A Framework for Agent-based Human Interaction Support

Axel BÜRKLE, Wilmuth MÜLLER, Uwe PFIRRMANN, Manfred SCHENK
Fraunhofer Institute for Information and Data Processing
Fraunhoferstraße 1, 76131 Karlsruhe, Germany

ABSTRACT

In this paper we describe an agent-based infrastructure for multimodal 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 coding 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.

Keywords: Human Interaction Support, Sensor Infrastructure, Cognitive Architecture, Ubiquitous Services, Context-Awareness, Software Agents

1. INTRODUCTION

Services supporting human interaction should be delivered to humans in an implicit and unobtrusive way. This implies that a service is aware of the current context of its user in order to react adequately to the user's situation and environment. The CHIL (Computers in the Human Interaction Loop) project [1] [2], within which our work described in this paper was performed, develops such context-aware, implicit, and unobtrusive services. Repositioned in the background the machines discretely observe the humans and attempt to anticipate and serve their needs like electronic butlers.

The selected scenarios are situations in which people interact with people, exchange information, and collaborate to jointly solve problems. The project focuses on two scenarios: meeting and lecture rooms. They were chosen because human-human interaction plays a central role and is often critical for achieving the different goals of the participants in both scenarios. In the scope of CHIL, the following services relevant to this paper have been implemented:

- **Connector:** The Connector is a context-aware service ensuring that two parties get connected by the most appropriate media at the right time and place. Based on the observed context and each party's preferences, it decides when and how it is most appropriate and desirable for both parties to be connected.
- **Travel Service:** The Travel Service 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.
- **Memory Jog:** The Memory Jog Service provides non-obtrusive information to boost productivity in the scope of meetings, lectures and presentations in "smart rooms". The current functionality of the Memory Jog focuses on tracking humans and their activities within a room, and accordingly providing pertinent information to participants. Information is provided both when users request it, but also automatically, based on context. Information pertains to people, as well as their activities in the scope of a meeting or lecture, such as the topics, keywords, agenda items.
- **Meeting Manager:** The Meeting Manager Service handles all meeting related issues. It keeps track of organizational meeting data, e.g. the meeting location, the time schedule and the list of planned participants. 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 [1] [2] were implemented by the various CHIL partners. In addition, a series of basic services for common and elementary domain tasks are available. In order to realize the CHIL vision, an appropriate architecture and system infrastructure has been essential. Within our work we developed the architecture and especially collaborative context-aware agents which enable the implementation and provision of the specified services.

This paper is structured as follows: Section 2 addresses related research on architectures for systems supporting human interaction. Section 3 introduces the CHIL reference model architecture. Sections 4 and 5 describe the infrastructures for sensors and services. An example scenario with several cooperating services is shown in section 6. Section 7 concludes this paper with an outlook on future work.

2. RELATED WORK

The increasing distribution of mobile devices, efficient algorithms for the processing of visual and acoustic signals and the progress in the domain of semantic reasoning have promoted context-aware human interaction support as an important and worthwhile research area. In the last few years several international projects have been initiated and a lot of research work has been done to develop context-aware components as well as supporting architectures.

The *m4 project* (multimodal meeting manager) [3] presents a client-server architecture using JSP-generated frames in a meet-

ing browser to produce the output of audio and video streams and services. The *AMI Project* (Augmented Multi-Party Interaction) [4], 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 [5], on architecture frameworks for developers and administrators [6], on the design process and the development of frameworks and toolkits [7], on context-aware broker agents [8], on client-server architectures with a central server and multiple clients to support the management of multiple sensor input for different services [9], on flexible, decentralized networks connecting dynamically changing configurations of self-identifying mobile and stationary devices [10], on architectures for coordination of I/O-devices and the exploitation of contextual information [11], and on systems that span wearable, handheld, desktop and infrastructure computers [12].

