# Software Architecture Trends and Promising Technology for Ambient Assisted Living Systems

Martin Becker

Fraunhofer IESE
Kaiserslautern, Germany
martin.becker@iese.fraunhofer.de

**Abstract.**
Driven by the ongoing demographical, structural, and social changes in all modern, industrialized countries, there is a huge interest in IT-based equipment and services these days that enable independent living of people with specific needs. Despite of promising concepts, approaches and technology, those systems are still rather a vision than reality. In order to pave the way towards a common understanding of the problem and overall software solution approaches, this paper (i) characterizes the Ambient Assisted Living domain, (ii) briefly presents relevant software architecture trends, esp. applicable styles and patterns and (iii) discusses promising software technology already available to solve the problems.

## 1. Introduction

Driven by the ongoing demographical, structural, and social changes in all modern, industrialized countries, there is a huge interest in IT-based equipment and services these days that enable independent living of people with specific needs, esp. elderly. In Europe the term Ambient Assisted Living (AAL) [AAL] is used to denote such solutions. Among the prominent goals of AAL solutions are: (i) increase quality of life for care taking and care giving persons, (ii) reduce need for external assistance, (iii) reduce health and care costs for the individual and the society, (iv) avoid stigmatisation.

The philosophy behind AAL is considerably influenced by the Ambient Intelligence [AHS01] paradigm which means a major paradigm shift towards embedding the individual human in an intelligent environment which guarantees proactive assistance and enables totally natural interaction. These environments are envisoned to consist of networked device ensembles, following common strategies in order to react intelligently to the user's situation, interaction and goals in a coherent way. Despite of promising concepts, approaches and technology, AAL is still rather a vision than reality. The fact that solutions from different fields with different quality profiles have to be merged in order to meet the AAL goals, can be considered as one of the major inhibitors in the field.

In order to pave the way towards a common understanding of the problem and overall software solution approaches, this paper (i) characterizes the AAL domain, (ii)

Dagstuhl Seminar Proceedings 07462
Assisted Living Systems - Models, Architectures and Engineering Approaches
http://drops.dagstuhl.de/opus/volltexte/2008/1455

briefly presents relevant software architecture trends, esp. applicable architecture styles and patterns and (iii) discusses promising software technology already available to solve the problems.

The paper is structured as follows: In section 2, we provide a classification scheme for AAL services, identify some user and environmental challenges to be addressed by any IT-based AAL solution, and derive a list of essential system qualities from these challenges. Section 3 elaborates how the qualities elicited above can be met by current architectural styles and patterns. Section 4 takes a look at promising technology that can ease the development of AAL systems. It covers middleware, semantic models and context management. Finally, we draw some conclusions in Section 5.

.

## 2. Ambient Assisted Living Domain

This section briefly characterizes the Ambient Assisted Living domain. To this end, it first provides a classification scheme for living assistance services, then identifies some user and environmental challenges to be addressed by any IT-based AAL solution, and finally derives a list of essential system qualities from these challenges.

### Classification Scheme

The potential range of services belonging to the living assistance domain is huge. It encompasses any assistive service that eases daily life. The *classification scheme* in Fig. 1 [NKBL06] structures this domain into six stereotypical subdomains:

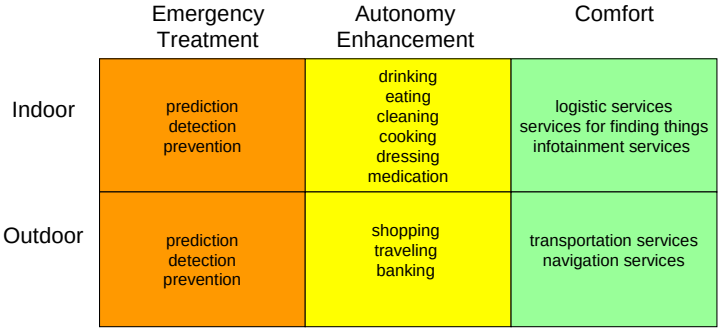



Fig. 1. A classification scheme for the living assistance services

In the first dimension it is worthwhile to distinguish between indoor and outdoor living assistance. Indoor assistance services are rendered in a well defined locality scope: in apartments, homes, cars, hospitals, and elderly care homes. Indoor assistance services can be built upon a well-known hardware/software installation in the location, thereby providing a stabile environment.

Outdoor assistance services support persons during activities outside their homes: while shopping, traveling, and during other social activities. These services have to

face with highly unstable environmental conditions such as availability of technical installation and other resources.

Another useful classification dimension is the type of service provided. According to Fig. 1, we distinguish between three types of services:

1. *Emergency treatment* services aim at the early prediction of and recovery from critical conditions that might result in an emergency situation and the safe detection and alert propagation of emergency situations.
2. *Autonomy enhancement* services enable an independent living of the assisted persons, in case of lacking capabilities of the assisted persons.
3. *Comfort services* cover all areas that do not fall into the categories (a) and (b) above. These services ease the daily life but are not required necessarily.

It is clear from the discussion that comfort services do not have the same importance and social impact as the other two categories. Furthermore, the classification of a service depends on the current capabilities of the assisted person and their environmental situation thus can change over time. Although all three service categories are desirable in general, as they make life easier for the target group and their respective environment, emergency treatment can be considered to form the kernel of any AAL service portfolio. This is motivated by the significantly increasing probability of emergencies with age coupled with the decreasing capability to cope with the emergencies.

