# Improving Modularity, Interoperability and Extensibility in Ambient Intelligence

Marco Gomes, Davide Carneiro, André Pimenta, Milton Nunes, Paulo Novais, and José Neves

CCTC/DI - Universidade do Minho
Braga, Portugal
{marcogomes,dcarneiro,apimenta}@di.uminho.pt,pg22797@alunos.uminho.pt,
{pjon,jneves}@di.uminho.pt

**Abstract.** Ambient Intelligence (AmI) and its related fields emerged some years ago with the exciting promise of pervasive intelligence, magic interaction mechanisms, and everywhere availability. This promise would be materialized in homes that knew all about our habits and preferences, proactive workplaces to support people's work or personal digital assistants to improve our daily living in all aspects possible. This somewhat utopian vision, expected by many to have already taken place, remains unaccomplished and far from it. Many challenges still lay ahead which delayed and continue to delay the expected technological unravelling. In this paper we focus on the immense technological challenges of designing and implementing AmI Systems. Specifically, we propose a technological approach that will contribute to overcome some of these challenges by making developed AmI solutions more modular, interoperable, and extensible. This will result especially advantageous for large development teams or teams that span multiple institutions.

**Keywords:** Ambient Intelligence, Interoperability, Switchyard

## 1 Introduction

Ambient Intelligence is one of those sub-fields of Artificial Intelligence that stimulates our creativity. It results very easy for us to imagine scenarios in which the artefacts around us have intelligence or consciousness, constantly interact with us in a natural way and are always available. Some of these examples have moved from the imagination of book writers and movie producers to the paper or screen, to result in pieces describing a possible and very appealing future, one of the most popular examples being the futuristic world depicted in the film *Minority Report*. Here, Captain John Anderton interacts with a series of futuristic interfaces and intelligent tools to assist in his fight against (still to-be-committed) crimes. In the books, examples of an exciting future can be found, for example, in the fictional universe of *The Hitchhiker's Guide to the Galaxy*. In this world doors, for instance, are conscious (although their single ability is to feel valued when people use them).

While the second example is extracted from a humorous piece, the first allows to think quite seriously on the future that awaits us. Hopefully. Indeed, the actual broad implementation of even the simplest examples seen in these pieces seems still distant in time. Moving from fiction to more serious grounds, the same issue exists: the envisioned new world fostered by Ambient Intelligence and "foreseen" by Bogdanowicz et al. [7] is still far from reality.

Indeed, many of the technological requirements and challenges pointed out by the authors still remain nowadays or are only partly solved. Meanwhile, as depicted in the following section, new challenges emerge that need to be addressed for the sake of the reliability and acceptance of such systems. This paper makes an analysis of these challenges, with a particular focus on technological challenges. We propose an approach based on the novel SwitchYard framework to facilitate the development of more modular and extensible AmI systems. The main aim is to empower development efforts by distributed teams and the technology-independent integration of different systems or modules, to foster the development of AmI.

## 2 Current Challenges in AmI

As stated in the introductory section, there are several challenges that are, still today, holding AmI development back. One of these challenges, often disregarded by computer scientists (who form the backbone of AmI development) concerns privacy, identity and security issues. In [4] the authors make a thorough analysis of 70 AmI projects, principally in Europe, concerning these issues. They conclude that in general, current projects present a rather too sunny view of our technological future, ignoring or postponing dealing with some pressing issues. The authors also make an interesting reference to the SWAMI project (Safeguards in a World of Ambient Intelligence) which, against this trend, has constructed what they deemed "dark" scenarios [8], to show how things can go wrong in AmI and where safeguards are needed. As Rouvroy puts it, the challenge here is to preserve the individual freedom to build one's own personality without excessive constrains and influences while have control over the aspects of one's identity that one projects on the world [6].

Marzano, on a different view, looks at the cultural implications of an unregulated or indiscriminate growth of AmI, making a parallel with the industrial revolution [1]. As, at the time, more was (later proved to be) not necessarily better (take for instance consequences such as the pollution), right now, *smarter* may also not be necessarily better. Indeed, we may simply not want a smart juicer or a talking toaster.

But let us focus on the technological challenges that are still ahead. One the one hand, we have the challenges that are related with the physical constraints and nature of the necessary hardware. In [3] the authors examine the intricate relationship between the growing need for more computational and communicational power to support increasingly complex services and, at the same time, the

need for smaller, more lightweight and efficient devices. It is easy to understand how the objectives of these two fields conflict.

Another issue holding back a faster development of AmI is the scatter of research efforts. Indeed there are currently many different institutions doing research on very similar topics. When these institutions want to conciliate efforts they may find it difficult to do so since they use different technologies, standards or approaches. We believe that facilitating this integration and interoperability could result in a coming closer of different teams, whom could join efforts and more efficiently work together for the same goal. With this objective in mind, we are developing an open architecture to support an AmI system: open not only in the sense that it relies on open software but also, and most importantly, that it can easily integrate external services, as well as provide its own to external requesters. This architecture and its main advantages are described in the following sections.