The CHIL research does not only focus on context-aware technologies and services in the scope of human interaction, but also on the specification of a flexible agent-based architecture, which allows an easy integration, combination and communication on several layers of abstraction. The architecture presented in the following sections is proposed as a reference model architecture for multi-modal perceptual systems.

3. CHIL REFERENCE MODEL ARCHITECTURE

Due to the scale of the CHIL project with its large number of partners contributing to the system with a diversity of technical components such as services and perceptual components, as well as their complexity, a flexible architecture that facilitates integration of components 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. Furthermore, the architecture described here supports flexible exchange of components and the replacement of unavailable components with simulators through well-defined interfaces.

CHIL Layer Model

In order to realize proactive and intelligent services, both context-delivering and collaborating components have to be integrated. In our project, context is provided by perceptual components as well as 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 of the ontology. Collaboration is enabled by a set of intelligent software agents communicating on a semantic level with each other. The layered system architecture that facilitates the integration of different components is presented in Figure 1. 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, based on agent communication. All layers use a common ontology as a backbone. A detailed description of the layers is given below. The agent infrastructure of the CHIL system is described in more detail in section 5.

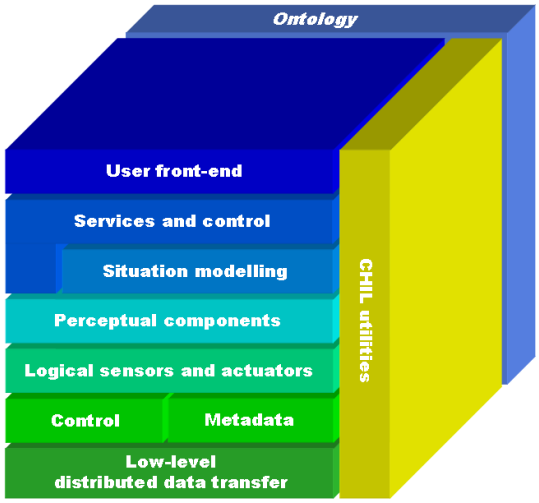


Figure 1: CHIL architecture layer model

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, smartphone) which are represented by corresponding Device Agents. The User Profile stores personal data and service-related preferences of a user.

***Services and Control*:** This layer comprises the service agents and their management. Services include reusable basic services as well as complex higher-level services composed of suitable basic services. The interaction with other agents within this layer and the *User Front-End* layer uses the communication mechanisms of the agent platform, while communication with the other layers uses internal mechanisms.

***Situation Modelling*:** The *Situation Modelling* layer is a collection of abstractions representing the environmental context in which the user acts. The context information acquired helps services to respond better to varying user activities and environmental changes. For example, the *Situation 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? An ontological knowledge-base maintains up-to-date state of objects (people, artifacts, 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*.

***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. They process the streams, interpret them, and extract information relating to people's actions such as people's locations, IDs, hand gestures, pose recognition etc.

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 modeling aspects of it. It 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.

***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.

***Control and 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 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. 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.

***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 a great number of 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 [13]. It comprises several modules that are physically represented by separate Web resources with distinct URLs, among them the fully integrated CHIL agent communication ontology. 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.

4. SENSOR INFRASTRUCTURE AND 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 [14]
- Cameras (fixed, active with pan, tilt and zoom (PTZ) or panoramic (fish-eye))
- RFID-based location trackers [15]

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 perceptual components such as multimodal speech synthesis and targeted audio.