### User and Environmental Challenges

AAL services intend to enable people with specific needs to live an independend and save life. To this end the services have to cope with the following *user and environmental challenges*, known from literature and revealed with intensive discussions with the target user group in our assisted living lab [ALL, RBK07]:

1. *Low, declining capabilities*. Some of the capabilities of the assisted persons are rather low and thus raise the demand for external assistance. Furthermore the capabilities of the users decline over time. Hence they require different assistance services over time.
2. *Differing needs of the users*. The capabilities, goals, and habits of the individuals differ substantially, at first between the individuals and at second over time. Hence there is no one-size-fits-all solution.
3. *Limited resources*. The available human assistance and financial resources are limited these days and will become scarce within the next years.
4. *Low acceptance of technical problems*. The assisted persons show only little acceptance of AAL services [AAL06], if they encounter problems with them.
5. *Involvement in active life.* Despite of demand for external assistance, the assisted persons want to be involved in active life as long as possible. The total replacement of human assistance by a pure technical assistance is not accepted.
6. *Keep control*. Although the demand for living assistance increases over time, the assisted persons want to keep the control over the assistance services.
7. *Avoid stigmatisation*. The stigmatisation of the assisted persons through visible assistance devices should avoided if possible in order to raise the acceptance of the AAL services.

8. *Keep privacy*. Information about the current and past situation of the assisted person should only be provided to persons and institutions that are actually involved in providing the assistance services.

## System and Service Qualities

The aformentioned user and environmental challenges are characteristic for the AAL domain and require specific solution approaches. From them we can derive the following *system and service qualities* for any technical AAL solution. The challenges that drive these qualities are listed in braces.

- *Affordability* (3). The services and the required technical installation and resources must be affordable.
- *Usability and User Experience* (1, 4, 6, 7). If the assistance services require some explicit interaction with the assisted person or some device usage, the assisted person must have the required capabilities and should have a positive experience during the service usage. Multi-modal interaction paradigms that combine several modes are a powerful approach to enhance usability. *Anticipatory* interfaces that proactively contact persons in certain situations are considered mandatory.
- *Suitability* (1, 2, 3, 4, 5, 6, 7). The services must meet the demands of the assisted person and they must have a direct benefit from them, otherwise the acceptance will decline over time.
- *Dependability* (1, 4, 6, 8). The solutiones must be *robust* against all kinds of misuse and errors and the services must be *available* even in the presence of hardware component crashes and shortage of resources. Furthermore, the systems should perserve the *safety* of the assisted persons and not harm them. Despite of continuously monitoring persons, the solutions must guarantee a well defined degree of *security and privacy* for the persons under observation.
- *Adaptivity* (1, 2, 3, 4). The systems must able to adapt themselves at runtime. Adaptivity on different levels and scales is considered as one outstanding characteristic of AAL systems. To support this, the systems must monitor themselves, the users and their environment. Based on the gained insights, the system can perform a
  - *self-configuration*, which denotes the ability of the system to integrate dynamically new software components and remove existing ones not needed any more.
  - *self-healing*, which denotes the ability to detect problems of components and take appropriate countermeasures.
  - *self-optimization*, which denotes the ability of the system to adapt its algorithmic behavior to changing needs of the application.
  - *self-protection*, which denotes the ability of the system to protect itself against misuse.
- *Extensibility* (1, 2, 3). The system must support its seamless extension by new devices and services at runtime, in order to adapt the system to changing demands over time.

- *Resource Efficiency* (1, 2, 3). The available resources, i.e., processing power, time, memory, communication bandwidth and energy, have to be utilized as efficiently as possible.
- *Heterogeneity* (1, 2, 3, 5, 7). The system typically consists of several subsystems provided by different manufacturers.

Obviously, there are a lot of tradeoffs between these qualities, e.g. between afordability and dependability, or between adaptivity and resource efficiency, that need to be addressed. Unfortunately, there is currently only a poor understanding how these qualities actually contribute to the overall achievement of the user goals and how these tradeoffs can be handled in an optimal way. This complicates the development of appropriate solutions considerably, and leads to suboptimal solutions. A sound quality model revealing the required qualities and their interdependencies would be really helpful here, but is not available today and requires intensive interdisciplinary research within the coming years.

A strategy to cope with the evident quality tradeoffs in the AAL domain could be "just enough quality" [BWAK06]. As the assistance service provided by humans these days are not 100% all the time, minor quality problems of the AAL solutions should also be tolerable, as long as they can be compensated by human assistants in time and do not decrease the quality of life for the assisted persons.

## 3. Architecture Trends

The first artefact in the solution space that adresses quality explicitly is the architecture. This section elaborates on how the system qualities elicited in section 2 are met by current architecture trends. First it presents a reference model für AAL systems, then it gives an overview on promising architectural styles, and finally discusses some patterns to address awareness and adaptivity in the systems.

Architectures are the central design artifacts that address quality requirements such as cost, dependability, performance, etc. of the overall solution. In any software-intensive system especially the software architecture plays a pivotal role for the quality achievement. The software architecture of a computing system is the structure or structures of the system, which comprise software elements, the externally visible properties of those elements, and the relationships among them [BCK03].

