Towards a Reference Middleware Architecture for Ambient Intelligence Systems

Majid Bavafa
Islamic Azad University, Natanz Branch
Iran
Majid_n2@yahoo.com

Neda Navidi
Amirkabir University
Iran
Neda_navidi@aut.ac.ir

Abstract—**Ambient-intelligence (AmI) systems raise a series of new challenges in software and system development: Mobility, adaptability and heterogeneity are new concerns that have to be addressed. Many of these concerns are common and therefore should be addressed by a common AmI infrastructure instead of each individual application. This position paper proposes a bottom-up approach for the development of such a middleware infrastructure, which allows for easy customization to different hardware topologies.**

Keywords—**Ambient intelligence application, middleware, reference architecture, platform.**

I. INTRODUCTION

Pervasive computing integrates computation into the environment and therefore signifies a shift from stationary, distinct computint nodes. Pervasive Computing can be seen as an integral part of Ambient Intelligence (AmI). AmI is currently seen as the next evolution step in the information society. Thereby the spectrum of computer science is being broadened from stationary systems to ubiquitous, intelligent, highly adaptable and most notably human-centric systems [1], [2]. In other words AmI systems are intelligent computer systems, which are non-invasively embedded into human environments, with the goal to ameliorate life in the most natural way based on the individual human needs [3].

The technological basis of AmI applications is a layered system, containing computing nodes of various sophistication levels.

Typically the lower levels aim at the collection of contextual information with the help of various sensors and at the implementation of according actions with various actuators. The middle and upper layers usually consist of nodes with enough computing power to allow for the interpretation of the raw contextual information on the one hand and for the automated decision making on the other. Moreover nodes of the upper levels may provide value-added services like the collection of statistical data or the integration with business processes.

The existence of different application layers poses additional difficulties to the development of AmI applications. Programmers are not only required to implement highly intelligent nodes that can adapt to the users' needs but also have to deal with the inherent heterogeneity of AmI systems, regarding the programming interfaces, communication protocols, and data structures. Therefore the need for a middleware platform simplifying application integration by acting as glue between and within the different layers becomes apparent.

Although the need for middleware is recognized in the AmI community current research usually takes a top-down approach focused on the seamless integration of low-level nodes in high-end layers.

In this position paper a bottom-up approach is proposed, in which only the lower layers are obligatory while the upper layers are optional. This alternative is better aligned with the embedded character of AmI systems, which in turn poses hard requirements concerning resource consumption. Hence, the paper proposes reference middleware architecture, which can be instantiated according to the service layers given in a specific situation.

II. APPLICATION SCENARIOS

The middleware architecture presented in this paper is being developed in the context of the BelAmI research project whose goal is the investigation, development and establishment of innovative technologies and system development methods in the area of Ambient Intelligence [4]. BelAmI takes a demonstrator-driven approach. In other words all results are to be demonstrated by implementing example applications from various domains (e.g. assisted living, automotive), correlated to AmI.

This paper concentrates on the assisted living domain. Since the domain is relatively broad the scope has been initially limited in BelAmI to an observation and reminder services (e.g. monitoring blood pressure or food and beverage consumption). Figure 1 gives an overview of the envisioned assisted living home. As shown in the picture (lower right corner) the system contains an intelligent refrigerator, which is being used as initial test bed for the middleware architecture.

The refrigerator provides a series of sophisticated services to needy persons as well as care personnel. The core functionality concerns the monitoring subsystem, which monitors the door state (opened or closed), the temperature within the refrigerator as well as the content state (expiry date) and position (in or out of the refrigerator) of food. This is based on the identification of items within the refrigerator.

The monitoring service is used by an item information service, which aggregates and filters information about all expiry dates and the alert service, which issues alerts (e.g. when the door is left open, or when food has expired).
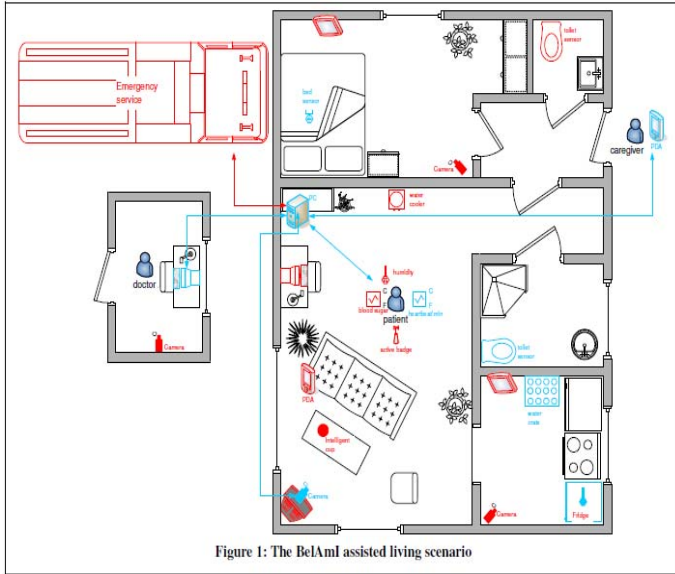

Figure 1: The BelAmI assisted living scenario

The refrigerator is also able to deliver value-added services based on the monitoring and the item information service. For example the cooking assistance service analyzes the available items and the expiry dates and proposes an according recipe. Finally the consumption monitor service delivers statistical information about the items potentially consumed and is meant to be used by service personnel.