Situation Recognition

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 complicated 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 situation 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 modeled by this network. Figure 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 situations: "MeetingSetup", "OngoingMeeting" and "MeetingFinished". Starting with the "Arrival" state of the initial "MeetingSetup" situation, the state changes to "WelcomeParticipant" when the *Person Identification* perceptual component signals that a person is entering the room; the incoming person is welcomed by displaying a message on the whiteboard and the state switches back to "Arrival". A *Meeting Detector* perceptual component retrieves and compiles location and activity information about all attendees, recognizes, in connection with its knowledge about the scheduled participants, the start of the meeting and triggers the transition to the "OngoingMeeting" situation. During the meeting, the *Body Tracker* perceptual component tracks the locations of the participants;

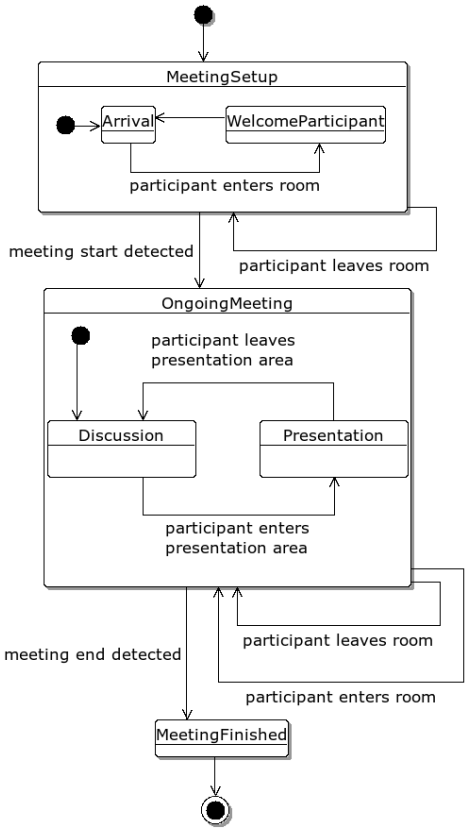


Figure 2: A network of situations model for tracking meetings in a smart room

it detects when a person approaches or leaves the presentation area and switches the state accordingly between "Discussion" and "Presentation". The Meeting Detector again determines the end of the meeting by means of the attendees' locations and activities and triggers the final "MeetingFinished" situation.

5. AGENT-BASED SERVICE INFRASTRUCTURE

Ubiquitous services are usually based on complex heterogeneous distributed systems comprising sensors, actuators, perceptual components, as well as information fusion middleware. In projects like CHIL, where a number of service developers concentrate on radically different services, it is of high value that a framework ensures reusability in the scope of a range of services. To this end, we have devised a multi-agent framework that meets the following target objectives:

- Facilitates integration of diverse context-aware services developed by different service providers.
- Facilitates services in leveraging basic services (e.g. sensor and actuator control) available within the smart rooms.
- Allows augmentation and evolution of the underlying infrastructure independently of the services installed in the room.
- Controls user access to services.
- Supports service personalization through maintaining appropriate profiles.
- Enables discovery, involvement and collaboration of services.

Agent Description

Figure 3 shows the CHIL software agent infrastructure, composed of three levels of agents: personal and device agents that are close to the user, basic agents providing elementary tasks including a communication ontology and specific service agents.

