Embedding Agents within Ambient Intelligent Applications

Gregory M. P. O'HARE [a,1], Rem COLLIER [a] Mauro DRAGONE [a]
Michael J. O'GRADY [a] Conor MULDOON [a] and Alcides DE J. MONTOYA [b]
[a] *CLARITY: Centre for Sensor Web Technologies, University College Dublin, Ireland.*
[b] *Universidad Nacional de Colombia, Medellin Campus, Colombia*

Abstract. This chapter reflects upon the challenges that confront the deployment of Ambient Intelligence (AmI) applications. Ambient Intelligence demands that everyday artefacts be imbued with intelligent reasoning capabilities together with the capacity for collaborative intelligent behaviour. Traditional ambient devices do not provide the requisite computational platform to support such requirements. With the ongoing developments of ubiquitous devices, however, the situation is changing. This chapter discusses a software stack, which supports the needs of ambient applications that incorporate embedded intelligence.

Keywords. Wireless Sensor Networks (WSNs), Ambient Intelligence, Embedded Agents, WSN Middleware, Ambient Intelligence Software Stack.

Introduction

This chapter reflects upon the challenges that confront the deployment of Ambient Intelligence (AmI) applications. Ambient Intelligence demands that everyday artefacts be imbued with intelligent reasoning. Such artefacts are situated within an environment that is comprised of many heterogeneous devices, which offer the possibility of intelligent behaviour provided they operate collectively and engage in opportunistic collaborative behaviour. The need for distributed intelligence and the natural fragmentation of expertise, such that it relates to particular tasks and activities at particular locations and times, lends itself naturally to a Multi-Agent System (MAS) approach.

Multi-Agent Systems have traditionally been deployed in comparatively resource rich environments, such as desktop computers. The challenge of how to incorporate intelligence within computationally challenged devices, which typify ambient applications, is the focus of this chapter.

1. Ubiquitous Sensing

Wireless Sensor Networks (WSNs) are perceived as being the key enabling technology of ubiquitous sensing [25]. Yet how such networks will be manifested in real world

[1]Corresponding Author: Gregory O'Hare, School of Computer Science & Informatics, University College Dublin, Belfield, Dublin 4, Ireland; E-mail: gregory.ohare@ucd.ie.

scenarios remains to be seen. For the most part, such networks remain theoretical entities, in some cases demonstrated as prototypes, and many challenges remain to be overcome before widespread deployment and popular adoption can be expected. Naturally, researchers have continued to speculate about how ubiquitous sensing will ultimately be realised. Five usage paradigms are now briefly considered.

1.1. Pervasive Computing

Pervasive computing, or ubiquitous computing, was conceived in the early 1990s by the late Mark Weiser [31]. A pioneering vision, the key observation was that as computing technologies proliferated, they could be increasingly embedded in everyday artefacts, and would be accessible in a seamless and transparent fashion. As well as embedded computation, ubiquitous connectivity was perceived as a key element. The vision promised was enticing, and led to the establishment of a variety of initiatives in domains necessary to the fulfilment of the original vision. In the intervening time span, while significant progress has been made in a raft of technologies, pervasive computing has not as yet been validated as a technologically viable or commercial computing paradigm.

One concept closely associated with pervasive computing is that of context-aware computing [4]. All events occur within a context, for example, location, time-of-day, and so on. If context can be captured and interpreted meaningfully, a more holistic user experience may be possible. It must be observed that context as a construct is not limited to the physical world; rather it can include other salient factors such as personal attributes and social situation amongst many others.

Significant research effort has been expended towards making pervasive computing a reality. Yet its realization in practice would appear some way off. This is not a negative criticism of what researchers have achieved; rather it is an acknowledgment that the obtainment of the pervasive computing vision in practice is far more difficult that originally envisaged. Some practical reasons as to why pervasive computing has not been manifested as originally envisaged include security, privacy, lack of sophisticated semantic models and a lack of core system engineering practices. More recently, the issue of design has received increased attention; specifically the objective of good design should be to make pervasive computing systems available both physically *and* cognitively [30].

1.2. Mobile Computing

Given the prevalence of laptops, it is difficult to believe that mobile computing was once a theoretical concept. First realized in the 1980s, mobile computing is now considered the predominant computer usage paradigm. When viewed from a miniaturization perspective, pervasive computing was the logical next step. However, it was a development in another domain that was to prove pivotal to the success of mobile computing: telecommunications.

Wireless telecommunications is an undisputed success story, and has radically overhauled an industry that had provided a single voice carrier service for decades. Today, data is the major revenue generator, resulting from the tranche of innovative services that everybody is familiar with. The subscriber has been the ultimate beneficiary of these developments, with access to computing services in an almost on-demand fashion. Indeed, the ubiquity of mobile computing could be regarded as a singular practical manifestation of the pervasive computing vision.

More recently, the potential of the physical mobile phone handset as a potential sensor platform has been explored [16]. The current generation of mobile phones incorporates a suite of embedded sensors of various hues and sophistication - accelerometers, cameras and positioning technologies amongst others. MobSens [13] is just one of a number of documented platforms that demonstrates the viability of the mobile platform as a mobile sensing platform. Indeed, the potential of this platform for ubiquitous sensing has formed the basis of crowd sourcing [29] or participatory sensing [1] where large numbers of people can contribute to a data collection effort. Recent advances have resulted in so-called *citizen* observatories where large numbers of citizens contribute opportunistically sensed data via their mobile sensing platform. Such data is often incomplete, and contradictory; however, its sheer volume compensates for its inherent inaccuracies.