For the time beeing, there is no commonly accepted (reference) architecture for AAL systems. Different approaches are followed to meet the functional and quality requirements in the present and developing systems. Due to this, we will present some relevant architectural structures in the reminder of this section, namely a reference model and some architecture styles.

### Architecture Reference Model

An architecture reference model is a division of the functionality together with data flow between the pieces [BCK03]. AAL systems are assistance systems. The assistance functionality comprises two aspects: (i) an easy access for the assisted

person to autonomy enhancement or comfort services, e.g. home control, social interaction, etc., and (ii) the anticipatory assistance of the assisted person with proactive emergency treatment or autonomy enhancement services, e.g. automatic alarms, home automation, notification. In order to support the anticipatory assistance the system must be some kind of closed loop controller [Shaw95] that senses its environment and especially the persons living therein and influences the environment with its actuators. The rendered functionality can be decomposed into two blocks: Awareness and Presence.

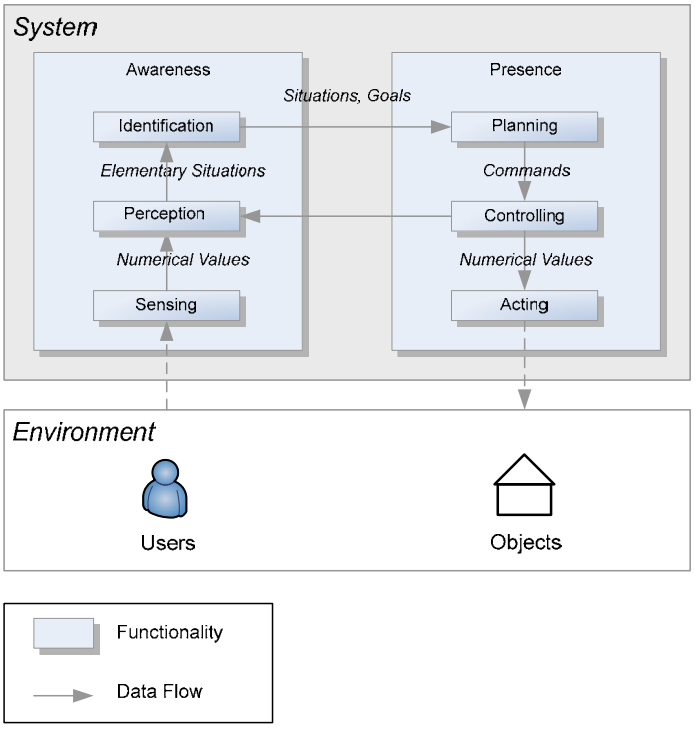



Fig. 2. Functional Blocks of AAL system

*Awareness* denotes the sound understanding of the assisted persons' current situation – in terms of physical and mental health condition –, the persons' context, and goals, as well as the environmental situation. Additionally, the system must be able to predict the response of the persons and their environment to possible actions and must be able to align predicted and actual responses. To operationalize this awareness, the systems must comprise a model about the user, the environment, and the system itself. We denote these models as Human Capability Model (HCM), Environment Model (EM), and System Model (SM). These models must allow the personalization of the rendered assistance services, e.g. customizing it by taking the existing disabilities of a person into account. The basic idea of a HCM [NKBL06] is illustrated in Fig. 3.

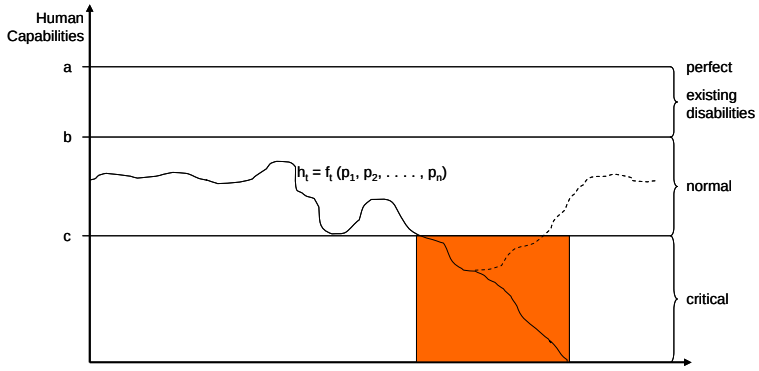



Fig. 3. A Human Capability Model.

The figure shows three horizontal lines: (a) defines a fictive perfect health condition of a human without any disabilities, (b) defines a particular person's best possible health condition regarding existing disabilities, and (c) defines the borderline between normal and critical health conditions. Taking existing disabilities of a person into account is mandatory for any emergency treatment system, in order to prevent misinterpretation of situations. For instance, a person with locomotor disturbance will move slower than a person who does not suffer from this disability.

The fictive curve below line (b) defines the overall health condition *h* of the person as a function over time. The function depends on a large, unknown number of parameters and is virtually impossible to describe in a precise mathematical sense. Even worse, some of these parameters cannot be measured directly, but can only be estimated by means of secondary measures. However, an approximation of this model can be achieved by continuously acquiring data about a persons' location, activities, and vital state, i.e. by monitoring and understanding the daily routine of the assisted person.

