Companion-Systems: A Reference Architecture

Thilo Hörnle, Michael Tornow, Frank Honold, Reinhard Schwegler,
Ralph Heinemann, Susanne Biundo, and Andreas Wendemuth

Abstract *Companion*-Technology for cognitive technical systems consists of a multitude of components that implement different properties. A primary point is the architecture which is responsible for the interoperability of all components. It defines the capabilities of the systems crucially. For research concerning the requirements and effects of the architecture, several demonstration scenarios were developed. Each of these demonstration scenarios focuses on some aspects of a *Companion*-System. For the implementation a middleware concept was used, having the capability to realize the major part of the *Companion*-Systems. Currently the system architecture takes up only a minor property in projects which are working on related research topics. For the description of an architecture representing the major part of possible *Companion*-Systems, the demonstration scenarios are studied, regarding their system structure and the constituting components. A monolithic architecture enables a simple system design and fast direct connections between the components, such as: sensors with their processing and fusion components, knowledge bases, planning components, dialog systems and interaction components. Herein, only a limited number of possible *Companion*-Systems can be represented. In a principled approach, a dynamic architecture, capable of including new components during run time, is able to represent almost all *Companion*-Systems. Furthermore an approach for enhancing the architecture is introduced.

Thilo Hörnle · Reinhard Schwegler · Susanne Biundo
Institute of Artificial Intelligence, Ulm University, 89069 Ulm, Germany
e-mail: thilo.hoernle@uni-ulm.de, reinhard.schwegler@uni-ulm.de, susanne.biundo@uni-ulm.de

Michael Tornow · Ralph Heinemann · Andreas Wendemuth
Institute of Information Technology and Communications, Otto-von-Guericke University, 39016 Magdeburg, Germany
e-mail: michael.tornow@ovgu.de, ralph.heinemann@ovgu.de, andreas.wendemuth@ovgu.de

Frank Honold
Institute of Media Informatics, Ulm University, 89069 Ulm, Germany
e-mail: frank.honold@uni-ulm.de

1 Introduction

Future technical systems will use *Companion*-Technology to increase their usability and extend their functionality. By adding the feasibility to adapt specific system components to the individual user in any given context of use, the system could increase its efficiency and cooperativeness by assisting the user with his everyday tasks. *Companion*-Systems should be highly available in an individual way and they should always act on behalf of their users, striving for a maximum degree of reliability (cf. Chap. 1. By these properties *Companion*-Systems differ from contemporary technical systems. To offer these properties, some new capabilities need to be implemented as shown in Fig. 1. In that way, existing applications, services, or technical systems can be wrapped by specialized modules to form a *Companion*-System. The required modules from Fig. 1 can be classified as follows:

Recognition: These modules encompass diverse sensors for emotion and intention recognition. In addition, ongoing changes in the environment are also recognized to allow adaptive reasoning and responses of other modules.

Knowledge: A central knowledge base (KB) infers and provides knowledge based on the recognizers' inputs. Since sensor data may be afflicted with uncertainty, the KB provides probability distributions along with its requested values.

Planning, Reasoning, Decision: The process of adaptive output generation decides about task planning, dialog strategy and user interface (UI) configuration. Each involved reasoning process infers its results with the use of the knowledge as provided by the KB.

Interaction: Modules for adaptive dialog and interaction management along with different input and output devices are responsible for providing a suitable UI. With that, the user can perform either implicit or explicit interaction.

A *Companion*-System is a kind of evolution of cognitive technical systems, as it combines miscellaneous cognitive properties implemented in different modules. The application itself as well as the linked process of human-computer interaction (HCI) needs a perception of the emotion and the intention of the user as well as of the situation. According to Knappmeyer et al. [15], context data can be collected via physical sensors, logical sensors, virtual sensors, and user profiles. A context-provisioning middleware can be used to analyze and aggregate such data. Decision processes of application and service logic are able to access the so aggregated data (ibid.). *Companion*-Technology utilizes all these sensor classes to comprehend the current situation of the user and the environment. This knowledge in combination with the system's internal state is what we call the context of use (CoU). In our knowledge-centered approach, the KB acts as the one provider for any information that is related to the CoU.

To gain an easy access on the complex sensor data from the different sensors, hierarchical fusion concepts for combining data are necessary. Fusion can be performed on different levels [5, 6]. Based on Caschera et al. [5], this chapter addresses fusion approaches on the data-level and feature-level (also known as early and late or semantic fusion). Fusion on the application-level as mentioned by Glodek et al.

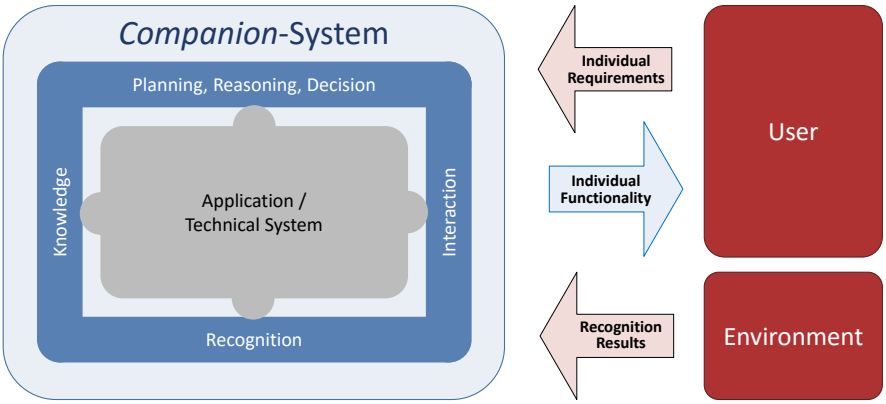

Fig. 1 Abstract integration of the *Companion*-Technology in a technical system to generate a specific *Companion*-System (cf. [4]). A basic technical system is wrapped by different aspects of *Companion*-Technology. The added concepts relate to the four areas of recognition, knowledge acquisition, planning and reasoning, as well as interaction.

[6] takes place in diverse modules (application, planning, reasoning, decision) and does not have a direct impact on the architecture.

The range of applications in the domain of *Companion*-Systems is multifaceted. In order to cover a variety of possible applications we focus on the following three exemplary settings as possible implementations for *Companion*-Systems: (1) an adaptive *Wiring Assistant* for setting up a home cinema system (see Chap. 24), (2) an adaptive and emotion-aware *Ticket Vending Machine* (see Chap. 25) and (3) a multisensor setup for data acquisition (see Sect. 3.2).

The first setting realizes a cooperative system, where the focus is on the adaptivity of modules for planning, dialog management and multimodal interaction management. Thereby, the process of output generation is realized as a pipeline as described by Honold et al. [9]. A central KB provides the involved modules with knowledge required for reasoning. The second setting focuses on multimodal data-acquisition and includes fusion concepts on different levels. This setting makes use of diverse sensors and addresses fusion of temporal and spatial events. The third setting tackles the challenge of almost real-time multimodal data acquisition. Hence, this setting requires an infrastructure that allows exact timing and fast processing under conditions of chronological synchronism.