III. HARDWARE TOPOLOGY

For the development of the middleware architecture various technical devices have been analyzed, which might be integrated in the refrigerator application. The analysis produced the following device groups:

A. Fixed Nodes Group (FNG)

All nodes belonging to this group form a fixed infrastructure of an AmI system. These nodes possess high computing power, high communication bandwidth but they are not mobile because of power constraints or because they are attached to the network by wire. Desktop computers are typical nodes in this group.

B. Portable Nodes Group (PNG)

These nodes have high computing power and high communication bandwidth as well. However, they usually have less capability than FNG nodes. Moreover they are portable, meaning that they can appear or disappear within an AmI system at run-time (of course FNG nodes could disappear as well, because of a hardware failure, but for PNG nodes this is not treated as failure but as usual case). Notebooks or pocket pcs are typical PNG nodes.

C. Sensor / Actuator Nodes Group (S/ANG)

Nodes belonging to S/ANG have only very specific computing capabilities (like digital signal processors) in order to process sensor data and communicate (usually using proprietary protocols). They can be portable as well as stationary. Microcontrollers and RF appliances forming a sensor / actuator network are typical S/ANG nodes [6], [7], [8].

D. Device Group (DG)

Nodes of this group are responsible for the acquisition of context information or for the reproduction of the application's output. In other words DG nodes have no computing power and are integrated through one of the other three groups. Bluetooth headsets, temperature sensors, or a video screen are examples of DG nodes.

These groups are illustrated in Figure 2. Nodes belonging to a group, can communicate with each other, as long as they are located in the same subnet (meaning that their communication range is large enough to reach each others for wireless nodes). PNGs and FNGs can communicate with each other also, but S/ANGs have to be integrated with one FNG or PNG in order to be able to communicate with nodes from this network (PNGs and FNGs must have bridges to the S/ANG). DG nodes are integrated via a device bay, which is a special-purpose bridge for this kind of devices [9].
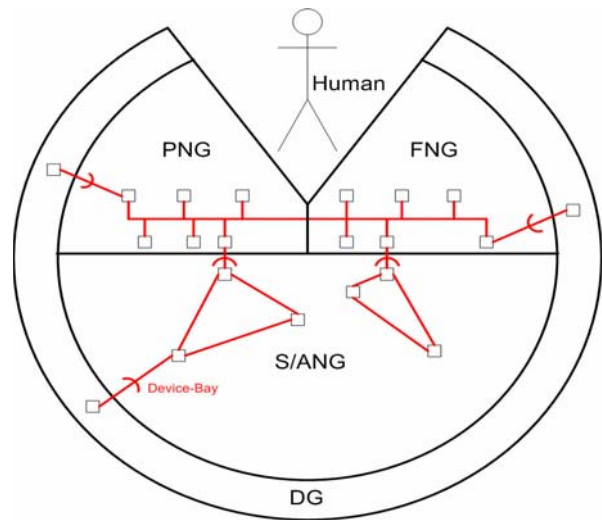

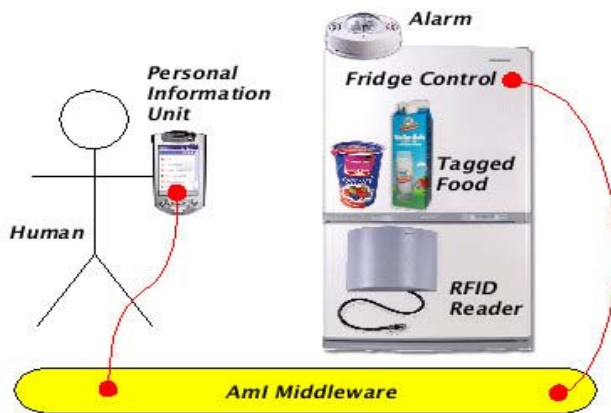Figure 2. Hardware topology within AmI systems

Figure 3. The different refrigerator components

Figure 3 illustrates a physical view of the refrigerator application. In the first step the personal information unit is a PDA connected by wireless LAN to the middleware. The refrigerator is equipped with temperature sensors, a door sensor, an RFID-Reader and alarm appliances. Items in the refrigerator are tagged with RFIDs containing the particular item type and expiry date. The antenna of the RFID reader is mounted inside the fridge.