## 3 Architecture

In our pursuit to develop an architecture that covers the main AmI technological needs, we first definite it at a conceptual level. The proposed architecture is logically divided into several packages that encapsulate a set of features and tasks. Figure 1 presents its high-level view, detailing the five packages that compose the system.

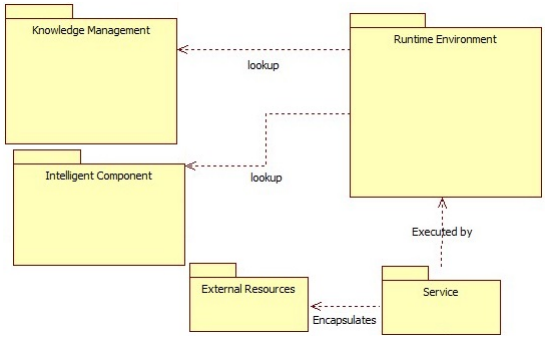


**Fig. 1.** High-level view of the architecture.

The Runtime Environment is the main component of the architecture, where system operations are executed. It is also through this environment that the remaining components are accessed. The Service component, executed by the Runtime Environment, contains all services of monitoring and data collection through the use of sensors, and is responsible for encapsulating all external resources collected by the system, represented by the External Resources. The tasks carried out by the Runtime Environment are also supported by queries

to components Knowledge Management and Intelligent Component. Knowledge Management supports the management of information and knowledge collected by the system. The Intelligent Component contains components of Intelligent Systems/Artificial Intelligence used in this architecture for data processing.

The Runtime Environment corresponds to the core component of the architecture where the main features are executed. It is also divided in two subcomponents, which are nonetheless interconnected: the Runtime Management and Runtime Platform. The first is a platform for managing processes running on the system. It is composed of a Data Manager that contains an interface to link the Knowledge Manager, an Execution Manager responsible exclusively for the management of the execution of system processes and the Service Directory that contains all information from the execution of services in the architecture. It should also be noted that the Runtime Management has an interface to connect the Service component. The Runtime Platform is a platform for the execution of the Runtime Environment. This is composed of a Service Bus that, through the interface provided by the Runtime Management, establishes communication between the two subcomponents and also a Service Execution Engine that represents the execution engine services in the architecture.

AmI systems are commonly described as electronic environments that seamlessly interact and adapt to human needs, in which people are surrounded by intelligent and intuitive interfaces embedded in all kinds of objects. To take full advantage of the information gathered ubiquitously from various sources in the environment there is a need for a software infrastructure that allows an easy integration, promotes interoperability, and focuses on extensibility. Considering these aspects, in this work we present an infrastructure to support an efficient approach for the development of AmI applications, following an approach based on Service Oriented Architectures (SOAs). Indeed, SOAs are being increasingly adopted in both the academic and industrial arenas, even to integrate Multi-agent Systems [2].

The more appropriate way of doing so is to adopt a Service-Component Architecture (SCA): a group of OASIS specifications that has become an industry standard. It is intended for the development of applications based on SOA, which defines how computing entities interact to perform work for each other. Originally published in November 2005, SCA is based on the notion that all the functions in an system should exist in the form of services that are combined into composites to address specific business requirements. In other words, it allows to build service-oriented applications as networks of service components. SCA is used for building service components, assemble components into applications, deploy to (distributed) runtime environments and reuse service components built from new or existing code using SOA principles. This approach is advantageous in AmI for the following reasons:

- Interoperable. Provides loose coupling allowing to integrate without need to know how components are implemented. Components can be written using any language, and can use any communication protocols and infrastructure

to link them, making it easier to integrate components to form composite applications.

- Maintainable. Composition of solutions is clearly described as declarative application of infrastructure services. Simplification for all developers, integrators and application deployers.
- Flexibility of Development. Service Components are easier to develop because the semantics of each independent Service Component are significantly less complex than the overall of a single, (relatively large) monolithic application; each Service Component can be developed by a different team of developers, each of whom focus only on their component without having to know the details of work done by others. Components can easily be replaced by other components and services can be easily invoked either synchronously or asynchronously.
- Reuse. Since each Service Component has well-defined interfaces, each component can be developed, tested and debugged independently of the other components. This not only speeds up project implementations but, in the case of well-designed Service Components, also leads to significantly enhanced reuse.
- Dynamic Deployment and Runtime Modification/Replacement. Service Components can be dynamically deployed to remote nodes at runtime, and components within a process can be easily replaced by new or updated components, further reducing the time taken to modify or change an existing process in response to business requirements.
- Configuration Management and Version Control. Service Components facilitate version control and dynamic configuration management, allowing fine-grained control over deployments across the enterprise.