The remainder of this chapter analyzes the requirements of these three heterogeneous exemplary settings with regard to the architecture in order to come up with a dynamic reference architecture for future *Companion*-Systems.

2 Requirements and Middleware

Evolving a technical system to a *Companion*-System requires some additional functionalities. These functionalities originate from different modules and research domains. To facilitate interdisciplinary research and to ensure flexibility, *Companion*-Technology with inter-modular dependencies should not be realized in a monolithic manner. Spreading desired functionality over several modules in a distributed system comes with further advantages. In that way, modules can be shared and exchanged to increase availability and maintainability.

To handle the complexity that results from the interplay of different modules and various sensors, we recommend the use of a dedicated middleware concept for inter-process communication within such a distributed system. In addition, the employed middleware shall facilitate the use of different operation systems and programming languages, and shall further allow offering application programming interfaces (APIs) for any module.

The following Sect. 2.1 focuses on the impact of derived requirements from *Companion*-Technology on the architecture. The use of a suitable middleware concept is discussed in the subsequent Sect. 2.2.

2.1 Impact of Requirements on the Architecture

A *Companion*-System's architecture encompasses different modules, which add desired properties to some basic technical system or service. In that way, applied *Companion*-Technology shall originate, add, and combine characteristics like "competence, individuality, adaptability, availability, cooperativeness and trustworthiness" [4]. To realize these characteristics, a basic application can be wrapped by several different *Companion*-Technology components, as illustrated in Fig. 1. Some of these modules lead to specific requirements for the architecture to fulfill their function (e. g. the availability of linked modules of reference, or a minimum infrastructural data throughput with respect to a desired quality of service). Therefore a complex and inter-component architecture evolves, whereas every subsystem inherits its own architecture. For easy maintenance, the use of coherent concepts is an important aspect; furthermore, privacy and security issues have to be covered by the architectural approach as well.

One major aspect of *Companion*-Technology is user-adaptive behavior. For this reason, a user-specific KB is necessary to support the involved reasoning processes and the application. Only with sufficient knowledge about the user, a technical system is able to act and decide as a true *Companion* in a satisfying and user-intended way. Although eligible, this property often raises concerns of privacy, reliability, and trustworthiness. From this follows that the architectural communication approach shall at least support an encryption mechanism to ensure secure inter-process communication. Another reasonable approach might be to virtually split the KB and to swap user-related data to one of the user's personal devices (e. g. his smartphone). A

third aspect addresses the process of output generation to ensure a UI that respects specific privacy guidelines. This could be realized by utilizing an additional *privacy decision engine*, as motivated by Schaub [23].

Besides the mentioned modules, the applied architectural concept itself could help to realize the aforementioned characteristics. Most architectural approaches so far are rather rigid. Once a system is set-up, the architecture remains static without any changes during runtime. In that regard, an architecture-specific managing component could help to dynamically configure the architecture's topology on-demand, even during runtime. Such an ability would lead to a dynamic, knowledge-based architecture, which allows to individually integrate required and available modules.

The analysis of the *Wiring Assistant* setting (see Chap. 24 and [2, 3, 9]) shows that several modules act as processing pipeline (e. g. output generation via planner, dialog management (DM), fission, and interface components). Despite this dependency, all of these components refer to the central KB as a host for necessary data. If one of these modules stops working, the whole process chain is affected by that. To meet the requirement of continuous availability, the architecture shall provide two additional mechanisms. First, it shall allow to detect the status of each participating module, and second, it shall offer a possibility to dynamically add and remove modules to the overall architecture. With such adaptability, envisaged but stalled modules can be replaced by other available modules that offer the same functionality. This could be realized for the architecture's modules in the same way as already implemented with the available input and output devices, as applied in the *Wiring Assistant* setting. There, the fission module adaptively decides about the configuration of the utilized end devices.

A system's competence depends on its knowledge. *Companion*-Systems shall be able to gain and infer necessary knowledge to some extend by means of recognition and cognition. This requires different sensors and fusion mechanisms to derive and extract desired information for further reasoning. Thereby, the architecture shall allow to easily integrate all kind of sensors; even dynamically at runtime. This concept of a *Self-Organizing Sensor Network* is explained more in detail in Sect. 4.

Knowledge about the time of data acquisition is of great importance for the semantically correct fusion of such sensor data [5]. Fusion of old data might result in misinterpretation. There are several methods for taking care of timing, e. g. by synchronizing data acquisition and data processing or by adding time stamps to recorded and processed data. Both concepts support time-related processing of data within the fusion process. *Companion*-Systems require data fusion on different levels e. g. raw sensor data, semantic features or logical data (see Sect. 3.1 and 3.3).

The system needs the ability to be customized to the actual application by configuring a set of required modules. Due to the number of components and their function-specific requirements, it is hardly possible to implement a *Companion*-System in a monolithic way. Since the components are implemented as independent modules, the overall system needs a standardized way of communication, preferable via network. This is the topic of the next section.

2.2 Middleware

Companion-Technology can be implemented in different ways, using various technologies. According to Knappmeyer et al. [15], a middleware concept can be used to "overcome problems of heterogeneity and hide complexity". As almost every device can handle a network communication, this would be a good and simple opportunity to connect all components to each other, whether they operate on the same or on different computers. To realize a flexible and extensible middleware a client server protocol would be best. Furthermore this central server component would be able to record all the communication for the debugging process, and take care that the communication is still working if one component is disabled.

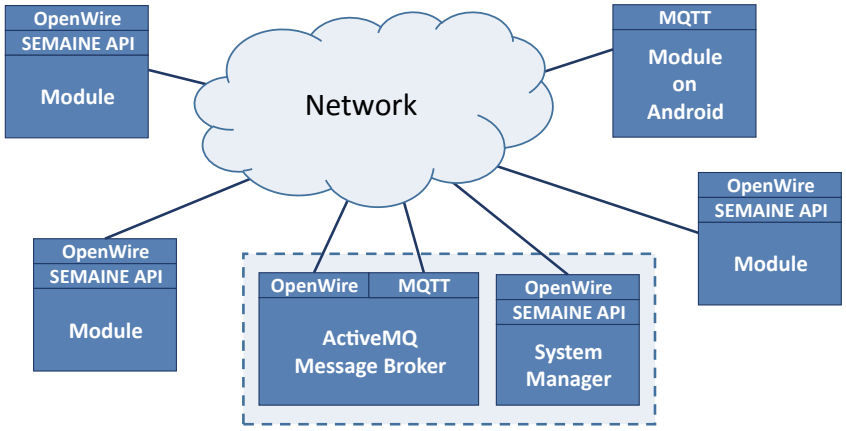

Fig. 2 *SEMAINE* Middleware Adoption

For reducing development effort, the middleware from the message broker system-based *SEMAINE* project [24] was used and adapted. It is set up using the well known and tested ActiveMQ[1] server from the Apache foundation utilizing the OpenWire[2] communication protocol. This communication protocol is an improved version of the Java Messaging Service (JMS), which also offers implementations for programming languages beyond Java. The OpenWire protocol uses topics or direct connections between the communicating components. Furthermore, the *SEMAINE* implementation offers password-protected communication to ensure privacy.