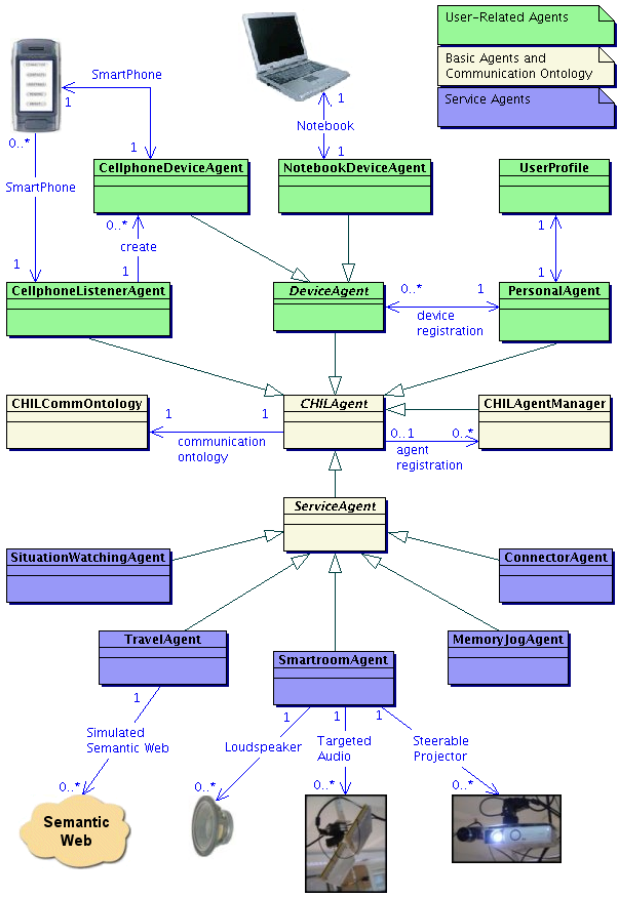


Figure 3: Agent infrastructure

User-related Agents: Every person in the CHIL environment has his own Personal Agent, that acts as a personal secretary. Users interact with the system only via their self-adapting personal agents assigned during the login procedure. The Personal Agent manages (via dedicated device agents, bound to specific devices) interactions with its master: It 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. Moreover, the personal agent communicates with the Situation Watching Agent to be updated about its master's current context (e.g. location, activity) 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.

Each device has its own Device Agent that manages communication between the device and a user's Personal Agent. The Notebook Device Agent is a concrete implementation of a Device Agent managing the communication between the graphical user front-end on the user's notebook and the user's Personal Agent. The Cellphone Device Agent is a

concrete Device Agent communicating with the user's cellphone.

The Cellphone Listener Agent watches a specific port; in case of an incoming call from a cell phone, it creates a Cellphone Device Agent and passes over the socket connection to it. The new device agent will then handle further communication between the cell phone and the user's Personal Agent.

Basic Agents and Communication Ontology: The main basic agents are the CHIL Agent and the CHIL Agent Manager. CHIL Agent is the basic abstract class for all agents used in the CHIL environment. It provides methods for essential agent administrative functionality (agent setup and takedown), for ontology-based messaging (create, send, receive messages and 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 FIPA (*Foundation for Intelligent Physical Agents*) [16] i.e. to comply with the FIPA Interaction Protocols and the FIPA Communicative Acts. The message transfer is based on a well-defined communication ontology as recommended by 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 Agent Manager; hence, the agent manager is, at any time, aware of all available agents, their abilities and the resources required by the agents for carrying out their services. The CHIL Agent Manager can act both as a matchmaker or a broker. As a matchmaker it provides a requesting agent with a handle to appropriate service agents capable of satisfying the request. As a broker, the CHIL Agent Manager decomposes the request 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 their 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, as well as by Service Agents, which may compose various functions from other services in order 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; the service itself may either be implemented as agent or be covered by an agent providing a suitable interface to the service. This applies to the Connector Agent, the Memory Jog Agent, the Travel Agent and the Meeting Manager Agent (not shown in figure 3). The functionality of these agents and the corresponding services have been specified in section 1, section 6 describes the Connector Service and the Travel Service in detail.

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 optical and acoustic output devices in the smart room. It may communicate 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"). For example it is informed via subscription to the situation model when a meeting participant has entered the meeting room, when the meeting has started and when a presenter enters or leaves the presentation area. Moreover, it can provide location information of important artifacts 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.

Personalization

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 exclusively by the user with 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 master 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.

Intelligent Messaging

In a distributed system with various cooperating services it is of high importance that the services understand each other correctly in order to successfully complete their tasks. The significance of semantic understanding is even increased by the fact that these services are implemented by a great number of developers in different places. To this end, the information exchange between agents is completely based upon a well-defined communication ontology, as proposed in the FIPA Abstract Architecture Specification [16], ensuring that the semantic content of messages is preserved across agents.