The awareness of the system again can be decomposed in three functional blocks as illustrated in Fig. 2 in the left column:

- *Sensing* senses its environment and especially the persons living therein and provides its results as numeric values.
- *Perception* transforms sensed numeric values into symbolic ones, e.g. the location of a object, and detects relevant situations. While doing so, the signal quality is improved by means of sensor fusion. Perception can be considered as a kind of parser. If required, it also tracks the situations in form of traces over time and aggregates them into traces on different abstraction levels.
- *Identification* analyses the perceived situations and assesses them. It also recognizes recurring patterns as well as deviations from them and thus identifies higher level situations, e.g. a situation of helplessness. Illustratively explained, it forms sentences from words.

The *presence* part of the assistance system comprises those functionalities that directly have an effect on the assisted persons or their environment, e.g. a medication reminder, or an automatic control of the lights during night. This functionality can be decomposed into the following functional blocks:

- *Planning* reasons on the current situations and goals is gets from the Awareness block about if and which kind of assistance is required, plans this assistance, and issues respective high level commands, e.g. remind of medication.
- *Controlling* executes commands first by decomposing them into single actions (split the sentences into words) and second by translating the actions into numeric values.
- *Acting* influences the environment according to the numeric values.

From the data flow perspective AAL systems form a kind of signal or data processing pipeline, which seems to be a common style of Ambient Intelligence systems [HeKi04]. In order to enable some self-adaptation and learning, a closed internal feedback loop between Controlling and Perception is required in the system as illustrated in Fig. 2. This also allows to detect deviations from the expected behavior. It has to be mentioned, that the explicit human computer interaction is considered as a special kind of Sensing, Perception, Controlling, and Acting, for the sake of simplicity here.

## Architectural Styles

Several ways are conceivable to realize the conceptual decomposition of functionality described above in the system physically. From a physical perspective the overall system topology of the AAL system will consist of tens to hundreds of different, interacting nodes, ranging from tiny sensor nodes, over mobile or embedded systems with fairly low computational power, up to powerful machines as illustrated in Fig. 4.

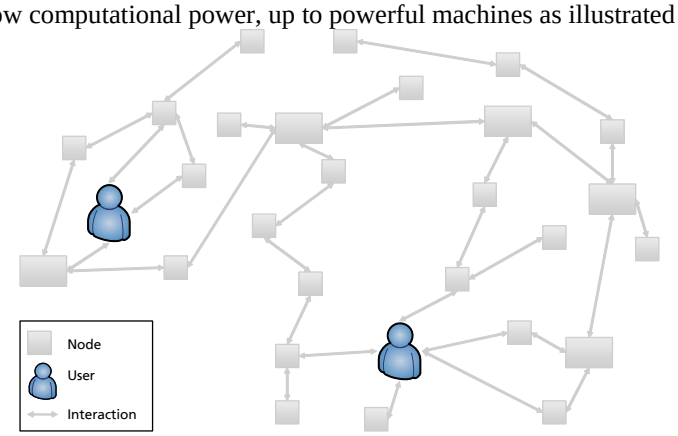



Fig. 4. AAL System structure.

As a major challenge for the engineering of AAL systems is considered how this diversity in time and space can be integrated in a seamless way to render the assistance services and a coherent way. Several *architectural styles*, defining the element types and their relationships of in the system architecture [BCK03], are currently discussed for the applicability in the AAL domain. Defining such a style is a

complex task, as the actual taken style has a very strong influence on the overall system qualities.

Promising *architectural styles* for the AAL domain these days are:

- *Service-Oriented-Architecture (SOA)* is widely regarded as the software paradigm of the next decade [SOPR], especially in the field of information systems. A central quality of SOA is to support an easy exchange of implementations and orchestation of new functionality which contributes positively to the modifiability and extensibility of the systems by separating the contract (service) from the implementation (component). This results first in a loose coupling of the services. The fact, that the services are composed based on contracts also increases the reusability of the services and components that render the services. Open issues are how to specify the service contracts in an appropriate way and how to assure quality of service end to end, i.e. if several services are involved in the overall service provision.
- *Service-Oriented Device Architecture (SODA)* [DCKM+06] is an adaptation of SOA and aims at providing high-level abstractions also on physical-world devices, rendering the ever-increasing number of interfaces and access methods transparent to system design & development and abstracting interaction between IT systems and physical world devices in a set of services.
- *Peer-To-Peer systems (P2P)* are known for their robustness, performance, and extensibility. They use diverse connectivity between participants in a network and the cumulative bandwidth of network participants rather than conventional centralized resources and thus circumvent performance bottle necks, and single points of failures. A pure peer-to-peer system consists of equal peer nodes that in principle can render the same functionality and simultaneously function as both "clients" and "servers". Obviously, this style works well in a class of nodes that have the same capabilities, e.g. processing power and communication bandwidth. In heterogenious systems that consist of small sensor nodes up to powerfull servers, the style will not be appliable for the whole system but rather for device ensembles of the same class.
- *Event-driven architecture (EDA)* follows the publish-subscribe pattern. EDA systems are promoting the production, and reaction to events, where an event denotes a significant change [Mani06]. Building applications and systems around an event-driven architecture allows these applications and systems to be constructed in a manner that facilitates more responsiveness, because event-driven systems are, by design, more normalized to unpredictable and asynchronous environments [Hans05]. In EDA is it faily easy to realize n:m communcation structures where the producers of events are not aware of the type and number of event consumers, which is a considerable difference to SOA. This considerably contributes to the extensibility and modifiability of the systems.
- *Component and Connector (C2)* [TMAW+96] systems are structured in components and connectors. The connectors form a kind of information exchange busses for the components. The style allows an easy n:m communication between the components in different pipelines and thus the seamless integration of new solutions. C2 requires a communication strategy per connector (aka. channel) that determines who actually receives the available data. These strategies are controlled by the connectors themselves [HeKi04].