1.3. Internet of Things

Internet of Things (IoT) [9] envisages everyday items being able to identify themselves and communicate information about themselves. In some instances, computation, to a greater or lesser degree, may be supported. All devices would of course be connected to the internet. Obviously, this vision was inspired by the success of the internet. The Web-of-Things (WoT) [5] builds on this vision, integrating devices or "things" into the WWW, modelling them as WWW resources and making them available using standardized WWW techniques. The Sensor Web [10] construct is broadly similar in scope and objectives.

One of the key problems facing IoT was that of the address space; However, IPv6 resolves this issue. Ongoing research standardization activities focuses on making IPv6 more suitable for inherently low power sensor networks [6]. A more serious problem that must be overcome is the need for a modular software design approach that supports heterogeneity in a range of forms and is yet inherently scalable [27].

1.4. Ambient Intelligence

Ambient Intelligence (AmI) [21] as a paradigm was conceived in the late 1990s. It was motivated, at least partially, by the observation that pervasive computing, by now a decade old, was not materializing in the manner originally envisaged. Then, as now, no pervasive computing application could be acquired off-the-shelf; they remained in the laboratories as prototypes. AmI assumes the existence of ubiquitous computing and communications infrastructures. It sees the problem as being one of interaction, and proposes the adoption of Intelligent User Interfaces (IUIs) as a means of facilitating interaction between users and embedded pervasive computing systems. By now, and even up to the present, significant research had focused on the computing element. The end-user was neglected, even though the original ubiquitous computing manifesto considered intuitive interaction essential, and was unapologetic concerning the overriding goal of the manifesto: applications.

IUIs [17] have been the subject of research for a number of years, even predating the AmI initiative. Though primarily a Human Computer Interaction (HCI) discipline, it incorporates Artificial Intelligence (AI), psychology, and cognitive science amongst others. It does not focus on making the device itself smart; rather it focuses on the interaction. Thus brain computer interfaces, multi-modal interfaces, gesture recognition

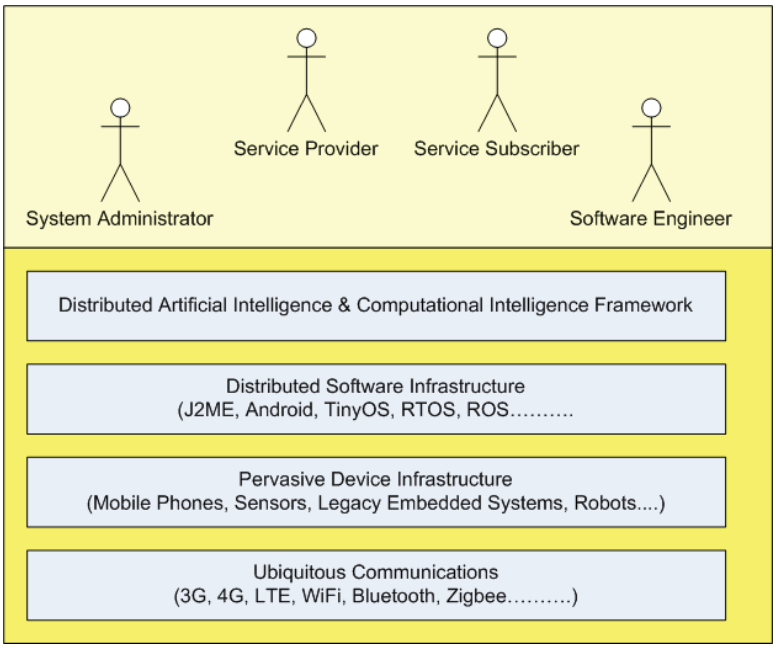

Figure 1. Intelligent User Interfaces should encapsulate sufficient functionality to meet the needs of a diverse user population.

and natural language understanding could all be harnessed for IUI development. Note that AmI itself does not mandate any particular, or combination of, technologies. Conventionally, there is a tendency to treat the user interface as synonymous with the interface that the archetypical user will perceive; this need not be the case. An AmI interface should encapsulate within itself the capability to manifest itself to a variety of different stakeholders as their requirements and roles dictate (Figure 1).

Research challenges identified for AmI include activity modelling, recognition, and prediction [3]. The requirements for robust, power efficient sensor platforms has also been acknowledged.

1.5. Ubiquitous Robotics

Ubiquitous robots [11] [15], that is, robots embedded in ubiquitous and intelligent environments, are the most recent class of networked robot applications motivated by the increasing interest raised by ubiquitous computing scenarios.

Multi-robot systems already break the more historical view of robots as monolithic control systems running on specific computational units and in charge of their own *hard-wired* set of sensors and effectors. In particular, they replace this view with one in which *robotic* resources are also perceived as belonging to, and forming integral parts of, larger networks. Research in Ubiquitous Robotics further such a vision by explicitly addressing interoperability issues between heterogeneous hardware and software components, and also by tackling open environments, where resources, users, robots and other types of artificial agents can dynamically join and leave the system.

Incorporating robots in Wireless Sensor Networks (WSNs) or Wireless Sensor and Actuator Networks (WSANs) solutions is also an important and extended challenge for robotics and a key enabler for a range of advanced robotic applications, including Ambient Assisted Living (AAL) and environmental monitoring. The increasing number of WSNs/WSANs deployments provides a powerful business case for supporting this type of hybridisation. Robots can act as user-friendly interfaces to these systems and also enhance them by providing important benefits such as sensor deployment, calibration, failure detection and power management.