As shown in Fig. 2, the *SEMAINE* middleware is another layer between the OpenWire protocol and the application layer. This allows abstraction and simpli-

[1] Apache ActiveMQ: An open source messaging and integration patterns server – `http://activemq.apache.org` [accessed: 2015-12-18]

[2] Apache ActiveMQ – OpenWire: A cross language wire protocol – `http://activemq.apache.org/openwire.html` [accessed: 2015-12-18]

fies the implementation of modules on the application layer. The communication is organized by the so-called *System Manager*, which is a Java application that offers visualization concepts of the distributed system's topology, the senders and receivers of messages along with the topics, as well as the sent messages. The UI of this singleton-like central instance can support debugging processes and can perform on-demand logging of communication items and communication-specific events.

To support the integration of small ubiquitous sensors other message broker concepts have to be taken into account. Therefore we added an adapter for the MQTT-protocol[3]. This can also be used to connect android devices as described by Glodek et al. [6].

3 Deriving System Requirements from Prototypical Systems

When working on *Companion*-Technology, the problem of finding suitable architectures for those systems comes along. Several prototypical systems were developed for researching the structure of *Companion*-Systems. In this section some of the prototypical systems will be analyzed to gain information about the general problems of architectures and *Companion*-Systems. The topics of the prototypical systems are following the data flow starting with the sensors and data fusion and finishing with situation- and user-adaptive functionality. For developing and demonstrating the sensor setup and the data fusion of sensor and input signals (cf. Fig. 3) a prototypical system was set up, supporting the user in the task of using a *Ticket Vending Machine* as described in Chap. 25. The architecture of this system includes sensors, a multi level data fusion, and the application. While the actual architecture of this system is covered in Sect. 3.3, a general overview is given in Sects. 3.1 and 3.2. Sect. 3.5 describes the second analyzed system. It is an assistance system, which helps a user to set up a home theater. It utilizes a KB and a planner to resolve problems for the user. Based on that, the system uses modules for dialog and interaction management to guide the use to the solution of said problems. Thus, both systems cover different parts of the described *Companion*-Technology. The applied concepts are compatible to each other and need to be combined according to the requirements. Both described scenarios were set up using the middleware as described in Sect. 2.2.

3.1 Sensor Setup and Data Fusion

This section focuses on the general definitions of sensor signal acquisition and sensor data fusion to give a short introduction on the topic. In complex systems like *Companion*-Systems, there are many layers between the physical sensor and the ac-

[3] Message Queue Telemetry Transport (MQTT) is a machine-to-machine connectivity protocol for the internet of things – `http://www.mqtt.org/` [accessed: 2015-12-18]

tual application, as only in very simple systems the application is directly connected to physical sensors.

There are various publications with differing definitions dealing with sensor setup and data fusion. Knappmeyer et al. [15] distinguish the categories from sensors between physical sensors, which are in contact with the environment; virtual sensors, results of software-based algorithms, and logical sensors, like calender data.

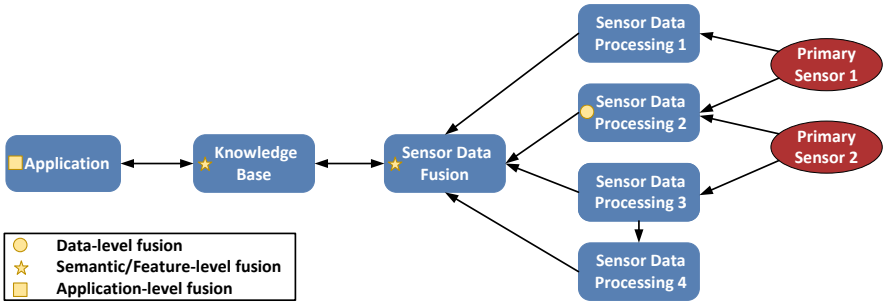

Fig. 3 A potential instance of a basic architecture of sensor data fusion for *Companion*-Systems.

Fig. 3 shows the basic sensor setup and data fusion structure from a sensory perspective. In comparison to the architecture proposed by Knappmeyer et al. [15], the physical sensors are called primary sensors since they deliver direct and unprocessed information of the environment. A primary sensor is an electronic device which records data from the environment and converts it to a machine-processable digital representation (raw data). Thus the primary sensors "Primary Sensor 1" and "Primary Sensor 2" in Fig. 3 just record data and send it to linked sensor data processing components.

The sensor data processing is realized in software, extracting relevant information from raw data, provided by the primary sensor(s). In that manner, data is transferred to a more abstract or symbolic representation. Primary sensors and sensor data processing modules constitute virtual sensors (cf. Sec. 1), similar to the definition given by Knappmeyer et al. [15]. Virtual sensors emit the result of simple sensor data processing, as well as data-level fusion processes.

A virtual sensor deals with data of a primary sensor, which can also be located in a mobile device. The virtual sensor's processing probably needs to be run on a computing server due to low computing power of most mobile devices. This case induces an additional communication layer between the primary sensor and the sensor data processing, which needs to be established. The data communication management is covered by a *Self-Organizing Sensor Network*, which can change and control the sensor setup under runtime conditions (see Sect. 3.4).

Several virtual sensors can share a primary sensor as exemplarily shown in Fig. 3. Furthermore a virtual sensor can rely on data from sensor data processings; e. g.

emotion recognition from speech [26] needs knowledge about the structure of the spoken sentence, thus the speech recognition needs to be run first.

Data fusion for *Companion*-Systems [6] is a major research topic of the research for the *Companion*-Technology. A wide range of fusion methods on different data abstraction levels was developed in recent years. From the architectural point of view mainly the fusion levels are relevant, as various kinds of data can be processed in the data fusion for *Companion*-Systems. In their surveys on multimodal fusion methods Atrey et al. [1] as well as Caschera et al. [5] give an overview on the fusion levels as well as the fusion methods. They differentiate feature- or recognition- level fusion, so called early fusion, which is performed on a very low level of abstraction. The decision-level fusion also known as late fusion (which processes abstract data) results e. g. from sensor data processings (ibid.).

The different levels of data fusion that are relevant for *Companion*-Systems are discussed by Glodek et al. [6]. The different fusion concepts are shown in Fig. 3 and Fig. 6. The authors distinguish between perception-level and data-level fusion, the KB- or semantic-level fusion and additionally the application-level fusion. The first level is the closest to the sensor data and is therefore dealing with raw or only slightly processed data and complies with the feature-level fusion (cf. [1]). Fusion on the feature level is usually performed in a separate fusion module, but can be required in sensor data processing as well (see Fig. 3: *Sensor Data Processing 2*). The data is represented on a semantic level after the sensor data processings. Hence, the fusion algorithms have to deal with more complex data.