- *Multi Agent System (MAS)* is a promising style to realize distributed adaptive systems. MAS systems are composed of several software agents - sometimes claimed to be autonomous - collectively capable of reaching goals that are difficult to achieve by an individual agent or monolithic system. MAS can manifest adaptivity and complex behaviors even when the individual strategies of all their agents are rather simple [MAS].
- *Blackboard* systems consist of three major components: (i) the specialist modules, which form the knowledge sources, (ii) the blackboard, a shared repository of problems, partial solutions, and suggestions used for information exchange between the specialists and (iii) a control shell, which controls the flow of problem-solving activity in the system. Blackboard is a prominent style to solve ill-defined problems in the field of Artificial Intelligence.

In all probability, none of the aforementioned styles will fit perfectly in the AAL domain. It is expected [SOPR] that some of these styles, e.g. SOA and EDA, will converge to some point (and in some maner), in order to meet the different quality demands in the best possible way. In the BelAmI project [BelAmI] we are investigating such a hybrid style that combines SOA and EDA to support a context awareness of services and provides a dedicated support for the adaptivity (aka. self adaptation) of the system – two central qualities of any AAL system. In the reminder of this section we elaborate two patterns, how these qualities can be approached.

## Patterns

*Context-awareness* is a property of a system that uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task [DSA01]. *Context* here denotes any information that can be used to characterize the situation of an entity. An entity is a person, place, or object that is considered relevant to the interaction between a user and an application, including the user and application themselves [DSA01]. In order to make the components of a system and thus the whole system context-aware, a feasible and often followed pattern is to provide a dedicated connector for the provision and consumption of context in the system that is managed by a so-called ContextManager. Context can thus be provided and received in a n:m way per push or pull. Among the tasks of the ContextManager is to manage the access to context and to broke the information between the involved components. Based on the context connector blackboard-like structures can be realized that revealed advantages in ill-defined problems. Arbitrary data flows can be realized and the access to the context can be controlled easily. By hiding the context providers from the context consumers the ContextManager additionally supports a loose coupling of the involved components, which improves the systems modifiability and extensibility. Open issues these days are, how to represent the context in a standardized way that allows an easy exchange of context between different (sub) systems and how to represent the quality of the context, e.g. accuracy, trustability, etc.

On top of a sound context management adaptivity can be realized by a dedicated adaptation management and configuration support as proposed by the Madam project [MADAM]. Here the aspect of adaptation is clearly separated by the sole

functionality of the components, which eases the centralized optimization of the necessary adaptations. In this approach, the system components that need to be adapted register necessary adaptions that should be executed in specific contexts with an AdaptationManager. The AdaptationManager monitors continously the current context and plans required adaptations. The adaptations are conducted by a dedicated Configurator in the system that assures a consistent adaptation if several components have to be adapted or if the components must be in specific states in order to be adaptable. A considerable advantage of this approach is that the adaptation knowledge can be clearly separated from the application knowledge, that adaptations are handled in a uniform way and that global planned adaptations can be realized in a straightforward way.

## 5. Promising Technology

In this section we take a look at promising technology that can ease the development of AAL systems. First middleware approaches are addressed then semantic models and context management are covered.

### Middleware

AAL is the concept of networked devices communicating with each other and following strategies in order to react intelligently to the user's situation, interaction, and goals (context and situation awareness) to provide adequate assistance in the daily routine. In order to achieve this, it is necessary to have distributed software infrastructures, which enable the self-organization of devices and their software components. Technological developments in the field of intelligent components, which are able to interpret situations and develop strategies regarding the functions to execute, are also necessary [Aart04, Aart05]

A substantial technical contribution to AAL solutions can be expected by adequate middleware. In a distributed computing system, *middleware* is defined as the software layer that lies between the operating system and the applications on each site of the system [Krak].

Middleware for AAL must provide decentralized communication among its components in order to avoid a single point of failure. To provide extensibility, the middleware must additionally be able to execute conflict resolution strategies to guarantee reasonable data-flow even if there are competing components.

Different technologies and approaches face single aspects of the mentioned requirements:

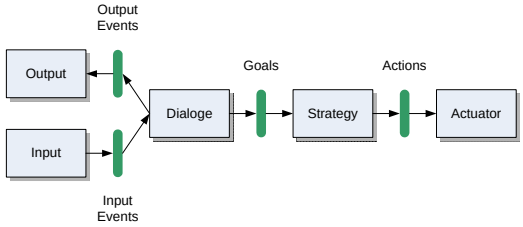
*JXTA* technology is a set of open protocols begun by Sun Microsystems in 2001 that enable any connected device on the network, ranging from cell phones and wireless PDAs to PCs and servers, to communicate and collaborate in a P2P manner [JAXTA]. It is the most mature general purpose P2P framework these days. [JAXTA2].