The same approach also has the potential to solve many problems that hinder the spread of pure robotic solutions. One example is the difficulty of understanding their environment with noisy and imprecise sensor capabilities. In contrast, ubiquitous robotics in general advocates the augmentation of the robots' communication and interaction capabilities with those afforded by sensors and actuator embedded within the environment.

Ubiquitous service infrastructures can also act as a service provider for the robot(s). A robot, for instance, can query the infrastructure to find which services are supported (e.g. localization) and subsequently decide to request the service, for example, receiving positional updates, reporting its own position, and the position of other agents (robots and humans) operating in the same environment [11].

Finally, the domestic service robot described in [8] is a good example that demonstrates the combination of all these abilities by checking the user's e-mail, accessing the sound captured by wireless microphones, tuning the TV onto the user's favourite channel, and accessing the Internet for retrieving stories to narrate to children.

2. Towards Distributed Embedded Intelligence

Traditionally, systems that harnessed computational or artificial intelligence techniques were usually centralised. Even if the nature of the application suggested a distributed approach, a critical constraint was the computational resources available. Internet technologies, grid computing and cloud-based services collectively offer potential solutions; however, at the extremity of many networks, there exists a suite of resource-poor platforms upon which deploying intelligent solutions is a formidable challenge. Such platforms include mobile phones, home networked devices and other legacy embedded systems, and of course sensors. The latter are of most interest as they are perceived as a key enabling technology of AmI. Computational Intelligence can be harnessed in diverse ways in sensing platforms, for example, routing, scheduling and data fusion [14]. However, harnessing sophisticated intelligence algorithms and techniques is not a viable proposition on the leaf nodes and base stations of sensor networks due to the inherent paucity of computational resources on such platforms. One potential solution to this problem would be to distribute the intelligence throughout the network, an approach that has proved successful elsewhere [24] [22] [23].

What would be achieved by infusing intelligence throughout a sensor network? From an application domain or services perspective, enabling key decision making functionality at extreme nodes can facilitate a more modular design, resulting in a network that is robust and fault-tolerant, thus enabling a better quality of service. From a system management perspective, distributed decision making could prove indispensible for achieving autonomic behaviour. Ensuring operational longevity is also a fundamental

objective; intelligent techniques offer potential here. In totality, it can be seen that there are a number of objectives and demands on an arbitrary sensor network, both from a service, management and end-user perspective. However, these may conflict and in some cases, must be resolved at runtime. Distributed embedded intelligence offers one tool through which designers and operators may ensure that their operational policies are implemented at least in a best-effort basis at times of conflict.

2.1. Agents for Constrained Devices

In realizing distributed embedded intelligence, a multi-agent system offers a compelling metaphor. One instrument by which to realise this metaphor is Agent Factory [2]. The Agent Factory Framework is an open source collection of tools, platforms, and languages that support the development and deployment of multi-agent systems. The framework is broadly split into two parts:

1. support for deploying agents on laptops, desktops, and servers;
2. support for deploying agents on constrained devices such as mobile phones and WSN nodes.

The former support is delivered through Agent Factory Standard Edition (AFSE) [2], and the latter support is delivered through Agent Factory Micro Edition (AFME) [18].

Agent Factory incorporates a communications infrastructure, which supports an Agent Communication Language (ACL) that is shared and understood by all agents. The necessity to support inter-agent communication, in general, has led to the development of an international ACL standard, which Agent Factory is in broad compliance with, that have been ratified by the Foundation for Intelligent Physical Agents (FIPA). FIPA has, since 2005, been a standards body affiliated with the Institute of Electrical and Electronic Engineers (IEEE). It forms an autonomous standards committee with the objective of facilitating interoperability between agents and other non-agent technologies.

Agent Factory is implemented in the Java programming language. In its latest incarnation, the framework has been restructured to facilitate the deployment of applications that consist of multiple agents that employ a diverse range of agent architectures. As such, the framework has become an enabling middleware layer that can be extended and adapted for different application domains.

AFME is a light weight agent platform that has been designed for devices consistent with the Java Micro Edition (JME) Constrained Limited Device Configuration (CLDC) specifications. AFME was originally used for the development of applications for cellular digital mobile phones, but has since been ported to the leaf nodes of a WSN and specifically to Sun SPOT motes. There were several reasons for the development of AFME, as opposed to using the original platform, which was designed for desktop environments, for both embedded devices and desktop machines. CLDC is the de facto standard Java platform for constrained devices. The original framework could not operate in a CLDC environment and therefore could not be used for the majority of embedded devices. Furthermore, the software footprint of the original framework was too large for constrained devices and the original framework had dependencies on functionality not supported in CLDC. Conversely, CLDC devices, such as the Sun SPOT, provide support for function-

[2] http://www.agentfactory.com/index.php/Main_Page

ality that is not part of standard Java, such as a capability to transmit messages over a radio.