The classes recognized on the perception level of a knowledge-based data fusion module are enriched with models created by human experts aiming at a more abstract or contradiction-free representation of data. Furthermore the temporal variation and context information are taken into account on this fusion level. On the KB level, this data is combined with context information in a second semantic data fusion, which provides information to the application. Application-level fusion is directly involved in the decision making process and combines abstract information from multiple sources as the explicit user input, the KB, the DM, and the actual application.

The primary sensors can share trigger signals for synchronization to take care of relying on a common time base (see Sect. 3.2). Thus the data fusion processing can be restricted to a certain time slot.

3.2 Concepts of Sensor Synchronization

In cases where many sensors are analyzed, it is necessary to ensure that all sensors' data ground on the same time base. The common methods to realize a synchronous recording are using a hard-wired synchronization signal, embedding a time code, or using the network time for distributed systems. Due to the wide range of used sensors, embedding the same time code in every recorded sample is hard to realize. The hard-wired synchronization signal and the network time base were tested

to gain their potential for *Companion*-Systems. From the synchronization point of view digital sensors can be divided in value-based and frame-based devices. Value-based sensors use only one recording clock controlling the sensor data acquisition, which is usually the case for audio signal acquisition or biophysical data. Data of frame-based sensors enclose the basic raw data items plus a data structure, e. g. RGB cameras or laser rangefinders. In the latter case, the sensor synchronization can be realized by synchronizing to the frame clock or to the raw data clock.

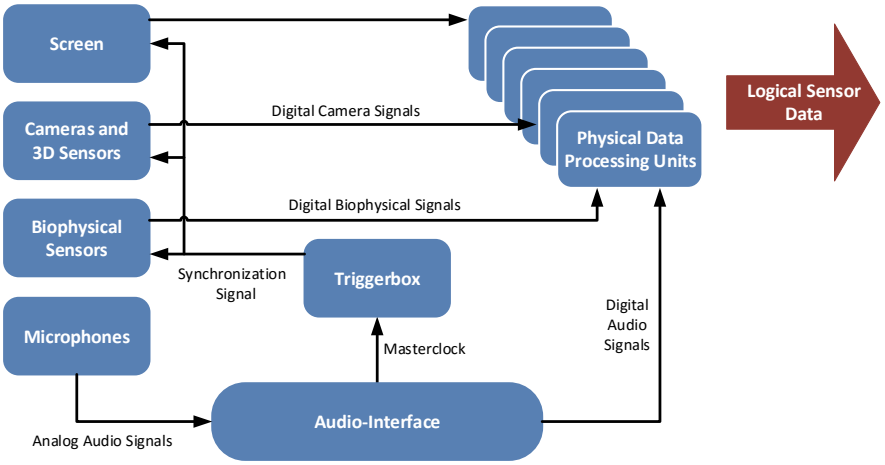

Fig. 4 Sensor Synchronization Structure

The common approach for laboratory conditions is the hard-wired synchronization signal. A structure for a multimodal synchronization method as shown in Fig. 4 was developed and tested for a group of psychological experiments, e. g. for the Last Minute experiment (see Rösner et al. [22]). Such experiments focus on the psychological aspects of HCI. The aim is to build validated classifiers to gain the emotional state or disposition of the user from the different modalities. The synchronization method is set up by an asynchronous communication protocol and a hard-wired synchronous clock. In these experiments many modalities, namely four high resolution video cameras, two 3D cameras, several sensors for biophysical data measurement (cf. [14]), and several audio channels are used. Furthermore the screen of the user is recorded for analyzing the user's reactions and the run of the experiment.

In these experiments the fastest recording clock was used as a master clock. As the used cameras were working with an internal pixel clock the camera synchronization was frame-based. Thus, the 44.1 kHz audio sampling rate of the eight line in/out audio interface Yamaha Steinberg MR-X816 was used as master clock. In recent experiments the Yamaha 01V96i was used as audio interface to improve the recorded audio quality and set the number of lost audio samples. Using this clock, the trigger signal for the cameras and stereo cameras is derived using the computer-controlled

timer counter module (Measurement Computing USB4303) by dividing it by 1764 (cf. *Triggerbox* in Fig. 4). The resulting 25 Hz signal is fed into the trigger input of those cameras and results in images, which are synchronous to the audio clock.

This method cannot be used directly for the biophysical measurement system "Nexus32" and the screen recorder. Both devices use built-in clocks and offer no opportunity to feed them with an external signal. The Nexus32 is able to record a signal that is derived from the audio clock as well, thus its data can be reconstructed close to the desired synchronous state. The screen recorder is recording with the clock of the graphic card, which cannot be fed by any other signal. The only opportunity here is to record an audio stream with the computer.

Due to the required computing power for recording the different modalities, the recording system is set up as a distributed system, using an asynchronous communication of a topic-based message broker system. The different programs can be divided into the recorder of the different modalities and the management system. The management system is able to control the available recorders and start a recording with a specified name. By starting the recording process, all recorders are set in an armed state via the asynchronous message broker system. To start the synchronous trigger signal it sends a code word via USB to the USB4303 device. Along with all audio signals, the reference/trigger signal gets recorded and enables the synchronization of visual, 3D, and audio channels.

The advantage of this system is that all signals, which are recorded with a sampling rate, derived from the base clock are perfectly aligned. The technical overhead for a compact sensor phalanx covering multiple sensor principles is acceptable, but is getting inefficient for a large sensor array distributed over a wide area.

The setup is flexible by design, but only under laboratory conditions. For a real-world scenario it is a rather static and monolithic approach by requiring hard-wired sensor setups and it synchronizes only the acquisition of a data recording. By synchronizing the sensor data processing, the effort will pay off, but the effort for a distributed system is very high. There are only very limited possibilities to correct the missing or misaligned data in an online processing. Thus, this synchronization method is mainly capable for recording of a dataset, which will be processed offline.

A middleware-oriented synchronization method is used in recording settings, in which a flexible setup with sensors without external inputs was required. The synchronization is realized by using *SEMAINE* network time. This enables the synchronous recording of sensors like webcams, Kinect etc., which does not allow hard-wired synchronization. Due to the network delay, the accuracy is limited to the range of milliseconds, which is appropriate for HCI interactions [25]. Advanced HCI capabilities are one of the main objectives of *Companion*-Systems, hence the system needs to meet at least weak real-time conditions.

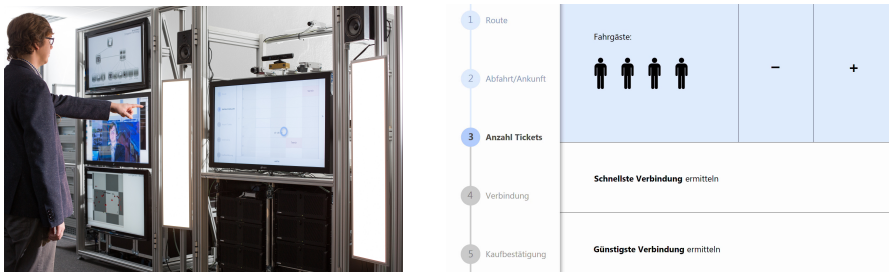

Fig. 5 Setup and application of the prototype system for multimodal sensor data acquisition, processing, and fusion.