Universal Plug and Play (UPnP) [UPNP] is a technology and industry consortium driven by the UPnP Forum. UPnP enables devices to join automatically to a network, find and use networked devices and services provided by one another without manual configuration or involvement in a peer-to-peer fashion. It describes and supports the discovery and communication between devices to find and use services; specifically focusing on but not restricted to PCs, Internet gateways, consumer electronics and PC peripheral devices [Amigo]. Unfortunately no conflict resolution mechanisms - apart from graphical user interfaces - are provided.

The OSGi Alliance is an independent non-profit corporation comprised of technology innovators and developers and focused on the interoperability of applications and services based on its component integration platform. OSGi technology provides a dynamic module system for Java™ which can be considered as some kind of universal middleware. It provides a general-purpose, secure, service-oriented, component-based environment for developers and offers standardized ways to manage the software lifecycle. OSGi-compliant devices can download and install OSGi bundles, and remove them when they are no longer required. The framework manages the installation and update of bundles in an OSGi environment in a dynamic and scalable fashion. To achieve this, it manages the dependencies between bundles and services in detail. It also supports the dynamic reconfiguration of applications without requiring restarts. From the engineering perspective it is noteworthy, that OSGi founds the basis of Eclipse integrated development environment. Hence the same technology can be used by tool and applications, which eases in principle the integration of the tools into the applications later on. This could finally enable a continuous, holistic engineering.

It should be mentioned that the UPnP Device Architecture specification and the OSGi Service Platform complement each other fairly well. The UPnP Device Architecture specification is a data communication protocol that does not specify where and how programs execute. That choice is made by the implementations. In contrast, the OSGi Service Platform specifies a (managed) execution point and does not define what protocols or media are supported [AMIG]. Hence both of them can be combined in a synergistic way. This approach is followed in the *AMIGO* project [AMIG2] which has developed an open, standardized, interoperable middleware and attractive user services to improve people’s lives by exploiting the full potential of home networking. The abstract system architecture developed in the project recognizes the needs for service discovery and service composition strategies but each component is responsible for the application of such strategies. This means these strategies are not public or transparent to the application developers. That most problably will complicate the extension of systems based on the AMIGO platform.

The middleware model *SodaPop* [HeKi2] that is applied in the project *DynAMITE* [DYNA] offers an iteresting possibility to group components and to route the communication between the different component groups with public conflict resolution strategies.




Component topology in DynAMITE

Figure 3 illustrates one possible component topology, which identifies components as multimodal input and output components, dialogue components, strategy components and actuators. The different component levels are connected to each other through semantic channels. DynAMITE follows the C2 architectural style. The semantic channels make it possible to define specific strategies for distributing messages. These strategies can be used if there are competing components, and they can enable the cooperation of components and the fragmentation of messages into sub-messages. These strategies can be implemented in a distributed way across the software infrastructure and can thus be carried out by all devices in an ensemble. The total ensemble can therefore react dynamically to changes in its composition, and it thus behaves in a self-organizing way. On the downside, DynAMITE lacks a system-wide QoS management, which decreased the suitability of the middleware for time crititcal applications, e.g. emergency treatment services.

Besides the aforementioned middlewares there are other relevant approaches followed in the projects: AlarmNet, BelAmI, EASY-LINE+, I2Home, I-LIVING, INHOME, INTERPLAY, LARES, MONAMI, MPOWER, NETCARITY, OASIS, OLDES, Oxygene, PERSONA, SENSACTION-AAL, SOPPRANO. Each of them addresses a specific set of services and qualities and applies specific mechamisms to meet them. For the time being there is no One-Size-Fits-All solution available and the interconnection of applications realized on different middlewares is still a challenging issue.

## Semantic Models

Another promising field of technology for AAL systems besides middleware is the Semantic Web. The self- and user-awareness of AAL systems requires some conceptual understanding of the system and the environment, e.g. reflected by a Human Capability Model [NKBL06]. These models must be represented at least in a semiformal way to enable a processing in the system, and the Semantic Web follows some promising approaches to this end.

The Semantic Web is an evolving extension of the World Wide Web in which web content can be expressed not only in natural language, but also in a format that can be read and used by software agents. This allows them to find, share, and integrate information more easily. Of considerable relevance these days is the Resource Description Framework (RDF), a variety of data interchange formats, and the Web

Ontology Language (OWL), all of which are intended to provide a formal description of concepts, terms, and relationships within a given knowledge domain.

The Resource Description Framework (RDF) is a family of World Wide Web Consortium (W3C) specifications originally designed as a metadata model but which has come to be used as a general method of modeling information, through a variety of syntax formats. The RDF metadata model is based upon the idea of making statements about resources in the form of subject-predicate-object expressions, called triples in RDF terminology. The subject denotes the resource, and the predicate denotes traits or aspects of the resource and expresses a relationship between the subject and the object.

The OWL (Web Ontology Language) [OWL] constitutes the formal W3C Recommendation for an ontology representation language, builds on RDF and adds more vocabulary for describing properties and classes: among others, relations between classes, cardinality, equality, richer typing of properties, characteristics of properties, and enumerated classes. The OWL standard defines three sublanguages, which differ in expressive power. The strongest OWL sublanguage does not place any restriction on the use of OWL resources. The other OWL sublanguages contain restrictions, but in return for this, ensure higher efficiency in reasoning. Besides the languages themselves there is a substantial set of engineering and language processing tools available, e.g. Protégé [Prot] for modelling and Pellet [Pell] for reasoning.