The CHIL Communication Ontology is based upon the "Simple JADE Abstract Ontology", an elementary ontology provided by JADE [17] that has to be used for every ontology-based message exchange within the JADE agent management system, and completely defined using the Web Ontology Language OWL [13]. It is fully integrated in the overall CHIL domain ontology, extending it by tokens which are necessary for agent communication, particularly agent actions for requesting services and response classes to hold the results of the services. The ontology classes are used in the Java environment by means of the JADE *abs* package, an abstract agent communication API found in *jade.core.abs*. Their handling is implemented by the basic CHIL Agent, which provides appropriate methods for ontology-based encoding, sending, receiving and decoding ACL (Agent Communication Language) messages to the agent community.

A second level of intelligent messaging is achieved by the implementation of an advanced agent coordination. Coordination of agents can be performed using well-defined conversation protocols [18]. As a first approach, we use the interaction protocols and communicative acts specified by FIPA [16]. The CHIL Agents' communication methods and initiator and responder classes for submitting and receiving messages ensure that the agent communication is strictly compliant to the FIPA specification. Together with the CHIL Agent Manger as the central agent coordinating instance, the FIPA compliance and the ontology-based message transfer form a highly sophisticated approach of Intelligent Messaging.

Pluggable Behaviors

One of the major goals of the CHIL agent infrastructure was to provide a mechanism that allows the distributed development, an easy integration and configuration of multiple services. The reason is that several complex context-aware services had to be implemented in different places by a great number of developers and had to be integrated in the CHIL system. The target framework should minimize the integration effort and allow the service developers to concentrate on the core service functionality.

A simple service can easily be integrated in the CHIL system by creating an agent which handles the framework tasks and the service control, and integrate the agent in the agent framework. 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 specific functions from other agents. For example, the Connector Service needs knowledge about the user's connection preferences (phone call, SMS, notebook, audio) and the social relationship between two or more participants (VIP, business or personal contact) to be able to establish a connection the best way and at the best time possible. Although this personal information is only used by the Connector Service, it must be controlled by the Personal Agent of each user in order to ensure user privacy. 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 using this agent would have to synchronize the agent's code. A technique like this would quickly raise significant problems concerning the coordination of the implementation and the configuration of software components.

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 behaviors 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:

Setup handler: are responsible for service specific initialization in the setup phase of an agent.

Event handler: are registered for certain events from outside the agent world, e.g. the user's GUI, a perceptual component, the situation model or a web service.

Pluggable responder: are triggered by 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 behaviors are dedicated to it and have to be instantiated, and adds them to its behavior 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.

Plug-in Support for Multiple Services

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

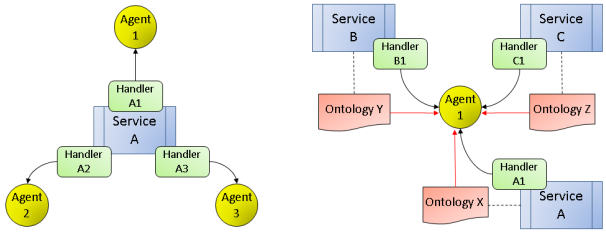


Figure 4: Pluggable handlers and scalable services

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 figure 4, right side. Hence, service specific ontologies have to be handled in the same way as Pluggable Behaviors: the framework must be capable to plug them in without the need for recompiling the agent's code. To this end, the Pluggable Behaviors mechanism has been extended to Scalable Services by realizing a plug-in technique for service specific communication ontologies.