The refrigerator control system (i.e. Fridge Control) shall be realized on a node from the S/ANG in the final setting.. This node has a proprietary connection to the RFID-Reader (UART), a connection to the door sensor, the alarm bell and the temperature sensor. Fridge control and its features are publicized as services by the middleware.

"The RFID-reader, the sensors, and the alarm bell belong to the device group. They are connected to the node Fridge Control which is visible as service to the nodes in the FNG and PNG."

IV. LOGICAL ARCHITECTURE

A logical view of the envisioned AmI system presents application components like the spoilt food monitor, working based on two different types of services:

- Application services like an alert service, which can generate acoustic or visual alerts.
- Technical services needed for the system to run.

While applications and application services are dependent on the functionality, the AmI system has to provide, we can identify the following technical services, needed generally in AmI systems:

A. Communication services

The AmI middleware should provide means for communication with each service or application around. Since not each application should implement synchronous communication from the scratch, the middleware should provide synchronous communication as well as asynchronous communication. Moreover the communication layer must be adaptive according to the requested Quality of Service characteristics. To this we envision the integration of an adaptive protocol layer as proposed in [5].

B. Event service

This part of the middleware is highly related to the communication services; since AmI applications often react based on events (e.g. the content of the fridge changes), an easy to use mechanism is needed to distribute these events amongst all AmI components which need this information.

C. Lookup service

Since AmI systems are very dynamic and new components can leave or join the system at runtime, a mechanism has to be provided enabling all components to find each other. Each service has to register itself to a lookup service, when becoming available. Applications may query the lookup service in order to get references to the services they need. As there may be more than one service, fulfilling the needs of an application, the lookup service should also be capable of taking further attributes like quality of service or context into consideration. This might be realized by using the similarity search service explained in the following.

Moreover the lookup service has to provide a mechanism for an application to register for notification when services appear which are even better suitable for the specific needs of this application. Since no application can automatically be sure to have access to all possible technical services on an AmI node the lookup service should be able to list technical services as well as application services.

D. Similarity search service

Due to the dynamicity of AmI systems, components will not always exactly match the needs. A mechanism has to be provided to find the best fitting device. For example, an application wants to find a color display using the lookup service, but there are only monochrome displays and color printers available. The lookup in this case should use the similarity search service in order to decide, which device will be more similar to the applications needs. The similarity search service might be realized using ontologies

E. Logging service

For evaluation purposes or reasoning calculations on large amounts of past sensor data a logging service is useful to store the sensed data persistent over time.

This enables other services to analyze the system's history although they might not have been present in the system all the time (for example a caregiver might enter the house of a patient and needs information about the amount drank by the patient within the last 24 hours).

F. Context Service

Context-awareness is a major feature of AmI applications. It basically involves context acquisition; interpretation, specification, and persistence [10]. These services are provided by the middleware. Position is an important aspect of context information, which requires extra attention. There are two different types of localization: absolute localization and relative localization (you are near alert bell "signal-horn"). Both types of localization information should be provided as good as possible, since each may be useful for an application.

All these technical services should be able to work, distributed among all nodes as well as centralized on a single node. The availability, complexity, and mightiness of the implementation of the abovementioned technical services will nevertheless depend on the computation and communication capabilities of the different classes of hardware nodes they are attached to.

On the logical communication level, there will be many logical busses, enabling different components of an AmI system to communicate with each other. By a publish/subscribe-mechanism, components can join these busses and distribute information on it.

Due to the level of detail addressed by this paper the described middleware architecture is variance-free. Yet we envision the middleware architecture as a reference architecture, which can be tailored according to the hardware nodes at-hand. To this end the low-level design is expected to contain variability for the purpose of covering the varying hardware characteristics. For the realization of this customizability we consider Product Line Engineering techniques. [11]

V. DYNAMIC INTEGRATION

Presumably the availability of services in AmI systems is not static. Physically or logically appearing nodes carrying services have to be dynamically integrated as well as uncoupled in case of disappearance.

The dynamic integration will be demonstrated by the spoilt food warning as follows: If a human takes expired food out of the fridge he has to be warned. Another time he opens the fridge taking something spoilt out of it, he has a Pocket PC in his hand. Looking at both scenarios in more detail from regarding the middleware, we expect the sequence shown in Figure 4. Steps one to five here show the initialization of the AmI system, which happens only once. The signal horn registers at the middleware (respectively at its lookup service). As there was no output devices registered before and no application had been interested in output devices at that moment, the middleware creates a channel, where applications, interested in output devices, can subscribe in order to be informed about new available output devices. Fridge Control queries for output devices with a given context (for example "as near to the user as possible"). As there is only one output device available, the middleware returns a

reference to the signal horn and the channel where output devices joining the system in the future will be propagated. As Fridge Control is interested in better devices, it subscribes this channel. Now the human takes food out of the fridge and the system checks, whether the food taken out is spoilt. As the food is spoilt here, a warning has to be sent to the output device by use of the communication mechanism provided by the middleware. If a more suitable output device is activated or brought in range of the ambient system it registers itself and its description is sent to all those, who subscribed. As this is an interesting device for Fridge Control, another lookup is done and the new output device will be used in the following (sketched by the dots at the end of the diagram).

