Personal Agent) about connection requests from other users.
The interface structure allows implementing the handler and the behaviour in two different classes; the handler will be recognised by the plug-in mechanism and provides the actual behaviour that will be added to the agent's behaviour queue. In this example, both are joined in one class: the responder implements the *PluggableResponder* interface and its *getAcceptedMessages* and *getBehavior* methods in order to be handled correctly by the plug-in

mechanism as well as extending a generic CHIL responder behaviour and overwriting the *prepareResponse* and *prepareResultNotification* methods.

```
public class InformAboutConnectResponder
  extends CHILSimpleAchieveREResponder implements PluggableResponder {
   public InformAboutConnectResponder(Agent agent) {
      super(agent, ((PersonalAgent)agent).matchAbsOntoRequestAction
                        (new AbsAgentAction("InformAboutConnect")));
   }
   public MessageTemplate getAcceptedMessages() {
      return getPA().matchAbsOntoRequestAction(
            new AbsAgentAction("InformAboutConnect"));
   }
   public Behavior getBehavior() {
      return this;
   }
   protected ACLMessage prepareResponse(ACLMessage request)
      throws CHILFipaRefuseException {
      // create the response message
      return responseMessage;
   }
   protected ACLMessage prepareResultNotification ACLMessage request,
      ACLMessage response) throws CHILFipaFailureException {
      // create the result message
      return resultMessage;
   }
}
```

Table 1. An example of a pluggable responder for the Personal Agent accepting ontology-based messages

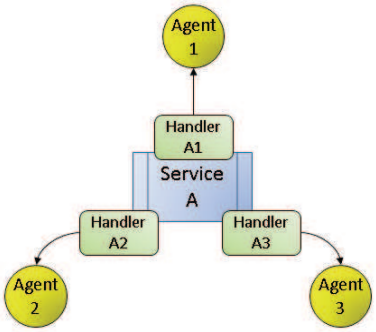
GetAcceptedMessages returns a JADE message template, i.e. a pattern that specifies all types of messages the responder will react to. In this case, it returns a template that accepts messages of type *request* containing the ontology-based agent action *InformAboutConnect*, the same message template that is used in the constructor. *GetBehavior* returns the actual behaviour, which realises the responder's functionality and which is added to the agent's behaviour queue. In this example it's the responder itself, but the developer may also create and return a separate behaviour object.

The *prepareResponse* and *prepareResultNotification* methods ensure FIPA compliance by implementing the agree/refuse and the failure/inform path of the FIPA Request Interaction Protocol respectively. The (optional) *prepareResponse* method receives the original request and informs the initiator, if the request is agreed to or refused. Additionally, it may start to compile the request. The (mandatory) *prepareResultNotification* completes the processing of the request and returns the results, or, in case of an error, a failure message to the initiator.

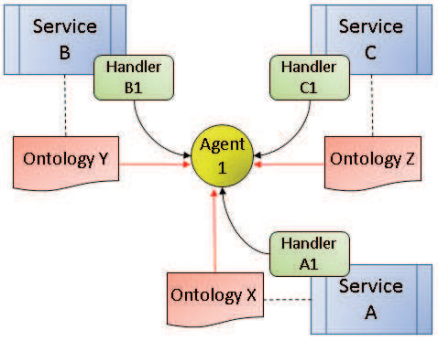
6.3.2 Scalable Services

The Pluggable Behaviours mechanism allows a service provider to plug new service specific functions in multiple agents without the need for recompilation (see Fig. 4(a)). However, to

fully exploit the features of all CHIL services and to raise the functionality of the whole system beyond the sum of its components, a service must be able to define new messages, which can be understood and compiled by other services. This means that each service provider should be able to define his own service specific ontology. As a consequence, each agent participating in such a multi-service communication has to be able to handle messages from different services and to work with several service ontologies, as illustrated in Fig. 4(b). Hence, service specific ontologies have to be handled in the same way as Pluggable Behaviours: the framework must be capable to plug them in without the need for recompiling the agent's code. To this end, the Pluggable Behaviours mechanism has been extended to Scalable Services by realising a plug-in technique for service specific communication ontologies.