A new service is specified and integrated into the CHIL system by means of a XML-based configuration file. The file defines the service ontology, all agents participating in the service and their handlers; the structure of the file can be seen as an XML schema definition in figure 5. Each pluggable handler is specified by its

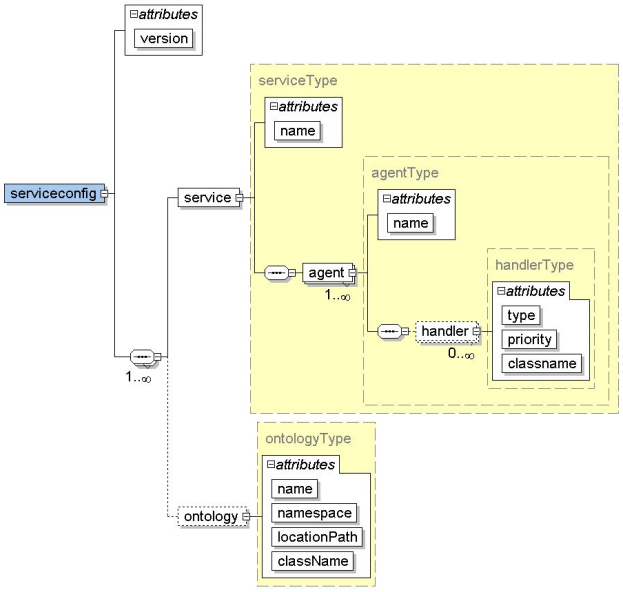


Figure 5: Service configuration XML schema

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 behaviors. 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.

Similar to a service configuration file that informs a service about all participating agents, a master configuration file informs the CHIL system about all participating services. This file is based on XML, too, and specifies the services that are activated on system start-up by their names and the location of their configuration files. Again, enabling and disabling complete services can easily be done by adding or removing the appropriate XML elements.

Advantages of the Agent Framework

The CHIL multi-agent framework has manifold advantages. 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 behaviors constitutes a powerful technique for the development, test, integration, configuration and deployment of new services and components. Developers may create new agents and behaviors and use this mechanism for easy behavior 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 behaviors in already existing agents in order to use the functionality of these agents. And they may profit from the flexible configuration facility in the development and test phase as well as in the deployment and integration phase by allocating behaviors to different agents and turning behaviors and even complete services on and off.

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.

6. PROTOTYPE APPLICATIONS

We have utilized this architectural framework for implementing three different non-intrusive applications, namely the "Travel Service", the "Connector", and the "Meeting Manager". These applications demonstrate how this agent framework is indeed appropriate for developing context-aware cooperating applications. Specifically, implementations have shown that it is possible to use this framework to interface with situation modeling logic, as well as to invoke basic services. The full value of the framework is demonstrated by plugging three services into the same instance of the framework based on the same underlying infrastructure for sensing and context-awareness. The following paragraphs present briefly the functionality of the current implementations. A fourth CHIL service not covered in this section, the Memory Jog Service, has been implemented by one of our project partners using the same reference architecture [19].

The Connector

The goal of the Connector Service is to facilitate human communication during events that occur in in-door environments. It manages intelligent communication links (connections and notifications) and handles connection and notification requests, either

simple requests (e.g. a meeting participant wants to talk to another) or more complex requests (e.g. a meeting participant wants to notify all other participants). The corresponding Connector Agent mediates between various Personal Agents and handles communications affecting two or more Personal Agents. Personal Agents may communicate with each other directly if there is only a direct communication between two Personal Agents. The Connector Service stores all pending connections and finds a suitable point of time for these connections to take place.

In order to describe the complex behavior of the Connector Service, we consider a sample scenario described in detail in [20]. In this scenario a meeting is scheduled in a CHIL smart room, where most of the meeting participants are already present in the meeting room. Another participant (Jeff) realizes that he will be late for the meeting and wants to inform the other participants about his delay.