In deploying agents on sensor nodes, developers are faced with a number of problems; perhaps the most obvious is the limited spatiotemporal computational and power resources available. It is for this reason that initial agent frameworks developed for WSNs were based on a weak notion of agency [32], whereby agents did not possess reasoning capabilities, the canonical example being that of Agilla [7]. Other more recent approaches focus on particular algorithms for agent interaction and coordination [28] [20] [19]. AFME agents are loosely based on the BDI notion of agency [26]. For the practical deployment of such agents on embedded and mobile devices, it is essential that resources are not squandered and are used in an intelligent and prudent manner. BDI languages are, for the most part, declarative, although in practice most of them are used in conjunction with imperative components. The imperative components are usually written in object-oriented programming languages, such as Java or C++. The agent languages represent a logical abstraction. Various interpreters can be built for them so that they can be used in different environments. In developing agents for sensor networks, the developer could be tempted to develop everything at an imperative level, but this would be little better than using the weak approach to agency. At the other extreme if the agent does too much reasoning, resources will be wasted and the system will be unresponsive. AFME supports both approaches. The decision as to whether something should be implemented at declarative or imperative level is not clear cut and ultimately comes down to the experience and knowledge of the developer. There are no failsafe development methodologies that ensure a good agent design. The design decisions made are of significant importance in the WSN domain where resources are extremely scarce. It is our belief that it is no longer the case that the BDI model of agency is too computationally intensive for resource constrained devices. This is primarily due to developments in computing technology, improvements in the efficiency of algorithms, and the dissemination of good design practices and the knowledge of algorithms to developers.

Each agent in AFME consists of a set of roles, which are adopted at various points throughout execution. Each role consists of a trigger condition and a set of commitment rules. Once an agent adopts a belief that matches the trigger, the role is adopted and the set of commitment rules are added to the agent's mental state. Subsequently, on each iteration of the agent's control process, the commitment rules are evaluated until either the role is retracted or the agent is terminated. The set of commitment rules adopted when a role is triggered specify the conditions under which commitments are adopted for the role. Once commitments have been adopted, the agent commences the commitment management process. Various arguments are passed to the commitment when it is adopted, such as the time at which it should commence, to whom the commitment is made, and the maintenance condition of the commitment. An identifier is specified, which acts as a trigger for the plan or primitive action to be performed. In subsequent iterations of the control algorithm, the commitment is invoked subject to the arguments specified.

3. The Need for a Middleware

Deploying WSN offers significant challenges in terms of heterogeneity of hardware, data and applications. To illustrate these challenges, let us consider an urban pollution

monitoring application that combines: roadside pollution sensors, automated weather stations, twitter feed sensors, weather feeds, and satellite images.

Hardware diversity comes from the need to combine physical sensors such as the pollution sensors and weather station with cyber sensors for the twitter feeds, the weather feeds and the satellite images. The mechanism for controlling and interacting with each sensor can range widely in terms of both the mode (how to interact with the sensor) and form (the functionality supported by the sensor). Further, such a system would need to be "future proofed" in that, it must be designed to support new sensing devices and data feeds as they become available. Such new devices may offer different sensing modalities or may require new data processing services be properly integrated into the deployed application.

Data diversity arises in terms of the type of data, the frequency of update and the scale of data. For example, the pollution sensors may sample once per hour, whereas the twitter sensor is real time. Conversely, the pollution sensors provide very structured and fixed data, whereas the twitter sensor data is far more diverse and must be processed to ensure that irrelevant information is discarded.

Application diversity refers to the relatively unlimited range of applications which would benefit from access to the sensor web. For example, the weather service sensor identified above could be utilized by applications as diverse as: home support services, intelligent traffic management and building management. From a software engineering point of view, it would be expected that all of these applications would be able to reuse the same weather service sensor implementation. This offers potential issues in terms of the update of such a weather service sensor as any change to its API will impact on all the applications within which it has been used.

However, the issue is not restricted to just the need to support such deployments in isolation, but also to support multiple applications in a single WSN deployment. Consider, for example, a home-based WSN. It is unreasonable to assume that a homecare monitoring application would utilize a separate deployment to a home security application given the commonality of required sensors as, at the very least, both would require motion detection sensors to track movement and window and door sensors to detect open and closed doors and windows. Not allowing a single deployment for both applications would result in duplication of sensors (you would need two motion tracking sensors in every room, and two contact sensors on every window and door), duplication of the control hardware (you would need two control boxes on the wall, one running the homecare application and one running the security application), and duplication of interface hardware (you would need a box to control your security system and a separate box to control your homecare application). All that is left is to consider what happens when a third or fourth application is required.

In practice, such diversity can only be solved in one way - through the use of a common middleware that allows:

- easy integration of heterogeneous sensor types;
- minimal configuration of sensors for ease of deployment;
- support for multiple applications with diverse quality of service requirements;
- mechanisms to support adaptability, scalability, and extensibility.

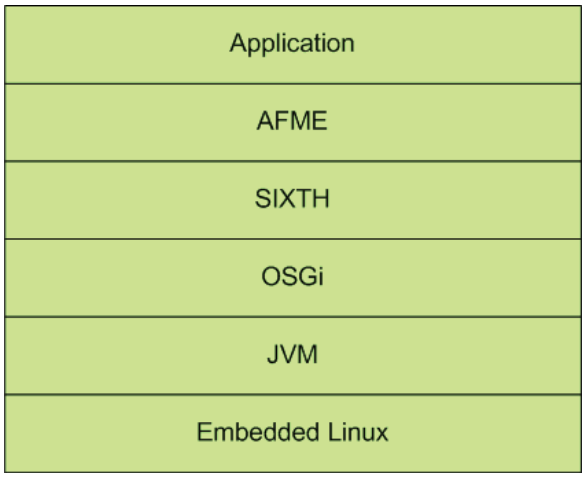

Figure 2. Ambient Intelligence Software Stack

4. The Software Stack for Ambient Intelligence

In order to deliver Ambient Intelligence on computationally challenged devices, we propose the layering of a set of key software components which we refer to as the Ambient Intelligence Software Stack (Figure 2).