(a)



(b)

Figure 4. Pluggable Handlers and Scalable Services

A new service is specified and integrated into the CHIL system by means of a XML-based configuration file. Table 2 shows an example of a service using its own ontology and adding an event handler to the service agent and a responder to the Travel Agent. The file defines all agents participating in the service, their behaviours and the service ontology. Each pluggable handler is specified by its type (setup, event and responder) and its class name. A priority value assigned to each handler can be used to determine the order of execution, which is particularly important for setup behaviours. The service ontology is specified by a name, the namespace, the location and the class name of the ontology class. Furthermore,

the configuration file provides an additional feature to system developers and administrators: it allows enabling/disabling certain functionality by simply adding/removing the appropriate configuration elements in the configuration file, without having to recompile the source code.

```
<?xml version="1.0" encoding="UTF-8"?>
<serviceconfig version="1.1"
 xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance
 xsi:noNamespaceSchemaLocation="CHIL_ServiceConfig_1.1.xsd">
  <service name="YourService">
    <agent name="YourAgent">
      <handler type="event" priority="1"
       classname="yourNamespace.yourService.yourAgent.YourEventHandler"/>
    </agent>
    <agent name="TravelAgent">
      <handler type="responder" priority="1"
       classname="yourNamespace.yourService.travelAgent.YourServiceResponder"/>
    </agent>
  </service>
  <ontology
     name="YourOntology"
     namespace="http://www.owl-ontologies.com/YourServiceOntology.owl"
     locationPath="$ChilHome/lib/yourService.jar"
     className="yourNamespace.ontology.YourOntology">
  </ontology>
</serviceconfig>
```

Table 2. Sample service configuration file using multiple agents and a service-specific communication ontology

Similar to a service configuration file informing a service about all participating agents, the CHIL system is informed about all participating services: a master configuration file, also based on XML, specifies the services that are activated on system start-up by their names and the location of their configuration files. A sample can be seen in table 3.

```
<?xml version="1.0" encoding="UTF-8"?>
<services version="1.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="../xmlschema/CHIL_Services_1.0.xsd">
    <service name="CoreService"          configFile="CoreService.xml"/>
    <service name="MeetingService"       configFile="MeetingService.xml"/>
    <service name="ConnectorService"     configFile="ConnectorService.xml"/>
    <service name="TravelService"        configFile="TravelService.xml"/>
    <service name="MemoryJog"            configFile="MemoryJog.xml"/>
    <service name="SmartroomService"     configFile="SmartroomService.xml"/>
    <service name="SitWatchService"      configFile="SitWatchService.xml"/>
</services>
```

Table 3. Master configurations file for services

6.4 Personalisation

A CHIL computing environment aims to radically change the way we use computers. Rather than expecting a human to attend to technology, CHIL attempts to develop computer assistants that attend to human activities, interactions and intentions. Instead of reacting only to explicit user requests, such assistants proactively provide services by observing the implicit human request or need, much like a personal butler would.

Each CHIL user is described by a user profile, which contains a set of personal attributes including administrative data relevant to the CHIL system (e.g. access rights, user capabilities and characteristics) and individual personality information like professional and personal interests, contacts and the social relationships between the contacts and the user (VIP, business or personal) as well as interaction, device and notification preferences such as notebook, PDA, cell phone call, SMS, MMS, targeted audio, etc. The administrative part of the user profile is maintained by the system administrator; personal data can be added and modified by the user exclusively by means of a suitable GUI.

Access to and control of the user profile is managed by the user's Personal Agent. Thus, the Personal Agent does not only operate as a personal assistant, but also as a privacy guard to both sensitive and public user data. Since the Personal Agent is the only one having access to the user's profile, it ensures user data privacy.

The Personal Agent controls, via dedicated Device Agents, the complete interaction between its user and the CHIL system: it knows what front-end devices its master has access to, how it can best receive or send information to and from its master and what input and notification types its master prefers. Furthermore, the Personal Agent communicates (using both requests and subscriptions) with the Situation Watching Agent to be permanently updated about its master's current context (location, activity, state of the environment) and the availability of the various devices in a dynamically changing situation. Based on the static data of the user profile and the dynamic context information, the Personal Agent handles user input and connection and notification requests to its master the best way and with the most appropriate media possible.