Figure 6: Smartphone client to access multiple CHIL services

Jeff uses his smartphone (Fig. 6, left) to trigger his Personal Agent (via the smartphone's Device Agent) which informs the Agent Manager of the delay. The Agent Manager asks the Meeting Agent for the meeting room number and a list of all scheduled meeting participants to inform them in the most efficient way. With this information the Agent Manager requests the responsible Connector Agent to deliver Jeff's message in an appropriate manner. After contacting the situation model via the Situation Watching Agent, the Connector Agent (being aware of a list of participants already in the meeting room) decides to notify them via the Smart Room Agent, which is aware of the output devices available in the meeting room. The Smart Room Agent chooses to display the delay message with the central video projector. For those participants who are not yet in the meeting room, the Connector Agent informs their Personal Agents which in turn may choose an appropriate output medium for their masters (e.g. notebook, PDA, mobile phone, etc.) according to their current situation and user profile. Finally Jeff is informed again by his Personal Agent that his message has been delivered to all meeting participants.

The Travel Service

The Travel Service provides assistance with planning and rearranging itineraries. It can either be evoked directly by the user through one of his personal devices (cf. Fig. 6, right) or triggered by another CHIL service. An example scenario has been implemented which demonstrates close cooperation between the Connector Service, the Travel Service and the Meeting Manager Service as well as a number of elementary services such as Meeting Service and Smart Room Service. All CHIL services are integrated into the CHIL reference architecture using the plug-in mechanism described in section 5.

This scenario follows up the Connector Service scenario where one participant (Jeff) is late for a meeting. The begin of the meeting will be delayed until Jeff arrives. The Personal Agents of the other participants know the planned itineraries of their masters. Triggered by Jeff's 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 (currently simulated) semantic web services of rail way 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 smartphones, 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). A possible outcome of the search could also be that the Personal Agent has to inform its master that he should leave the meeting as planned, since there is no suitable alternative itinerary.

If the CHIL user is not satisfied with any of the proposed itineraries or wants to look up travel connections by himself, he can use his CHIL-enabled personal device to do so. Figure 6, right, shows the query mask on a smartphone. An equivalent user front end is also available for other personal devices.

The Meeting Manager Service

The Meeting Manager provides assistance with organizing and running a meeting. The Meeting Manager keeps track with the people planned to attend a meeting and those already arrived in the meeting room. For keeping track with the time schedules and meeting room schedules, it invokes the elementary service Meeting Service. When a meeting participant enters the meeting room, it welcomes the participant in a personalized way. As soon as all planned participants are in the meeting room, the Meeting Manager automatically starts the meeting by displaying the agenda.

In the case that a participant is late, and following up the above described scenario "Jeff is late", the Meeting Manager is informed by the Connector Agent about the delay. It then decides, depending on the amount of the delay and the other meetings scheduled in the meeting room, if the meeting should start or if it should be delayed until Jeff arrives.

After having started the meeting, the Meeting Manager is informed by the Situation Watching Agent if a presenter enters the presentation area. It then automatically starts the presentation planned to be given by the presenter. Additionally, it keeps track

of the agenda and marks the agenda topics already discussed as well as those which are still open. At the end of the meeting, which is again detected via cooperation with the Situation Watching Agent, it displays the farewell message on the screen.

7. CONCLUSION

In this paper we have presented a reference model architecture for multi-model perceptual systems. Main focus was put on a distributed agent framework allowing 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 [17], and accordingly instantiated within two real life smart rooms comprising a wide range of sensors and context-aware middleware components. The benefits of this framework have been manifested in the development of two cooperating applications, one providing assistance during meetings and conferences, and another facilitating human communication.

As future work we will provide a set of tools to facilitate the development and integration of services, e.g. by the automatic generation of pluggable handler skeletons. We are also working on a semantic service description, which will be part of the common CHIL ontology. Raising service specification onto a semantic level will allow dynamic service discovery and integration and a more sophisticated service composition by the use of an inference engine and rules from the ontological knowledge base.

ACKNOWLEDGEMENTS

This work is part of the project CHIL, an Integrated Project (IP 506909), partially funded by the European Commission under the Information Society Technology (IST) program. The authors acknowledge valuable help and contributions from all partners of the project.