SCA provides a good basis for AmI applications [9], it is in line with our architectural model and it fulfils major AmI deployment requirements by promoting late bindings at deploy time and runtime with the support of several relevant technologies including POJO, SOAP, REST, BPMN, BPEL, JMS, Camel or Rules services. But most of all it is currently supported by several major commercial and open source products such as Jboss Switchyard, IBM WebSphere or TRENTINO (C++).

From the several available implementations of SCA we have chosen JBoss SwitchYard since it is an open source solution in a relative mature state, and also enhances some of the SCA advantages. Specifically, Switchyard advocates transparency when running a service during its whole lifecycle. Important aspects such as connectivity, orchestration and routing do exist on SwitchYard in a modular format, which means one can deploy them in an independent way. Using a SwitchYard graphical user interface (Fig. 2), one can build visual models of the applications, that are meant to improve the software engineer's ability to comprehend and communicate the full composition of their applications and also to speed up development and integration projects.

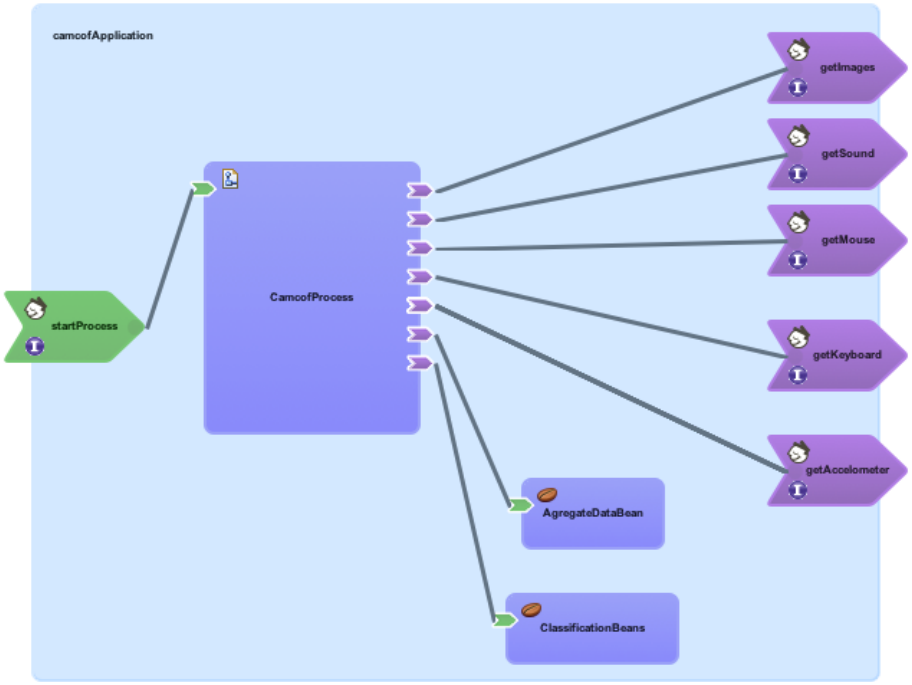


**Fig. 2.** Application composition using SwitchYard graphical user interface.

## 4 Case-Study of Context-aware Multimodal Communication system

This case study describes an application in which contextual information about the user is collected and used to detect states of stress and fatigue. The purpose is to enrich communication processes allowing for its users to communicate in ways that are closer to face-to-face communication. The estimation of stress and fatigue are based on the transparent analysis of the user's behaviour and interaction patterns [5]. In gathering data the following sensors were involved:

- **Accelerometer** - These devices, placed on the chair, keyboard and mouse, measure how the user is moving and the amount of force he is applying in the peripherals;
- **Mouse and Keyboard** - These devices provide information about how the user interacts with the peripherals (e.g. velocity of the mouse, typing rhythm, number of mistakes).
- **Microphone** - Microphones are used to measure the amount of noise in the vicinity of the user, allowing to perceive their social environment.
- **Video Camera** - An estimation of the amount of movement is calculated from the video camera. The image processing is based on difference techniques to calculate the amount of movement between two consecutive frames.

In order to materialize the architecture we developed a concrete instance for this specific application. All the described sensors are encapsulated by services running locally. Each of these services exposes different features using a Web Service interface. These services were integrated using SwitchYard allowing the service orchestration to support the automation of system processes by loosely coupling services across different applications. There is a clear separation between process logic and Web Services, providing the system with increased flexibility.

The main process was modelled using Business Process Modelling Notation (BPMN). In this standard (Fig. 3), a consumer service invokes a process flow via the service interface. The orchestration engine invokes services to process various service which in turn invoke further service requests until the workflow process is completed and results are provided to the service consumer. The service orchestration engine, a component of the SwitchYard, handles the overall process flows, calling the appropriate web services and determining the next steps to complete.