3.3 Architecture Aspects for Sensor Data Fusion

The setting with the *Ticket Vending Machine* (see Fig. 5 and Chap. 25) demonstrates an interactive HCI system, designed to research the cooperation of different subsystems for multimodal sensor data acquisition, processing and fusion, as well as the resulting benefit for the user interaction and the application. Therefore the primary sensors' readings are analyzed online using the developed algorithms in the sensor data processing (e. g. feature extraction and classification). Based on that, the results are combined in the hierarchical data fusion structure, as shown in Fig. 6 (cf. [25]).

The sensor phalanx of the system consists of a high resolution camera (AVT Pike F145C) for mimic analysis [18, 19], a Point Grey Bumblebee 2 stereo camera attached to a pose detection [17], a RGB-D camera (Kinect V1) for gesture recognition [8], a laser rangefinder array containing two SIC LD-MRS 4-layer laser scanners for environment recognition [21], and a head-mounted radio microphone (Sennheiser HSP2 via EW100) for speech and disposition recognition [7, 27]. In the following, only the virtual sensor results will be used, as the physical sensors are directly connected to their sensor data processing components. Those virtual sensor results are sent via the *SEMAINE* message broker system to different data fusion processes.

The levels of data fusion introduced in Sect. 3.1, are implemented in separate components with different requirements regarding to the data processings. The application-level fusion is localized in the input fusion component, which is reading the smart input modalities directly, e. g. the touch events, the speech and gesture recognition results. Furthermore it processes the results of the emotional fusion, which analyzes the state of the user and the environment fusion, which processes all data available regarding the surroundings.

The fusion of emotion-related sensor data uses mainly mimic and prosody for its decisions thus the system is able to determine the emotional state of the user. Hence the HCI can react properly if the user is happy, neutral, or unhappy. Information from the sensors is transformed from raw data to a complex semantic level by the preceding fusion levels and is combined in the input fusion component.

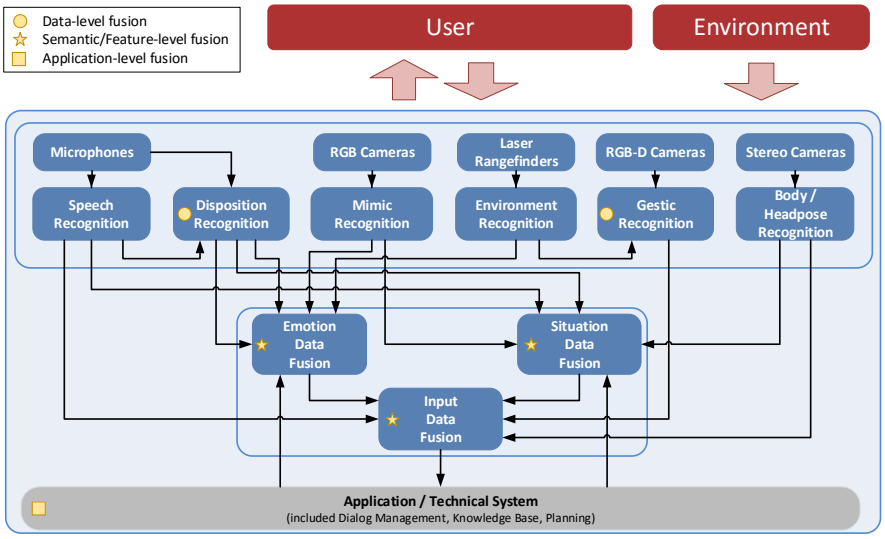

Fig. 6 Architecture of the system for multimodal sensor data acquisition, processing, and fusion.

As the environmental fusion is being fed by the laser rangefinders, the pose and gestures recognizers gain information to support the task of the user. In that way, an approaching person initializes the system, which responds with a greeting. According to the user's approach velocity, the system decides to start in either *standard* or *compact* dialog mode. Due to reduced complexity, the compact mode is more suitable for a precipitant user. Furthermore, the environment fusion determines the probable number of persons traveling with the user via the laser scanners and the RGB-D camera. The body pose gained from the stereo camera and the Kinect provides important information to the HCI, whether the user is currently corresponding to the system or someone else. Therefore the pose and gesture recognition data is used to extract information if the user is looking towards the system or not, or if he is probably on the phone. This information is used to minimize the misconception of the speech recognition. The information transferred to the application represents the essence of the complete multimodal sensor phalanx. The environmental fusion results are used as an additional input for the subsequent input fusion component. The results of the emotional fusion are functioning as a control input for the HCI and allows implicit interaction. Due to the data fusion, the system can cover a missing sensor and keep up the overall functionality.

All components are connected indirectly via the message broker system *SEMAINE*, controlled by a central management system, which is able to start up all required software components of the distributed system. All logical sensor data is processed online and the results are immediately sent to one of the fusion processes, but not directly to the application. The acquisition of all sensors apart from the Kinect is synchronized using the hard-wired method described in Sect. 3.2.

3.4 Self-Organizing Sensor Network

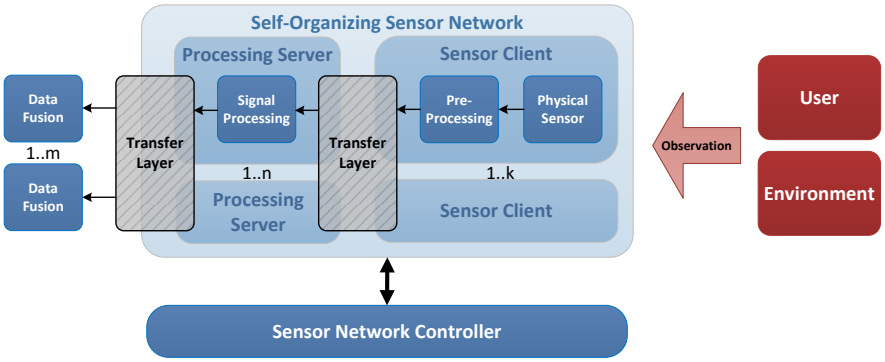

Fig. 7 The smallest version of a *Self-Organizing Sensor Network*. The *Transfer Layer* can have *n* inputs and *m* outputs. Based on that it is possible to build a complex sensor network as described in Fig. 9.

The sensor phalanx of the prototyped *Ticked Vending Machine* is set up as a distributed sensor network, with mostly sending abstract information via the *SEMAINE* protocol. But including sensors of mobile devices exceeds the computation power of those. Therefore the data processing needs to be relocated in some situations. The signal processing is commonly implemented in software executable on every available processor. By introducing a client server model for sensor data processing, computationally demanding processing parts can be transferred to any computation center. This allows *Companion*-Systems even to be used with smart phones or tablets. The idea of the *Self-Organizing Sensor Network* is to act as a subsystem of a *Companion*-System. It realizes and controls the required sensor data streams and distributes them according to the computation capabilities. Controlling the sensor data streams includes a network time based synchronization method to relate the sensor readings to a common time base.