6.5 Qualitative Advantages of the CHIL Agent Framework

The benefits of the CHIL multi-agent framework are manifold. On the one hand, the architecture undertakes a wide range of tedious tasks, easing the deployment of new services, and on the other hand it provides a transparent layer to the developer in terms of information retrieval. It offers high flexibility, scalability and reusability and it facilitates the integration of components at different levels, like

- Services,
- Perceptual Components,
- Sensors and Actuators, and
- User Interfaces.

Particularly the plug-in mechanism for agent behaviours constitutes a powerful technique for the development, test, integration, configuration and deployment of new services and components. Developers may create new agents and behaviours and use this mechanism for easy behaviour integration and agent configuration, thus facilitating and accelerating the process of development and testing. They may benefit from the reusability feature of the agent framework by including own behaviours in already existing agents in order to use the functionality of these agents. And they may profit from the flexible configuration facility,

allocate behaviours to different agents, turn behaviours and even complete services on and off, in the development and test phase as well as in the deployment and integration phase.
Another important quality factor is the use of an ontology based agent communication. Elevating the collaboration of components on a semantic level does not only augment the robustness of the system in terms of mutual understanding of internal components, but also reduces the error-proneness when integrating new components and enhances the interoperability with external systems significantly. A high level of scalability is ensured by the fact that all agents can be distributed to a theoretically unlimited number of computers.
Furthermore, the described technology for service composition enhances the scalability of the CHIL agent framework in terms of functionality. Additionally, detailed guidelines for service integrators are available, which help service developers to integrate their services into the CHIL architecture framework.

7. Prototype Implementation

The CHIL agent-based infrastructure has been utilized for implementing several non-intrusive services (cf. section 2.2) which demonstrate how this framework is appropriate for developing context-aware cooperating applications. The following paragraphs present one of the implemented example scenarios.

7.1 Example Scenario

This scenario incorporates the two CHIL services Connector Service and Travel Service and a number of elementary services such as Meeting Service and Smart Room Service. Both CHIL services are integrated into the CHIL architecture using the before described plug-in mechanism.
A meeting takes place in a CHIL smart room equipped with sensors and output devices. The presence of each meeting participant is considered to be crucial for the outcome of the meeting. Hence, the meeting will be delayed, if one of the participants is late. All meeting participants are known to the CHIL system and have CHIL-enabled personal devices, i.e. notebooks, PDAs or smart phones with a CHIL software client. Most of the participants have itineraries with flights or trains leaving shortly after the scheduled end of the meeting.
One of the participants realizes that he will be late for the meeting. He uses one of the functionalities of the Connector Service by sending an ”I’m late”-notification (together with the expected arrival time) to the CHIL system from his personal device (e.g. a smart phone, see Fig. 5, left). The system then informs the other participants about the delay via the smart room devices or personal devices, dependent on the current location of the participant, user preferences and the available output media.
The Personal Agents of the other participants know the planned itineraries of their masters. Triggered by the delay message each Personal Agent determines whether the delay is likely to let its master miss his return connection. If this is the case the Personal Agent providently initiates a search for alternative connections. It provides the Travel Agent with the necessary information including user preferences, e.g. if its master prefers to fly or take a train. The Travel Agent processes the request by retrieving information from online services of railway operators, airlines and travel agencies. Eventually, it sends a list of possible connections to the Personal Agent which notifies its user. Notification is done unobtrusively taking into account the current environment situation of its master (e.g. “in meeting”), the currently

available output devices (i.e. personal devices like smart phones, PDAs, notebooks and output devices of the smart room, e.g. targeted audio or steerable video projector) and the preferred way of notification (e.g. pop-up box or voice message).

Figure 5. Smart phone version of the CHIL service access client

A possible outcome of the search could also be that the Personal Agent informs its master that he should leave the meeting as planned, since there was no suitable alternative itinerary. In case the CHIL user is not satisfied with any of the proposed itineraries or wants to look up travel connections himself, he can use his CHIL-enabled personal device to do so. Fig. 5, right, shows the query mask on a smart phone. An equivalent user front end is available for notebooks.