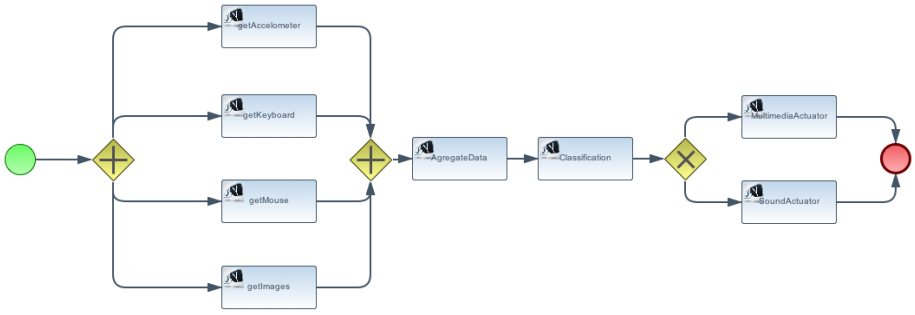


**Fig. 3.** The "camcof" process, view through BPMN

With this architecture it is possible to orchestrate the acquisition, transformation, and classification processes used to collect and to extract meaningful information. It is also possible to use different services (encapsulating different and disperse sensors), simultaneously or individually, coordinating all the elements of the main process.

## 5 Conclusions

Ambient Intelligence hasn't developed with the expected pace due to a number of challenges that were pointed out at the beginning of this paper. Many of these challenges still exist nowadays and include the difficulty in integrating and incorporating different approaches as well as the difficulty in developing highly modular systems that can be easily configurable and extended at need.

In this paper we presented an approach targeted at such scenarios. One that allows to build AmI systems in a decoupled way, without technological constraints, relying on a service-oriented approach. This approach is also distributed in the sense that services can be running anywhere in the world. For the manager, this is completely transparent. Nonetheless, high-level functionalities can be built when these services are combined using appropriate rules.

We believe that approaches such as this will not only make it easier to build larger-scale AmI systems but also increase the opportunity for researchers to more easily integrate their work, leading to more complex and AmI systems, with far richer functionalities. Moreover, services scattered around the world can transparently be used to achieve this goal in a far easier way.

## Acknowledgements

This work is part-funded by ERDF - European Regional Development Fund through the COMPETE Programme (operational programme for competitiveness) and by National Funds through the FCT - Fundação para a Ciência e a Tecnologia (Portuguese Foundation for Science and Technology) within project FCOMP-01-0124-FEDER-028980 (PTDC/EEI-SII/1386/2012).

## References

1. Aarts, E. H., Marzano, S. (Eds.). The new everyday: Views on ambient intelligence. 010 Publishers. (2003)
2. Aguero, J., Rebollo, M., Carrascosa, C., Julián, V.: MDD-Approach for developing Pervasive Systems based on Service-Oriented Multi-Agent Systems. Advances in Intelligent Computing and Artificial Intelligence Journal, Vol 1, No 6 (2013)
3. De Man, H.: Ambient intelligence: gigascale dreams and nanoscale realities. In Solid-State Circuits Conference. Digest of Technical Papers. ISSCC. IEEE International, pp. 29–35, IEEE. (2005)
4. Friedewald, M., Vildjiounaite, E., Punie, Y., Wright., D.: The Brave New World of Ambient Intelligence: An Analysis of Scenarios regarding Privacy, Identity and Security Issues. Security in Pervasive Computing. Proceedings of the third International Conference, SPC 2006, York, UK, April 18-21, 2006. Ed. Clark, J. A. et al.. Heidelberg, Berlin: Springer, pp- 119–133 (2006)
5. Pimenta, A., Carneiro, D., Novais, P., Neves, J.: Monitoring Mental Fatigue through the Analysis of Keyboard and Mouse Interaction Patterns. In Hybrid Artificial Intelligent Systems (pp. 222-231). Springer Berlin Heidelberg (2013)
6. Rouvroy, A.: Privacy, data protection, and the unprecedented challenges of ambient intelligence. Studies in ethics, law, and technology, 2(1) (2008)
7. Bogdanowicz, M., Scapolo, F., Leijten, J., Burgelman, J. C.: Scenarios for ambient intelligence in 2010 (pp. 3–8). Office for official publications of the European Communities (2001)
8. Wright, D.: The dark side of ambient intelligence. info, 7(6), pp. 33–51 (2005)
9. Giner, P., Pelechano, V.: An Architecture to Automate Ambient Business System Development. PProceedings of the European Conference on Ambient Intelligence. Heidelberg, Berlin: Springer, pp-240–257 (2008)