A generalizing abstraction is given in Fig. 7, where the client only contains the physical sensor and a preprocessing unit. A transfer layer is distributing the sensor data to several signal processing units if required. The required sensor synchronization is realized by time stamps derived from a high precision network time, which gives the required accuracy in the time domain. Several signal processing units can use the data of a certain sensor, e. g. prosody extraction and speech recognition. The transfer layer can realize connections with *n* inputs and *m* outputs and is able to realize a complex sensor network. The *Self-Organizing Sensor Network* enables the dynamization of the sensor setup (see Sect. 4).

3.5 Situation- and User-Adaptive System Functionality

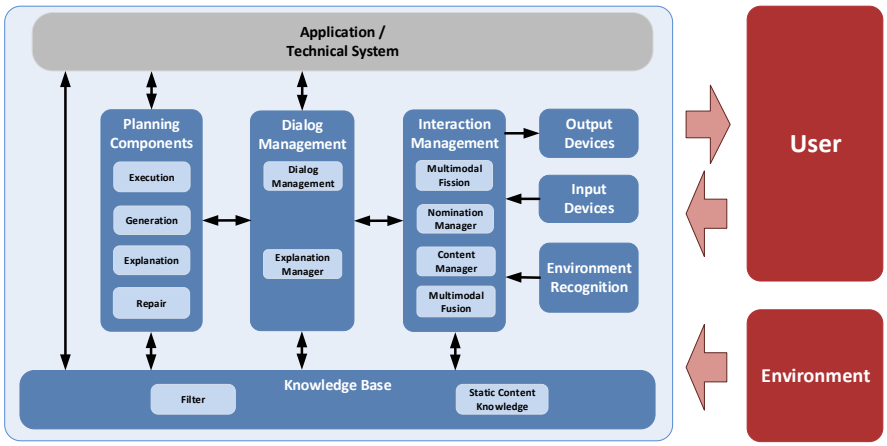

Fig. 8 Abstract view of the architecture of the prototyped *Wiring Assistant* for wiring a home theater, based on the architectural approach as presented by Honold et al. [12].

The *Wiring Assistant* setting [2, 3, 9] implements an assistance system for wiring a complex home theater (see Chap. 24). The architecture (see Fig. 8) and the used components are domain-independent, thus a variety of application areas can be implemented using this system structure. The domain-specific knowledge is stored in different areas of the KB. It manages all necessary knowledge for the working system. Every system component has a connection to the centralized KB. The data structures in the KB are separated in static and time-stamp-based information from the user, environment, and the system. The knowledge-based system integrates substantial planning-capabilities (Plan Generation, Execution, Explanation and Repair) with an adaptive multimodal user interface (Dialog and Interaction Management). This *Wiring Assistant* system is only equipped with a minimal sensor setup being sufficient for demonstrating the capabilities for an assistance system like this. The laser rangefinder is used for user-localization. Data from the used touch screen sensors in combination with data from the laser rangefinder is used for user identification. The integration of a comprehensive sensor-setting has already been implemented in the *Ticket Vending Machine* setting (see Chap. 25 and Sect. 3.3) and is used in that configuration. All components of the system communicate via messages through specific channels of the message-based middleware. By a user requesting assistance of the application, the assistance process is initialized by passing the user given task dedicated planning problem to the plan generation component. It generates a solution e. g. a wiring plan, which describes doing a complete wiring of the devices of the respective home theater system. This solution is sent to the plan explanation and plan execution components. The plan execution component monitors

the system, identifying the next plan step in dependency of the current state of the wiring process and sends it to the dialog management.

The dialog management component generates the facility of presentation for every activated plan step. To reach the goals of each plan step, the dialog management identifies the most successful dialog goal by combining the actual plan step with the associated hierarchical dialog model. The goal-oriented and adaptive dialog structure is dissected and step-wise forwarded to the interaction management.

In that way, every dialog step is transmitted to the interaction management, where the fission reasons about the currently valid UI configuration. In the present system, the UI can consist of multiple devices (see Chap. 10). Based on the fission's process, the most reasonable input and output devices are in charge of rendering their dedicated parts of the UI. The user's inputs can be made explicitly over several devices (e. g. touch or speech input) or can be interpreted as implicit input via the environment sensors (e. g. when moving from one spot to another). To get a distinct input, all input channels are fused in the input fusion module.

If the user has obscurities about why the system reacts in a certain way, he can ask the system at any time for an explanation of the currently presented plan step. The dialog management gets the user's demand through the interaction management and requests an explanation for the current step from the plan explanation component. The derived explanation is generated from a formal one, which shows the necessity of the step for the overall user task (see Chaps. 24 and 5). It is sent back to the dialog management, which starts the presentation process of the answer upon the user's demand.

In some cases, the plan step does not work (e. g. if a required cable is broken and the system does not know this yet). The plan execution component can be told about the problem via user interaction and initializes plan repair. The plan generation module derives a new plan with the modified requirements.

4 A Reference Architecture for *Companion*-Systems

Fig. 1 shows that *Companion*-Technology allows to transform a cognitive technical system to a *Companion*-System. The benefit of a reference architecture is a methodical setup for *Companion*-Systems [13]. It supports the generation of new implementations and the extension of capabilities of existing systems with regard to future upgrades. The goal of the reference architecture is to cover a multitude of possible implementations for a certain system. It further increases the compatibilities of different systems among each other and concludes in the support of an inter-system-communication and dynamization (also known as loose coupling (cf. [16])).

The knowledge about the architectures of the aforementioned prototypical systems and the experience of the synchronous recording setup, can serve as basis for a common reference architecture. To manage the dynamic architecture and its components, an additional control structure is necessary. The architecture's central component is the *Controller* (see Fig. 9). The controller consists of three components: the