Given the requirements for sensor middleware outlined above, it is vital that any proposed middleware solution be based on a model that is both simple to use and flexible. Further, it is clear that any such solution must embrace and manage the underlying diversity through the provision of appropriate mechanisms and structures.

Component-based approaches [12] offer a natural solution to handling this diversity whereby software systems are decomposed into a set of interacting components each of which encapsulates a specific system concept / function. Such a decomposition allows developers to reuse a common core set of components, reducing both the development time and the development cost. Further, when designed effectively, component-based approaches allow developers to reuse code for specific sensors across multiple applications, further reducing development time and cost.

One such component framework is the Open Services Gateway initiative (OSGi) framework. OSGi is the technology that underpins the well known Eclipse platform. It offers an exciting approach to developing middleware solutions due to its support for dynamic loading and updating of components, known as *bundles*, at runtime. Computation in OSGi is realised through the invocation of services that are made available by individual bundles through their advertisement via a shared *service registry*.

To demonstrate the value of component frameworks in the realisation of a WSN middleware, we now introduce SIXTH, a scalable, extensible, intelligent OGSi-based middleware for the Sensor Web, developed within CLARITY: Centre for Sensor Web Technologies, at University College Dublin (UCD). The principle goals of SIXTH are to support:

- seamless sensing of the Sensor Web, combining physical sensors based on technologies like Tiny OS or Squawk (Java); and cyber sensors that allow sensing of online data streams, such as Twitter, Facebook or RSS;

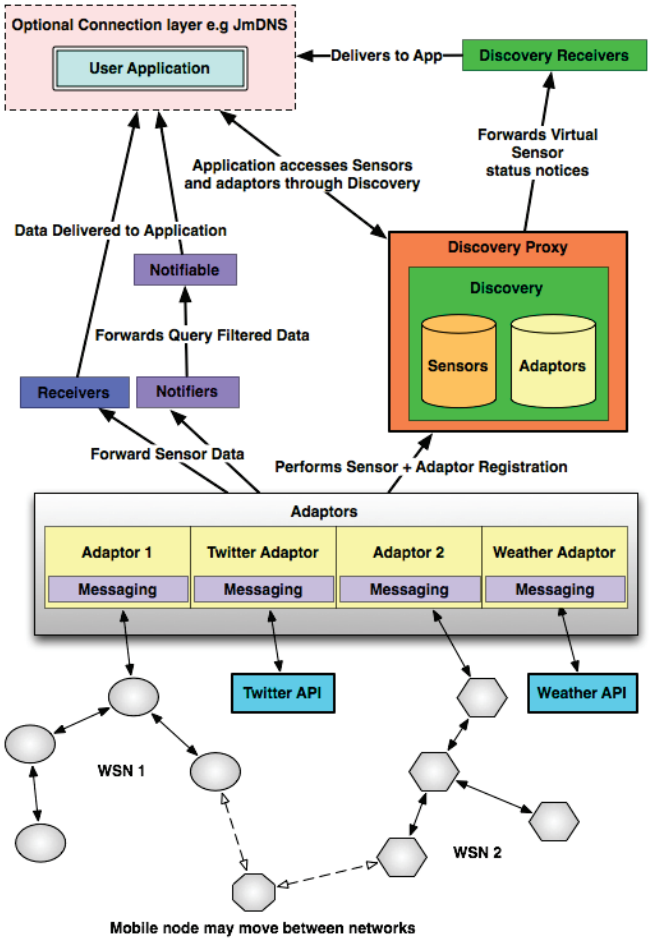

Figure 3. SIXTH: Middleware for the Sensor Web

- the development and deployment of dynamic and adaptive sensor networks that require runtime re-tasking of sensors and in-situ intelligence.

As can be seen in Figure 3, SIXTH is built around three basic components: a set of adaptors that implement a standardised interface to specific sensor systems; a discovery service, that advertises the availability of both adaptors and sensors; and a receiver/notifier mechanism that supports streaming of, possibly filtered, sensor data.

To standardise the interface to the adaptors, an abstract model of a sensor has been applied. In this model, sensors are represented as configurations (a set of properties) and the adaptor provides basic methods to allow the setting and getting of properties. Some properties in the model are mandatory (i.e. they apply to all sensors), whereas other properties are specific to the sensor type. Some properties are read only (e.g. the latest sensor readings), whereas other properties can be both set and got (e.g. duty cycle, sampling frequency). It is the responsibility of the adaptor to map set/get operations onto messages that trigger the appropriate update on the corresponding sensor. In this way,

we do not force a specific model on the sensor implementation, but instead localise the mapping between the sensor model and the SIXTH model to the adaptor.

Once created, the first action of an adaptor is to register itself with the discovery service. This makes the adaptor accessible to the application. This enables set/get operations to be applied to a sensor network as a whole (e.g. setting the duty cycle of all sensors). As individual sensors are discovered by the adaptor (in the case of cyber sensors "discovery" equates with "creation"), details are passed to the discovery service.

The role of the discovery service at this point is to act as a broker. Applications register with the service which then sends updates relating to the discovery of new sensors and adaptors. Where appropriate, the application can use the discovery service to gain access to specific sensors / adaptors. Access to individual sensors and adaptors is constrained via a security mechanism that is built into the discovery service.