### Context Models

Sound applications of OWL in the AAL domain comprise the specification of services, as done for instance in AMIGO [AMIG] and representation of contextual information (short: context). Context can be roughly distinguished along two dimensions: (1) user and environmental context and (2) short-lasting and long-lasting context. In order for a computer system to act context-aware, it has to understand the meaning and the impact of a situation, thus the context information has to be captured in a computer-understandable format.

In the recent years, ontologies have proven their suitability for this representation task. Two promising approaches for representing user context (context models) and for dealing with such information are SOUPA and GUMO.

SOUPA (Standard Ontology for Ubiquitous and Pervasive Applications,) [CPFJ04] is a set of ontologies for supporting pervasive computing applications. Ontologies are defined using the OWL language. SOUPA consists of two sets of ontology documents: SOUPA Core and SOUPA Extension. SOUPA Core defines generic vocabularies that are universal for different pervasive computing applications and covers Person, Policies for restricting allowed actions, Agent & Belief-Desire-Intentions, Time, Space, Events. SOUPA Extension defines additional vocabularies for supporting specific types of applications and provides examples for the future ontology extensions, e.g. Meeting & Schedule, Document & Digital Document, Image Capture, Location.

The GUMO ontology [Heck06], which is described in OWL as well, is to provide a general upper ontology for all types of user models in different application areas. The user model dimensions (derived from previous user modelling research) can be

grouped into the following categories: contact information, demographics, ability and proficiency, personality, physiological state, mental state, motion, nutrition, facial expression. The ontology does not only provide concepts, but also a comprehensive set of instances so that it is ready to use for general purpose user model dimensions.

## 5. Conclusion

The domain of Ambient Assisted Living systems is a very promising but also very challenging one. In order to enable people with specific needs to live an independend and save life, the systems have to cope with series of characteristic user and environmental challenges. They result in a set of service and system qualities that apply for all technical AAL solution. Prominent qualities are availablity, dependability, and adaptivity. Tradeoffs between these qualities complicate the engineering of AAL solutions substantially. Driven by the affordability and scarce resources it is clear that AAL solutions will somehow follow a just enough quality rationale.

The first artefact in the solution space that adresses quality explicitly is the architecture. From a functional perspective it is obvious, that any proactive assistance solution can be considered as a kind of closed loop controller. However, the physical structure of the solutions will differ substantially between the different solutions. Currently, several architectural styles are investigated for their suitability in the AAL domain. Unfortunately, none of the classical styles as SOA, Peer-To-Peer, or Multi-Agent-System can be considered as a perfect fit. Therefore hybrid approaches that combine different styles are subject of research these days. Dedicated context and adaptation management are relevant patterns to meet the quality demands in the domain.

Regarding technical solutions that are currently available it can be stated that the available middleware platforms provide a sound support for their underlying styles. However, there is a rather limited support for the required hybrid styles. Quality of Service and the the integration of systems across several middleware platforms are also open issues for research. Another promising field of technology for AAL systems is the Semantic Web. RDF and OWL can help to represent the information about the user, the environment and the system itself in a semi-formal manner. Modelling and reasoning tools have the potential to ease the development of aware AAL solutions substantionally. As with the functional services, the specification and management of informaton quality is an open field for research as well. A sound quality model revealing the required qualities and their interdependencies would be really helpful, but is not available today and requires intensive interdisciplinary endeavors within the coming years.

## Acknowledgements

The work presented in this paper was (partially) carried out in the BelAmI (Bilateral German-Hungarian Research Collaboration on Ambient Intelligence Systems) project, funded by the German Federal Ministry of Education and Research (BMBF), Fraunhofer-Gesellschaft, and the Ministry for Science, Education, Research and Culture (MWWFK) of Rhineland-Palatinate.

## References

[AAL] Ambient Assisted Living Joint Programme, http://www.aal-europe.eu/, last visited 20.01.2008

[AAL06] Steg, H.; Strese, H.; Loroff, C.; Hull, J.; Schmidt, S.: "Europe Is Facing a Demographic Challenge Ambient - Assisted Living Offers Solutions", http://www.aal-europe.eu/Published/Final%20Version.pdf, March 2006, last visited 20.01.2008

[Aart04] Aarts E., Ambient Intelligence: A Multimedia Perspective, IEEE Multimedia, p. 12-19, 2004.

[Aart05] Aarts, E.; Encarnação, J. L.: Into Ambient Intelligence, Chapter 1. In: True Visions: Tales on the Realization of Ambient Intelligence. E. Aarts and J. Encarnaçao (Eds.), Springer Verlag, Berlin, Heidelberg, New York, 2005

[ALL] Assisted Living Laboratory, http://www.belami-project.org/downloads/flyeraal.pdf/, last visited 20.01.2008

[AHS01] Aarts, E., Harwig, E., and Schuurmans, M.: Ambient Intelligence in Denning, J. (edt.): The Invisible Future, McGraw-Hill, New York, p. 235-250, 2001

[AMIG] AMIGO Project, http://www.hitech-projects.com/euprojects/amigo, last visited 20.01.2008

[AMIG2] AMIGO Project: State of the art analysis including assessment of system architectures for ambient intelligence (2005), Deliverable D2.2, http://www.hitech-projects.com/euprojects/amigo/deliverables/
Amigo_D2_2_WP2_final.pdf, last visited 20.01.2008

[BelAmI] BelAmI project, http://www.belami-project.org/, last visited 20.01.2008

[BCK03] Bass, L.; Clements, P.; Kazman, R.: "Software Architecture in Practice", Addison Wesley Publishing Company, 2003.