8. Conclusion

Developing complex sensing infrastructures, perceptual components, situation modelling components and context-aware services constitute extremely demanding research tasks. Given the tremendous effort required to setup and develop such infrastructures, we strongly believe that a framework ensuring their reusability in the scope of a range of services is of high value. This is not the case with most ubiquitous, pervasive and context-aware systems, which tend to be very tightly coupled to the underlying sensing infrastructure and middleware (Smailagic & Siewiorek, 2002; Ryan et al., 1998). It is however expedient in projects like CHIL, where a number of service developers concentrate on radically different services.

In this chapter we presented a reference architecture for context-aware services. We put the main focus on the distributed agent framework that allows developers of different services to concentrate on their service logic, while exploiting existing infrastructures for perceptual processing, information fusion, sensors and actuators control. The core concept of this framework is to decouple service logic from context-aware and sensor/actuator control middleware. Hence, service logic can be "plugged" in a specific placeholder based on well defined interfaces. The agent framework has been implemented based on the JADE environment and accordingly instantiated within real life smart rooms comprising a wide range of sensors and context-aware components. The benefits of this framework have been manifested in the development of different cooperating applications providing assistance during meetings and conferences as well as facilitating human communication.

9. References

Altmann, J.; Gruber, F.; Klug, L.; Stockner, W.; Weippl, E. (2001). Using mobile agents in real world: A survey and evaluation of agent platforms, *Proc. of the 2nd International Workshop on Infrastructure for Agents, MAS, and Scalable MAS at the 5th International Conference on Autonomous Agents,* pp. 33-39, Montreal, Canada, June 2001

AMI (2008). Augmented Multi-party Interaction (2008), Available from http://www.amiproject.org, accessed July 31, 2008

Brandstein, M.; Ward, D. (2001). *Microphone Arrays: Signal Processing Techniques and Applications*, Springer, ISBN 978-3-540-41953-2, Berlin, Germany

Chen, H.; Finin, T.; Joshi, A.; Perich, F.; Chakraborty, D.; Kagal, L. (2004). Intelligent Agents Meet the Semantic Web in Smart Spaces, *IEEE Internet Computing* 8, 6 (Nov. 2004), pp. 69-79, ISSN 1089-7801

Coen, M.; Phillips, B.; Warshawsky, N.; Weisman, L.; Peters, S.; Finin, P. (1999), Meeting the Computational Needs of Intelligent Environments: The Metaglue System, *Proceedings of MANSE'99*, pp. 201–212, Dublin, Ireland, Dec. 1999

Cost, R. S.; Labrou, Y.; and Finin, T. (2001). Coordinating agents using agent communication languages conversations. In: *Coordination of Internet agents: models, technologies, and applications*, Omicini, A.; Zambonelli, F.; Klusch, M.; Tolksdorf, R. (Eds.), pp. 183-196, Springer-Verlag, ISBN 978-3-540-41613-5, London, UK

Dey, A.K. (2000). *Providing Architectural Support for Building Context-Aware Applications*, Ph.D. dissertation, Georgia Institute of Technology, ISBN 0-493-01246-X, Atlanta, GA, USA

FIPA (2008). The Foundation for Intelligent Physical Agents, Available from http://www.fipa.org, accessed July 31, 2008

Garlan, D.; Siewiorek, D.; Smailagic, A.; Steenkiste, P. (2002). Project Aura: Towards Distraction-Free Pervasive Computing, *IEEE Pervasive Computing*, special issue on "Integrated Pervasive Computing Environments", Vol. 1, Issue 2, (April 2002), pp. 22–31, ISSN 1536-1268

Grimm, R.; Anderson, T.; Bershad, B.; Wetherall, D. (2000). A System Architecture for Pervasive Computing, *Proceedings of the 9th ACM SIGOPS European Workshop*, pp. 177–182, ISBN 1-23456-789-0, Kolding, Denmark, Sept. 2000, ACM, New York, USA

Huhns, M.N.; Singh, M.P. (1997). Agents are everywhere, *IEEE Internet Computing*, Vol. 1, Issue 1, (January 1997), pp. 87-87, ISSN 1089-7801

JADE (2008). Java Agent DEvelopment Framework, Available from http://jade.tilab.com, accessed July 31, 2008