Once the application has access to a sensor/adaptor, it is able to reprogram either individual sensors or all of the sensors connected to a given adaptor. This is realized at two levels: smaller adjustments are supported through the modification of sensor configurations, for example, changing the sampling frequency or modifying the alert thresholds; while larger adjustments are supported through our adoption of OSGi, which provides functionality for the dynamic installation and updating of bundles.

While the discovery service enables access to individual sensors/adaptors, access to the sensor data stream is managed via a receiver/notifier model. The basic mode of access is via a receiver. Receivers are created by the application and act as a collection point for sensor data. The receiver data stream is unfiltered and includes all the sensor data generated by all of the adaptors on a given SIXTH node. Custom data streams can be created through the notifier mechanism, which allows the application to specify the type of sensor data that they wish to receive.

In-situ intelligence is realized through the integration of Agent Factory Micro Edition (AFME), a lightweight agent platform for highly-constrained Java-enabled devices. AFME delivers state-of-the-art decision-making and coordination mechanisms that enable intelligence to be embedded within SIXTH-based deployments.

At the application level, zero-configuration is supported through the use of jmDNS, a Java-based implementation of multi-cast DNS that can be used for service registration and discovery in local area networks, with SIXTH. Individual SIXTH nodes advertise their presence as a URL that corresponds to the nodes remote management API. This API enables applications to request (possibly filtered) sensor feeds, re-tasking of sensors, and access to general information about the state of the underlying sensor network.

SIXTH is available from SourceForge [3], and currently includes packages that provide support for physical sensors that are based on technologies like Tiny OS or Squawk (Java), and support for cyber sensors that allow sensing of online data streams, such as Twitter.

5. A Practical Example of an Embedded Intelligent Solution

Middleware such as SIXTH provides a viable solution to seamlessly integrate robots in ubiquitous spaces comprising WSNs, Wireless Sensor and Actuator Networks (WSANs), and also home automation systems and Internet-based service infrastructures.

[3] http://sourceforge.net/projects/sixth/

In order to harness SIXTH 's features and illustrate its ability to support these type of ubiquitous robotic applications, we are using the Ubirobot test-bed hosted in CLARITY's office environment. The test-bed integrates a collective of mobile robots.

The simple use cases we target are designed to exercise functional and non-functional requirements commonly posed by the typical ubiquitous robot applications in our chosen domain, respectively:

- support reliable and robust robot navigation tasks in an office environment;
- interact with heterogeneous WSAN resources and heterogeneous networks;
- support open systems, in which resources (sensors), users, robots and other artificial agents can join and leave at run-time, for instance, as result of mobility, upon entering a new environment, re-configuration, component failure and network disruption;
- enable system interoperability, e.g. to support replacement of resources, their maintenance/update, and to maximize component re-use across different deployments.

Each robot is equipped with state of the art sensors, including both optical and 3D cameras, and laser and sonar range finders. In order to control each robot we employ a hybrid control architecture. In the lower-level, we employ standard robotic software, such as the Robotic Operating Systems (ROS) used to interface with the robots' hardware and to deliver common robotic skills, such as safe navigation, path planning and localization. In the higher-level of the control architecture, AFME is used to co-ordinate, sequence and configure these robot skills, and also to interact with other robots and agents via Agent Communication Languages (ACLs) over the WiFi network.

Figure 4 helps illustrating how such a control architecture can be extended with the SIXTH middleware, for instance, in order to support the introduction of a new robot with a mission to carry out an inspection of the status of all the sensors in the environment. The role of the architecture in such a use case scenario is illustrated in more detail in the remainder of this section.

A *Communication Agent* in the AFME layer keeps track of and interacts with the communication channels and information sources available to the robot by using a number of perceptors and actuators components. The*jmDNSPerceptor* leverages a *jmDNS* client to discover the AFME directory service, and also to monitor all the WSANs connected to the SIXTH middleware and accessible via the WiFi network. The *TinyOSPerceptor* and the *TinyOSActuator* are used, respectively, to receive and transmit message packets over the ZigBee network through a mote interfaced with one of the robot's USB ports. Another actuator, the *TinyOSProgrammerActuator*, is used to re-program the mote by flashing program binary images over the USB port. Once these resources have been discovered, other agents in the robot control system interact with them through the Communication Agent, without having to deal with the underlying network, messaging protocols and programming languages.

By way of example, imagine that the newly introduced robot does not have a priori knowledge of the new environment, its layout, its places, and the route to follow during its inspection. In addition, its autonomous localization capabilities are not effective in the new environment (for instance, the Computer Science building in UCD has a bridge with glass walls that the robot's laser fails to see). Fortunately for the robot, the SIXTH middleware provides a mapping service the robot can use to acquire the map of the