[BWAK06] Becker, M.; Werkman, E.; Anastasopoulos, M.; Kleinberger, T.: "Approaching Ambient Intelligent Home Care Systems", in Proceesings of 1st International Conference on Pervasive Computing Technologies for Healthcare, 2006

[CPFJ04] (Chen et al. 2004) Harry Chen, Filip Perich, Tim Finin, and Anupam Joshi: SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications, In Proceedings of the First Annual International Conference on Mobile and Ubiquitous Systems: Networking and Services (Mobiquitous 2004), Boston, MA, August 22-26, 2004.

[DCKM+06] De Deugd, S., Carroll, R., Kelly, K., Millett, Bill, Ricker, J., "SODA: Service Oriented Device Architecture," IEEE Pervasive Computing, vol. 5, no. 3, pp. 94-96, c3, Jul-Sept, 2006

[DSA01] Dey, A. K.; Salber, D.; Abowd, G. D., 2001. A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications. Human-Computer Interaction, 16, pp. 97-166, 2001

[DYNA] DynAMITE - Dynamic Adaptive Multimodal IT Ensembles, http://www.dynamite-project.org, last visited 20.01.2008

[Hans05] Jeff Hanson, Event-driven services in SOA, Javaworld, January 31, 2005

[Heck06] Heckmann, D.: Ubiquitous User Modelling, infix, 2006

[HeKi04] Hellenschmidt, M.; Kirste, T.: "A Generic Topology for Ambient Intelligence", in Markopoulos, Panos (Ed.), Ambient Intelligence Second European Symposium EUSAI 2004, Lecture Notes in Computer Science (LNCS) 3295, Springer Verlag, p. 112-123, http://www.igd.fhg.de/~mhellens/publ/EUSAI2004_ Hellenschmidt_LP.pdf, 2004.

[HeKi2] Hellenschmidt, M., and Kirste T. SodaPop: A Software Infrastructure Supporting Self-Organization in Intelligent Environments, in: Proc. of the 2nd IEEE Conference on Industrial Informatics, INDIN 04, Berlin, Germany, 24 – 26.06.2004.

[JAXTA] JAXTA™ Community Projects, https://jxta.dev.java.net/, last visited 20.01.2008

[JAXTA2] JAXTA, http://en.wikipedia.org/wiki/JXTA, last visited 20.01.2008

[Krak] Krakowiak, Sacha, What is Middleware?, http://middleware.objectweb.org/, last visited 20.01.2008

[MADAM] MADAM project, http://www.ist-music.eu/MUSIC/madam-project, last visited 20.01.2008

[Mani06] K. Mani Chandy Event-Driven Applications: Costs, Benefits and Design Approaches, California Institute of Technology, 2006

[MAS] Multi-Agent System, http://en.wikipedia.org/wiki/Multi-agent_system/, last visited 20.01.2008

[NKBL06] Nehmer, J., Karshmer, A., Becker, M., Lamm, R.: Living Assistance Systems – An Ambient Intelligence Approach. In: Proceedings of the International Conference on Software Engineering (ICSE), May 2006

[OWL] OWL Web Ontology Language Overview, http://www.w3.org/TR/owl-features/, last visited 20.01.2008

[Pell] Pellet: The Open Source OWL DL Reasoner, http://pellet.owldl.com/,

[Prot] The Protege Ontology Editor and Knowledge Acquisition System, http://protege.stanford.edu/, last visited 20.01.2008

[RDF] Resource Description Framework (RDF), http://www.w3.org/RDF/, last visited 20.01.2008

[RBK07] Ras E., Becker M., Koch J.: Engineering Tele-Health Solutions in the Ambient Assisted Living Lab, Proceedings of the Workshop First International Workshop on Smart Homes for Tele-Health, in conjunction with IEEE 21st International Conference on Advanced Information Networking and Applications, AINA'07, Niagara Falls, Canada, May, 2007

[Shaw95] Shaw, M.: "Beyond Objects: A Software Design Paradigm Based on Process Cotrol", ACM Software Engineering Notes, 20 (1), ACM Press, 1995

[SOPR] SOPRANO Project: "Review state-of-the-art and market analysis", Deliverable D1.1.2, May 2007, http://www.soprano-ip.org/ecportal.asp?id=226&nt=18&lang=1, last visited 20.01.2008.
[TMAW+96] Taylor, R. N.; Medvidovic, N.; Anderson Jr., K. M.; Whitehead, . E. J.; Jason, E.: "A Component- and Message-Based Architectural Style for GUI Software", in IEEE Transactions on Software Engineering, 22(6):390-406, http://citeseer.ist.psu.edu/taylor96component.html, 1996.
[UPNP] Universal Plug and Play, http://www.upnp.org, last visited 20.01.2008