A prototype for dynamic integration formerly elaborated by the Software Architecture Group at the TU Kaiserslautern showed, that dynamic integration works, if the application is periodically polling for better services [9]. However this overhead has to be avoided regarding energy consumption and communication bandwidth. The concept sketched above will avoid this overhead.
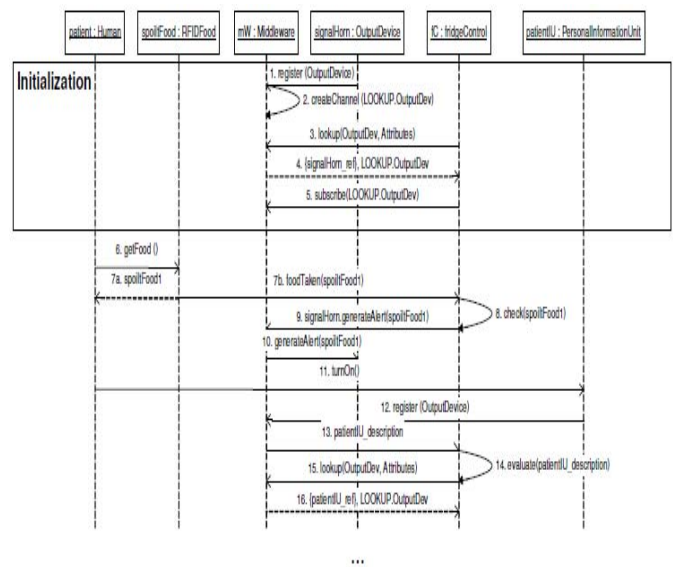

Figure 4.Exemplary sequence of dynamic integration within an AmI system.

VI. CONCLUSIONS

This position paper has proposed a reference middleware architecture for Ambient Intelligence Applications. A series of services have been discussed and a bottom-up approach has been elucidated. Moreover the tailoring support with respect to the hardware nodes available in a given situation has been outlined.

The work presented here is currently in an initial phase. The next activities planned subsume the proof of concept through the prototypical implementation (e.g. with CORBA or OSGi) of the middleware platform along with the refrigerator

application. The latter is to be extended soon with additional use-cases regarding context-sensitivity (e.g. the item information service can adapt expiry dates according to the temperature). Apart from that the request for adaptability will be investigated by the introduction of a variability service, which will supplement the dynamic integration service by managing the variability of computing nodes at run-time. Finally the long-term plans include the enrichment of the platform with additional services like security as well as the investigation of safety concerns.

ACKNOWLEDGMENTS

REFERENCES

[1] C. Mascalo, L. Carpa, W. Emmerich: Principles of Mobile Computing Middleware, Technical Report, Department of Computer Science,University College London, Great Britain, 2002

[2] M. Dertouzos: The Unfinished Revolution, Harper Business, 2002

[3] M. Weiser: The Computer for the Twenty-First Century, Scientific American, September 1991.

[4] Bilateral German-Hungarian Collaboration Project on Ambient Intelligence Systems Project Outline V2.2 April 2005.

[5] A. Geraldy, R. Gotzhein: Adaptive Integrated Protocols for Wireless Sensor Networks: Frame-Based Forward Error Control, Technical University of Kaiserslautern, 2004

[6] C. Decker, A. Krohn, M. Beigl, T. Zimmer : The Particle Computer System, Tele-cooperation Office (Teco) University of Karlsruhe, Germany.

[7] Mica Motes http://www.xbow.com/.

[8] J. Kahn, R. Katz, K. Pister: Emerging Challenges: Mobile Networking for 'Smart Dust', J. Comm. Networks, Sept. 2000, pp. 188-196.

[9] C. Bartelt, T. Fischer, D. Niebuhr, A. Rausch, F. Seidl, M. Trapp: Dynamic Integration of Heterogeneous Mobile Devices. In: Proceedings of the Workshop in Design and Evolution of Autonomic Application Software (DEAS 2005), ICSE 2005, May 21, 2005, St. Louis, Missouri, USA

[10] A. K. Dey: Providing Architectural Support for Building Context-Aware Applications, PhD Thesis, Georgia Institute of Technology, November 2000

[11] K. Pohl, G. Böckle, F. van der Linden: Software Product Line Engineering, Foundations, Principles, and Techniques. Berlin, Springer-Verlag, 2005.

102