Jennings, N.R.; Sycara, K.; Wooldridge, M. (1998). A roadmap of agent research and development. *Journal of Autonomous Agents and Multi-Agent Systems*, 1(1), (1998), pp. 7–38, ISSN 1387-2532

Johanson, B.; Fox, A.; Winograd, T. (2002). The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms, *IEEE Pervasive Computing*, Vol. 1, Issue. 2, (Apr-Jun 2002), pp. 67–74, ISSN 1536-1268

Maes, P. (1994). Agents that reduce work and information overload. *Communications of the ACM* 37, 7 (July 1994), pp. 30-40, ISSN 0001-0782

OWL (2008). W3C Semantic Web, Web Ontology Language (OWL). Available from http://www.w3.org/2004/OWL, accessed July 31, 2008

Roman, M.; Hess, C.; Cerqueira, R.; Ranganat, A.; Campbell, R. H.; Nahrstedt, K. (2002). Gaia: A Middleware Infrastructure to Enable Active Spaces, *IEEE Pervasive Computing*, Vol. 1, No. 4, (October 2002), pp. 74–83, ISSN 1536-1268

Russell, S.J.; Norvig, P. (2003). *Artificial Intelligence: A Modern Approach*. Prentice Hall, ISBN 9780137903955

Ryan, N.; Pascoe, J.; Morse, D. (1998). Enhanced reality fieldwork: the context-aware archaeological assistant. In: *Computer Applications and Quantitative Methods in Archaeology*, V. Gaffney, M. van Leusen and S. Exxon (Eds), Oxford

Shafer, S.; Krumm, J.; Brumitt, B.; Meyers, B.; Czerwinski, M.; Robbins, D. (1998). The New EasyLiving Project at Microsoft Research, *Proc. of Joint DARPA / NIST Workshop on Smart Spaces*, pp. 127–130, July 30-31, 1998, Gaithersburg, Maryland, USA

Smailagic, A.; Siewiorek, D.P. (2002). Application Design for Wearable and Context-Aware Computers. *IEEE Pervasive Computing* 1, 4 (Oct. 2002), pp. 20-29, ISSN1536-1268

Stiefelhagen, R.; Garofolo, J. (Eds.) (2007). Multimodal Technologies for Perception of Humans, *First International Evaluation Workshop on Classification of Events, Activities and Relationships*, CLEAR 2006, Southampton, UK, April 6-7, 2006, Revised Selected Papers Series: Lecture Notes in Computer Science , Vol. 4122 Sublibrary: Image Processing, Computer Vision, Pattern Recognition, and Graphics, Springer, ISBN 978-3-540-69567-7

Stiefelhagen, R.; Bowers, R.; Fiscus, J. (Eds.) (2008). *Multimodal Technologies for Perception of Humans*, International Evaluation Workshops CLEAR 2007 and RT 2007, Baltimore, MD, USA, May 8-11, 2007, *Revised Selected Papers Series: Lecture Notes in Computer Science* , Vol. 4625, Sublibrary: Image Processing, Computer Vision, Pattern Recognition, and Graphics 2008, XIII, Springer, ISBN 978-3-540-68584-5,

Ubisense (2007). The Ubisense Precise Real-time Location System. Available from http://www.ubisense.net, accessed July 31, 2008

Wellner, P.; Flynn M. (Eds.) (2005). m4 - multimodal meeting manager. Deliverable D4.3: *Report on Final Demonstrator and Evaluation*, 25 February 2005, Available from http://www.dcs.shef.ac.uk/spandh/projects/m4/publicDelivs/D4-3.pdf, accessed July 31, 2008

Advances in Human Computer Interaction

Edited by Shane Pinder

ISBN 978-953-7619-15-2
Hard cover, 600 pages
Publisher InTech
Published online 01, October, 2008
Published in print edition October, 2008

In these 34 chapters, we survey the broad disciplines that loosely inhabit the study and practice of human-computer interaction. Our authors are passionate advocates of innovative applications, novel approaches, and modern advances in this exciting and developing field. It is our wish that the reader consider not only what our authors have written and the experimentation they have described, but also the examples they have set.

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821