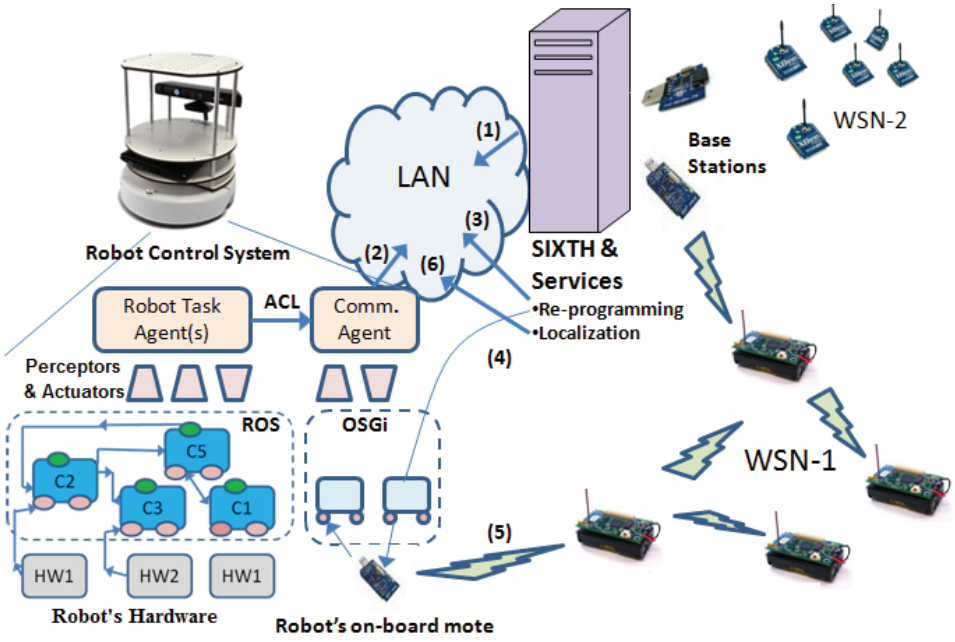

Figure 4. An Ubiquitous Robot System using AFME and SIXTH

environment and the route to be used to conduct an inspection of the building. It also provides location information by using the Radio Signal Strength Index (RSSI) measured from the robot's mote to the motes installed in the test-bed. However, in order to be used, this service requires the installation of a new TinyOS program on the robot's mote, which will then transmit RSSI data to be analyzed by the service.

The combination of AFME and the SIXTH middleware supports this use case with the following sequence:

1. After its introduction to the Clarity test-bed, via ACL-based communication, the robot control system discovers that it may need localization help and that the SIXTH middleware can provide it.
2. The Communication Agent informs the SIXTH middleware of the type of mote installed on the robot.
3. The re-programming service on the SIXTH middleware automatically generates a binary image for a TinyOS RSSI-based localization program, which is compatible with the robot's mote, publishes it on the SIXTH Web Server, and notifies the robot's Communication Agent of the URL to be used to download it.
4. The robot's Communication Agent downloads the binary image and flashes it on the mote.
5. Soon after that, the robot's mote starts transmitting to the WSN network, which collects the RSSI data.
6. The RSSI data is used by the localization service, which starts updating the robot of its estimated location.

6. Conclusion

Delivering distributed intelligence within harsh constraints and heterogeneous environments are a prerequisite to realising AmI applications that support real-time embedded sensing and dynamic actuation. This chapter has introduced an Ambient Intelligence Software Stack as a layering approach to the software cocktail necessary to support collaborating intelligent artefacts. This software stack incorporates a middleware for Wireless Sensor Networks (WSN), SIXTH, together with a shrink wrapped Multi-Agent Systems platform Agent Factory Micro Edition (AFME). This combination facilitates the easy integration of the diverse range of artefacts necessary for the construction of arbitrary AmI environments. The operation of this stack has been illustrated through a ubiquitous robotics scenario.

Acknowledgements

This work is supported by Science Foundation Ireland (SFI) under grant 07/CE/I1147. The authors would also like to acknowledge the support of the Irish Research Council for Science, Engineering and Technology and the European Commission Marie Curie Actions.

References

[1] J. Burke, D. Estrin, M. Hansen, A. Parker, N. Ramanathan, S. Reddy, and M. B. Srivastava. Participatory sensing. In *In: Workshop on World-Sensor-Web (WSWï£¡06): Mobile Device Centric Sensor Networks and Applications*, pages 117–134, 2006.

[2] R. Collier, G. O'Hare, T. Lowen, and C. Rooney. Beyond prototyping in the factory of agents. In M. Marík, J. P. Müller, and M. Pechoucek, editors, *Multi-Agent Systems and Applications III*, volume 2691 of *Lecture Notes in Computer Science*, pages 1068–1068. Springer Berlin / Heidelberg, 2003.

[3] D. J. Cook, J. C. Augusto, and V. R. Jakkula. Ambient intelligence: Technologies, applications, and opportunities. *Pervasive and Mobile Computing*, 5(4):277 – 298, 2009.

[4] A. K. Dey. Understanding and using context. *Personal Ubiquitous Comput.*, 5:4–7, January 2001.

[5] T. S. Dillon, H. Zhuge, C. Wu, J. Singh, and E. Chang. Web-of-things framework for cyberï£¡physical systems. *Concurrency and Computation: Practice and Experience*, pages n/a–n/a, 2010.

[6] M. Durvy, J. Abeillé, P. Wetterwald, C. O'Flynn, B. Leverett, E. Gnoske, M. Vidales, G. Mulligan, N. Tsiftes, N. Finne, and A. Dunkels. Making sensor networks ipv6 ready. In *Proceedings of the 6th ACM conference on Embedded network sensor systems*, SenSys '08, pages 421–422, New York, NY, USA, 2008. ACM.

[7] C.-L. Fok, G.-C. Roman, and C. Lu. Agilla: A mobile agent middleware for self-adaptive wireless sensor networks. *ACM Trans. Auton. Adapt. Syst.*, 4:16:1–16:26, July 2009.

[8] Y. Fujita. Personal robot PaPeRo. *Journal of Robotics and Mechatronics*, 14(1):60–63, January 2002.

[9] N. Gershenfeld, R. Krikorian, and D. Cohen. The internet of things. *Scientific American*, 291(4):76–81, October 2004.

[10] P. B. Gibbons, B. Karp, Y. Ke, S. Nath, and S. Seshan. Irisnet: An architecture for a worldwide sensor web. *IEEE Pervasive Computing*, 2:22–33, 2003.