REFERENCES

[1] R. Stiefelhagen, H. Steusloff, and A. Waibel, "CHIL - Computers in the Human Interaction Loop," in **5th International Workshop on Image Analysis for Multimedia Interactive Services**, Lisbon, Portugal, Apr. 2004.

[2] CHIL - Computers in the Human Interaction Loop. [Online]. Available: http://chil.server.de/

[3] m4 - multimodal meeting manager. [Online]. Available: http://www.m4project.org

[4] AMI: Augmented Multi-party Interaction. [Online]. Available: http://www.amiproject.org

[5] M. Roman, C. Hess, R. Cerqueira, A. Ranganat, R. H. Campbell, and K. Nahrstedt, "Gaia: A Middleware Infrastructure to Enable Active Spaces," **IEEE Pervasive Computing**, vol. 1, no. 4, pp. 74–83, Oct.-Dec. 2002.

[6] R. Grimm, T. Anderson, B. Bershad, and D. Wetherall, "A System Architecture for Pervasive Computing," in **Proceedings of the 9th ACM SIGOPS European Workshop**, Kolding, Denmark, Sept. 2000, pp. 177–182.

[7] A. K. Dey, "Providing Architectural Support for Building Context-Aware Applications," Ph.D. dissertation, Georgia Institute of Technology, Nov. 2000.

[8] H. Chen, T. Finin, A. Joshi, F. Perich, D. Chakraborty, and L. Kagal, "Intelligent Agents Meet the Semantic Web in Smart Spaces," **IEEE Internet Computing**, vol. 8, no. 6, Nov. 2004.

[9] B. Johanson, A. Fox, and T. Winograd, "The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms," **IEEE Pervasive Computing**, vol. 01, no. 2, pp. 67–74, 2002.

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

[11] S. Shafer, J. Krumm, B. Brumitt, B. Meyers, M. Czerwinski, and D. Robbins, "The New EasyLiving Project at Microsoft Research," in **Proc. of Joint DARPA / NIST Workshop on Smart Spaces**, July 30-31 1998, pp. 127–130.

[12] D. Garlan, D. Siewiorek, A. Smailagic, and P. Steenkiste, "Project Aura: Towards Distraction-Free Pervasive Computing," **IEEE Pervasive Computing, special issue on "Integrated Pervasive Computing Environments"**, vol. 1, no. 2, pp. 22–31, Apr.-June 2002.

[13] W3C Semantic Web, Web Ontology Language (OWL). [Online]. Available: http://www.w3.org/2004/OWL

[14] M. Brandstein and D. Ward, **Microphone Arrays: Techniques and Applications**. New York: Springer-Verlag, 2001.

[15] Ubisense. [Online]. Available: http://www.ubisense.net

[16] The Foundation for Intelligent Physical Agents. [Online]. Available: http://www.fipa.org

[17] Java Agent DEvelopent Framework. [Online]. Available: http://jade.tilab.com

[18] R. S. Cost, Y. Labrou, and T. Finin, "Coordinating Agents using Agent Communication Languages Conversations," in **Coordination of Internet Agents: Models, Technologies, and Applications**, A. Omicini, F. Zambonelli, M. Klusch, and R. Tolksdorf, Eds. Springer, Mar. 2001, ch. 7, pp. 183–196.

[19] N. Dimakis, J. Soldatos, L. Polymenakos, M. Schenk, U. Pfirrmann, and A. Bürkle, "Perceptive Middleware and Intelligent Agents Enhancing Service Autonomy in Smart Spaces," in **IEEE/WIC/ACM International Conference on Intelligent Agent Technology (IAT-06)**, Hong Kong, Dec. 2006.

[20] M. Danninger, G. Flaherty, K. Bernardin, H. Ekenel, T. Köhler, R. Malkin, R. Stiefelhagen, and A. Waibel, "The connector: facilitating context-aware communication," in **7th Int. Conference on Multimodal Interfaces (ICMI'05)**. New York, USA: ACM Press, 2005.