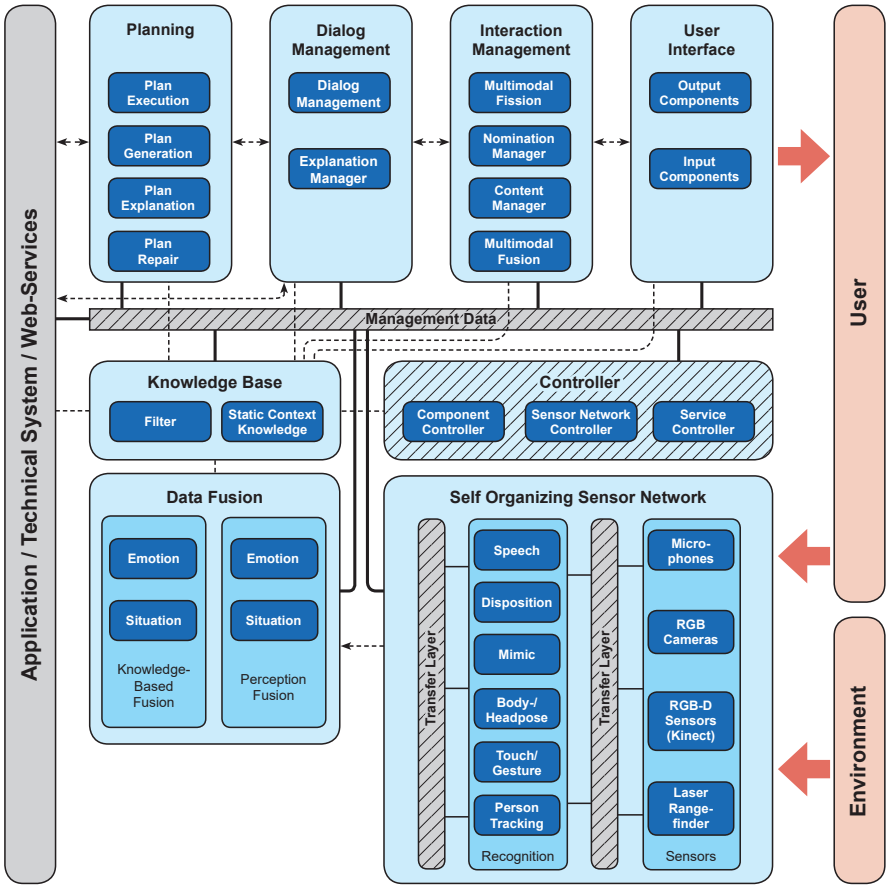

Fig. 9 A reference architecture for *Companion*-Systems with *Controller* components and *Self-Organizing Sensor Network*.

Component Controller, the *Sensor Network Controller*, and the *Service Controller*. Each controller focuses on different aspects of the overall system.

The *Service Controller* acts as the administrative interface for other components (e. g. management of requests to several controller components). It also controls the communication between the linked system modules like the *SEMAINE* system controller (cf. [24]).

All components in the system are administrated by the central *Component Controller*. It has knowledge about the configurations of possible and desired *Companion*-Systems. Using said knowledge, the controller can identify the necessary components for a specific and desired implementation. The controller accesses the current state of all available components, which enables the controller to react to a component failure according to the configuration data. Due to the goal of con-

tinuous availability and full functionality of the system, the controller manages the distribution of the functionalities between the appropriate, available modules within the system. For redundancy reasons it is also possible to operate more than the required modules with the same functionality. The controller should be able to create a functional system with reduced capabilities if there is no replacement for a missing component. In such a case, the functional reduction can be communicated to the user with an ad-hoc creation of a respective output using the dialog and the interaction management. This helps to keep the user's mental model up to date with respect to the system's offered functionality.

The *Sensor Network Controller* manages the *Self-Organizing Sensor Network* (cf. Sect. 3.4), which helps to decouple the sensor data processing and the physical sensors dynamically in order to optimize the system's observation capability.

The presented reference architecture is currently established as a concept. Its improvement goes along with the further research of *Companion*-Technology.

5 Lessons Learned and Future Work

In this chapter an architectural view on two prototyped systems and one data recording scenario is presented. The first system deals with sensor data acquisition and processing, as well as data fusion. The second system focuses on planning, reasoning, decision making, and user adaption. The system architectures of the described systems cover different aspects of *Companion*-Systems. Both systems have components of data fusion and an advanced human computer interface in common and are consequently sharing certain parts of the architectural components.

The fusion of the different architectures (see Fig. 9) leads to an architecture which is close to being referential for *Companion*-Systems. The resulting architecture is still static and by now limited to a certain range of use cases. A future goal is to increase the adaptivity of the architecture to reach a wider range of applications in technical systems.

The definition of the communication structure among all necessary components of a *Companion*-System is an important task in architecture development. A standardized interface for *Companion*-Systems would be helpful for an easy integration with existing and future projects. Furthermore such an architecture needs to cover aspects of the component interaction such as: loose coupling, service discovery, data communication management, synchronization, security, and privacy. To realize dynamization, the concept of loose coupling of modules can be applied as described by Krafzig et al. [16]. This concept is closely related with the idea of a Service-Oriented Architecture (SOA), where logical modules can be combined on demand based on a common service description language. The required modules could be discovered from a repository as described by Papazoglou et al. [20]. If necessary, specific and service-related UI elements can be realized as described by Honold

et al. [10]. Privacy issues can be addressed using Schaub's *PriMA*[4] approach [23] along with the process of modality arbitration as described by Honold et al. [11].

Mobile devices, such as smart phones and tablet-PCs, are probable target systems for the main interaction with the user. They contain sensors, recording a user's voice, face and position, and carry private data as well. Especially when using an individual, specific KB in insecure networks, it is a goal for the future to consider and integrate issues of security and privacy. The KB needs to contain information about several areas, which may be required to be secured against each other. Mobile devices are usually wireless components with very limited computing power, which will be part of the distributed system used to generate a *Companion*-System. Therefore no hard sensor data acquisition synchronization can be established. The synchronization needs to be realized via network-based communication.

A distributed system can be configured dynamically for different users' demands by setting up the *Companion*-System using existing hardware components as sensors and interaction devices in the current environment of any user. It is possible to add modules of the architecture — like KB or data fusion — solely using the network communication. Then, the time constraints regarding the data fusion are important for the design of the system architecture. Finally the benefit of *Companion*-Technology could increase if one running *Companion*-System could connect to and communicate with other *Companion*-Systems around.

Acknowledgements This work was supported by the Transregional Collaborative Research Centre SFB/TRR 62 "*Companion*-Technology for Cognitive Technical Systems" which is funded by the German Research Foundation (DFG). The authors thank the following colleagues for their invaluable support (in alphabetical order): Pascal Bercher, Peter Kurzok, Andreas Meinecke, Bernd Schattenberg, and Felix Schüssel.

References

1. Atrey, P.K., Hossain, M.A., El Saddik, A., Kankanhalli, M.S.: Multimodal fusion for multimedia analysis: a survey. Multimedia Systems **16**(6), 345–379 (2010). DOI 10.1007/s00530-010-0182-0
2. Bercher, P., Biundo, S., Geier, T., Hoernle, T., Nothdurft, F., Richter, F., Schattenberg, B.: Plan, repair, execute, explain – how planning helps to assemble your home theater. In: Proceedings of the 24th Int. Conference on Automated Planning and Scheduling (ICAPS 2014), pp. 386–394. AAAI Press (2014)
3. Bercher, P., Richter, F., Hörnle, T., Geier, T., Höller, D., Behnke, G., Nothdurft, F., Honold, F., Minker, W., Weber, M., Biundo, S.: A planning-based assistance system for setting up a home theater. In: Proc. of the 29th National Conference on Artificial Intelligence (AAAI 2015), pp. 4264–4265. AAAI Press (2015)
4. Biundo, S., Wendemuth, A.: Companion-technology for cognitive technical systems. KI – Künstliche Intelligenz (2015). DOI 10.1007/s13218-015-0414-8
5. Caschera, M.C., D'Ulizia, A., Ferri, F., Grifoni, P.: Multimodal systems: An excursus of the main research questions. In: I. Ciuciu, H. Panetto, C. Debruyne, A. Aubry, P. Bollen, R. Valencia-García, A. Mishra, A. Fensel, F. Ferri (eds.) On the Move to Meaningful Internet