[11] Y.-G. Ha, J.-C. Sohn, Y.-J. Cho, and H. Yoon. Towards a ubiquitous robotic companion: Design and implementation of ubiquitous robotic service framework. *ETRI Journal*, 27(6):666–676, 2005.

[12] G. T. Heineman and W. T. Councill, editors. *Component-based software engineering: putting the pieces together*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2001.

[13] E. Kanjo, J. Bacon, D. Roberts, and P. Landshoff. Mobsens: Making smart phones smarter. *Pervasive Computing, IEEE*, 8(4):50 –57, oct.-dec. 2009.

[14] R. Kulkarni, A. Forster, and G. Venayagamoorthy. Computational intelligence in wireless sensor networks: A survey. *Communications Surveys Tutorials, IEEE*, 13(1):68 –96, quarter 2011.

[15] A. LaMarca, W. Brunette, D. Koizumi, M. Lease, S. B. Sigurdsson, K. Sikorski, D. Fox, and G. Borriello. Plantcare: An investigation in practical ubiquitous systems. In *UbiComp '02: Proceedings of the 4th international conference on Ubiquitous Computing*, pages 316–332, London, UK, 2002. Springer-Verlag.

[16] N. Lane, E. Miluzzo, H. Lu, D. Peebles, T. Choudhury, and A. Campbell. A survey of mobile phone sensing. *Communications Magazine, IEEE*, 48(9):140 –150, sept. 2010.

[17] M. T. Maybury and W. Wahlster, editors. *Readings in intelligent user interfaces*. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1998.

[18] C. Muldoon, G. M. P. O'Hare, R. W. Collier, and M. J. O'Grady. Towards pervasive intelligence: Reflections on the evolution of the agent factory framework. In A. El Fallah Seghrouchni, J. Dix, M. Dastani, and R. H. Bordini, editors, *Multi-Agent Programming:*, pages 187–212. Springer US, 2009.

[19] C. Muldoon, G. M. P. O'Hare, and M. J. O'Grady. Collaborative agent tuning: Performance enhancement on mobile devices. In O. Dikenelli, M.-P. Gleizes, and A. Ricci, editors, *Engineering Societies in the Agents World VI*, volume 3963 of *Lecture Notes in Computer Science*, pages 241–258. Springer Berlin / Heidelberg, 2006.

[20] C. Muldoon, G. M. P. O'Hare, M. J. O'Grady, and R. Tynan. Agent migration and communication in WSNs. In *Proceedings of the 2008 Ninth International Conference on Parallel and Distributed Computing, Applications and Technologies*, pages 425–430, Washington, DC, USA, 2008. IEEE Computer Society.

[21] H. Nakashima, H. Aghajan, and J. C. Augusto. *Handbook of Ambient Intelligence and Smart Environments*. Springer Publishing Company, Incorporated, 1st edition, 2009.

[22] M. J. O'Grady, G. M. P. O'Hare, J. Chen, and D. Phelan. Distributed network intelligence: A prerequisite for adaptive and personalised service delivery. *Information Systems Frontiers*, 11:61–73, March 2009.

[23] G. M. P. O'Hare, M. J. O'Grady, R. Collier, S. Keegan, D. O'Kane, R. Tynan, and D. Marsh. Ambient intelligence through agile agents. In Y. Cai, editor, *Ambient Intelligence for Scientific Discovery*, volume 3345 of *Lecture Notes in Computer Science*, pages 286–310. Springer Berlin / Heidelberg, 2005.

[24] G. M. P. O'Hare, M. J. O'Grady, R. Tynan, C. Muldoon, H. R. Kolar, A. G. Ruzzelli, D. Diamond, and E. Sweeney. Embedding intelligent decision making within complex dynamic environments. *Artif. Intell. Rev.*, 27:189–201, March 2007.

[25] D. Puccinelli and M. Haenggi. Wireless sensor networks: applications and challenges of ubiquitous sensing. *Circuits and Systems Magazine, IEEE*, 5(3):19 – 31, 2005.

[26] A. Rao and M. Georgeff. Bdi agents: from theory to practice. In *Proc 1st International Conference on Multi Agent Systems*, pages 312–319, 1995.

[27] J. S. Rellermeyer, M. Duller, K. Gilmer, D. Maragkos, D. Papageorgiou, and G. Alonso. The software fabric for the internet of things. In *Proceedings of the 1st international conference on The internet of things*, IOT'08, pages 87–104, Berlin, Heidelberg, 2008. Springer-Verlag.

[28] R. Tynan, C. Muldoon, G. O'Hare, and M. O'Grady. Coordinated intelligent power management and the heterogeneous sensing coverage problem. *Comput. J.*, 54:490–502, March 2011.

[29] M. Vukovic, S. Kumara, and O. Greenshpan. Ubiquitous crowdsourcing. In *Proceedings of the 12th ACM international conference adjunct papers on Ubiquitous computing*, Ubicomp '10, pages 523–526, New York, NY, USA, 2010. ACM.

[30] V. Waller and R. B. Johnston. Making ubiquitous computing available. *Commun. ACM*, 52:127–130, October 2009.

[31] M. Weiser. Some computer science issues in ubiquitous computing. *Commun. ACM*, 36:75–84, July 1993.

[32] M. Wooldridge and N. R. Jennings. Intelligent agents: theory and practice. *The Knowledge Engineering Review*, 10(02):115–152, 1995.