[4] PriMA – privacy-aware modality adaptation

Systems: OTM 2015 Workshops, *Lecture Notes in Computer Science*, vol. 9416, pp. 546–558. Springer International Publishing (2015). DOI 10.1007/978-3-319-26138-6_59
6. Glodek, M., Honold, F., Geier, T., Krell, G., Nothdurft, F., Reuter, S., Schüssel, F., Hörnle, T., Dietmayer, K., Minker, W., Biundo, S., Weber, M., Palm, G., Schwenker, F.: Fusion paradigms in cognitive technical systems for human–computer interaction. Neurocomputing **161**(0), 17 – 37 (2015). DOI 10.1016/j.neucom.2015.01.076
7. Glodek, M., Tschechne, S., Layher, G., Schels, M., Brosch, T., Scherer, S., Kächele, M., Schmidt, M., Neumann, H., Palm, G., Schwenker, F.: Multiple classifier systems for the classification of audio-visual emotional states. In: S. D'Mello, A. Graesser, B. Schuller, J.C. Martin (eds.) Affective Computing and Intelligent Interaction, *LNCS*, vol. 6975, pp. 359–368. Springer (2011). DOI 10.1007/978-3-642-24571-8_47
8. Handrich, S., Al-Hamadi, A.: Multi hypotheses based object tracking in HCI environments. In: Image Processing (ICIP), 2012 19th IEEE International Conference on, pp. 1981–1984 (2012). DOI 10.1109/ICIP.2012.6467276
9. Honold, F., Bercher, P., Richter, F., Nothdurft, F., Geier, T., Barth, R., Hörnle, T., Schüssel, F., Reuter, S., Rau, M., Bertrand, G., Seegebarth, B., Kurzok, P., Schattenberg, B., Minker, W., Weber, M., Biundo, S.: *Companion*-Technology: Towards user- and situation-adaptive functionality of technical systems. In: Intelligent Environments (IE), 2014 10th Int. Conference on, pp. 378–381. IEEE (2014). DOI DOI10.1109/IE.2014.60
10. Honold, F., Poguntke, M., Schüssel, F., Weber, M.: Adaptive dialogue management and UIDL-based interactive applications. In: Proc. of the Int. Workshop on Software Support for User Interface Description Language (UIDL 2011). Thales Research and Technology France, Paris (2011)
11. Honold, F., Schüssel, F., Weber, M.: Adaptive probabilistic fission for multimodal systems. In: Proc. of the 24th Australian Computer-Human Interaction Conf., OzCHI '12, pp. 222–231. ACM (2012). DOI 10.1145/2414536.2414575
12. Honold, F., Schüssel, F., Weber, M., Nothdurft, F., Bertrand, G., Minker, W.: Context models for adaptive dialogs and multimodal interaction. In: Intelligent Environments (IE), 2013 9th Int. Conference on, pp. 57–64. IEEE (2013). DOI 10.1109/IE.2013.54
13. Hörnle, T., Tornow, M.: Reference architecture approach for companion-systems. Presented on the first Int. Symposium on Companion-Technology (2015)
14. Hrabal, D., Kohrs, C., Brechmann, A., Tan, J.W., Rukavina, S., Traue, H.: Physiological effects of delayed system response time on skin conductance. In: F. Schwenker, S. Scherer, L.P. Morency (eds.) Multimodal Pattern Recognition of Social Signals in Human-Computer-Interaction, *LNCS*, vol. 7742, pp. 52–62. Springer (2013). DOI 10.1007/978-3-642-37081-6_7
15. Knappmeyer, M., Kiani, S., Reetz, E., Baker, N., Tonjes, R.: Survey of context provisioning middleware. Communications Surveys Tutorials, IEEE **15**(3), 1492–1519 (2013). DOI 10.1109/SURV.2013.010413.00207
16. Krafzig, D., Banke, K., Slama, D.: Enterprise SOA: Service-oriented Architecture Best Practices. The Coad series. Prentice Hall Professional Technical Reference (2005)
17. Layher, G., Liebau, H., Niese, R., Al-Hamadi, A., Michaelis, B., Neumann, H.: Robust stereoscopic head pose estimation in human-computer interaction and a unified evaluation framework. In: G. Maino, G. Foresti (eds.) Image Analysis and Processing – ICIAP 2011, *LNCS*, vol. 6978, pp. 227–236. Springer (2011). DOI 10.1007/978-3-642-24085-0_24
18. Niese, R., Al-Hamadi, A., Panning, A., Michaelis, B.: Emotion recognition based on 2D-3D facial feature extraction from color image sequences. Journal of Multimedia **5**(5), 488–500 (2010)
19. Panning, A., Al-Hamadi, A., Michaelis, B., Neumann, H.: Colored and anchored active shape models for tracking and form description of the facial features under image-specific disturbances. In: I/V Communications and Mobile Network (ISVC), 2010 5th Int. Symposium on, pp. 1–4. IEEE (2010)
20. Papazoglou, M.P., Traverso, P., Dustdar, S., Leymann, F.: Service-oriented computing: State of the art and research challenges. Computer (11), 38–45 (2007)
21. Reuter, S., Dietmayer, K.: Pedestrian tracking using random finite sets. In: Information Fusion (FUSION), 2011 Proc. of the 14th Int. Conference on, pp. 1–8 (2011)

22. Rösner, D., Frommer, J., Friesen, R., Haase, M., Lange, J., Otto, M.: LAST MINUTE: a multimodal corpus of speech-based user-companion interactions. In: LREC, pp. 2559–2566 (2012)
23. Schaub, F.M.: Dynamic privacy adaptation in ubiquitous computing. Dissertation, Universität Ulm. Fakultät für Ingenieurwissenschaften und Informatik (2014)
24. Schröder, M.: The SEMAINE API: Towards a standards-based framework for building emotion-oriented systems. Advances in Human-Computer Interaction **2010** (2010). DOI 10.1155/2010/319406
25. Sharma, R., Pavlovic, V., Huang, T.: Toward multimodal human-computer interface. Proceedings of the IEEE **86**(5), 853–869 (1998). DOI 10.1109/5.664275
26. Siegert, I., Hartmann, K., Philippou-Hübner, D., Wendemuth, A.: Human behaviour in HCI: Complex emotion detection through sparse speech features. In: A. Salah, H. Hung, O. Aran, H. Gunes (eds.) Human Behavior Understanding, *LNCS*, vol. 8212, pp. 246–257. Springer (2013). DOI 10.1007/978-3-319-02714-2_21
27. Siegert, I., Hartmann, K., Philippou-Hübner, D., Wendemuth, A.: Human behaviour in HCI: Complex emotion detection through sparse speech features. In: A. Salah, H. Hung, O. Aran, H. Gunes (eds.) Human Behavior Understanding, *LNCS*, vol. 8212, pp. 246–257. Springer (2013). DOI 10.1007/978-3-319-02714-2_21