Middleware and architectures for space-based and situated computing

Valentin Cristea*, Ciprian Dobre, Alexandru Costan and Florin Pop

Computer Science and Engineering Department,
University 'Politehnica' of Bucharest,
313 Splaiul Independentei, Bucharet 060042, Romania
E-mail: Valentin.Cristea@cs.pub.ro
E-mail: Ciprian.Dobre@cs.pub.ro
E-mail: Alexandru.Costan@cs.pub.ro
E-mail: Florin.Pop@cs.pub.ro
*Corresponding author

Abstract: Situated computing is an actual research area that gathers the knowledge and investigation in mobile, wearable, ubiquitous and augmented computing. The aim is to sustain systems that collect information about contexts and user actions over a period of time and supply it to applications that dynamically adapt to context changes. Situated computing systems have various architectures, but most of them include a middleware layer that supports equipment with limited computing resources and offers functionalities that simplify applications development, deployment, execution, and maintenance. Middleware and architectures are the focus of this paper, which presents the main concepts, research results, and future trends in middleware for situated computing. Service oriented, event driven, agent-based, and peer-to-peer architectures are discussed. Middleware issues for situated computing, embedded devices, autonomic space management, proactive services, context awareness, and smart spaces are tackled as well.

Keywords: situated computing; SitComp; event-driven systems; context aware; middleware; architectures.

Reference to this paper should be made as follows: Cristea, V., Dobre, C., Costan, A. and Pop, F. (2011) 'Middleware and architectures for space-based and situated computing', *Int. J. Space-Based and Situated Computing*, Vol. 1, No. 1, pp.43–58.

Biographical notes: Valentin Cristea is a Professor of the Computer Science and Engineering Department of the University Politehnica of Bucharest (UPB). His main fields of expertise are large-scale distributed systems, context-aware distributed platforms, grid, and cloud computing. He is the Director of the National Center for Information Technology of UPB and leads the Laboratories of Collaborative High Performance Computing and E-Business. He has a long experience in the development, management and/or coordination of national and international research projects on autonomous distributed systems, grid computing, and VANETS. He is the PhD Supervisor and also teaches graduate courses on distributed algorithms and distributed systems.

Ciprian Dobre is an Assistant Professor of the Computer Science and Engineering Department of the University Politehnica of Bucharest (UPB). The main fields of expertise are space-based computing, monitoring and control of distributed systems, modelling and simulation, ad hoc networks. He is actively collaborating with Oracle from which he received his PhD grant of excellence. His PhD thesis was oriented on large-scale distributed system simulation. His research activities were awarded with the Innovations in Networking Award for Experimental Applications in 2008 by the Corporation for Education Network Initiatives (CENIC).

Alexandru Costan is a PhD student and Teaching Assistant at the Computer Science and Engineering Department of the University Politehnica of Bucharest (UPB). His research interests include: autonomic data storage, cloud computing, P2P systems. His PhD thesis is oriented on data storage, representation and interpretation in distributed environments. He received his PhD Excellency grant from Oracle in 2006 and was awarded an IBM PhD Fellowship in 2009.

Florin Pop is an Assistant Professor of the Computer Science and Engineering Department of the University Politehnica of Bucharest (UPB). His research interests are oriented to: contextualised services in distributes systems, predictive systems and intelligent optimisation techniques. He received his PhD in Computer Science in 2008 with 'Magna Cum Laudae' distinction. He received his IBM PhD Assistantship in 2006 (top ranked 1st in CEMA out from 17 awarded students) and a PhD Excellency grant from Oracle in 2006–2008.

1 Introduction

Situated computing (SitComp) is an actual and active research area. It aims to enhance the mobile, wearable, ubiquitous and augmented computing to help people interact with computers "while maintaining their normal flow of work in the real world" (Priyantha and Hirakawa, 2000). The situation is defined not only by contextual factors but also by user activities, personal status, requirements, emotions, and social behaviours. Situation-awareness means "the ability to capture and use the relationships among multiple contexts and actions over a period of time" (Yau et al., 2002). SitComp describes socio-technical systems in which situations of use and context play a central role in the use of computers (Beaudouin-Lafon and Mackay, 2000). The notion of context refers primarily to user's location, time, identity, and environment. But context has a more comprehensive meaning that covers other physical, human, social, and organisational aspects as well. Besides the location and time, the physical context can include the form and type of a computer and its connected networks. These have various characteristics that must be considered by SitComp systems to adapt their actions. In addition, they are indicative of the context environment. For example, desktop computers are typically used in office context while wearable computers can be used for human exploration in extreme environments. Human context refers to human capabilities that could be exploited by SitComp systems to enhance the human-computer interaction. It includes general aspects such as the ability to exchange oral messages (speech) or to give tactile feedback, but also specific personal information (Lieberman and Selker, 2000) about what the users like or dislike, what do they know or not know, what they did in the past (the history), user's emotional state, and focus of attention. The social context is represented by identities of people around the user and is very important for collaborative work and groupware systems in which the activities should be organised according to social interaction patterns and be adapted dynamically to context changes and users' needs. It is part of the environment, which also include objects around the user. Finally, since the organisational context adapts to a continuously evolving environment, it is important to design systems that augment the quality of the organisation they support and help its adaptation to the dynamic context.

The situational information can be used in different ways, which leads to different context aware features (Pascoe, 1998): contextual sensing is the ability to detect contextual information and present it to the user in a convenient form (e.g., user's position can be marked on a map display); contextual adaptation is the ability to automatically execute or modify a service that best corresponds to the current context (e.g., automatically switch to a less expensive communication link when this becomes available); contextual resource discovery is locating and exploiting resources and services that are relevant to the user's context (this may include information resources); contextual augmentation is the ability to associate information with the user's context.

SitComp is directly related to mobile computing and draws from several other fields such as: ubiquitous computing, wearable computing, augmented reality, computer-supported cooperative work, multimodal interaction, and user-centred design.

Different architectures have been adopted for SitComp systems: service oriented, agent-based, peer-to-peer, and others. They are inherently distributed and support a very high variety of mobile devices that can discover each other and form wireless ad hoc networks with variable configurations, topologies, and contexts. System adaptation to environments' changing conditions is ensured by the middleware architectural layer. This hides the heterogeneity of the underlying platform and offers a uniform abstraction to the application software (Hung et al., 2003). The middleware supports equipment with limited computing resources (sensors, mobile equipment) and provides additional functionality to respond to highly demanding applications. It plays an essential role in simplifying the development, deployment, and maintenance of SitComp systems, by resolving common problems such as those mentioned before, to which we can add the fault tolerance, scalability, security, and others. Anyway, some of the solutions used in the general distributed systems must be changed or adapted to the new requirements of the situated distributed computing systems. For example, the replication, which hides the identity or the location of the replica that is actually used, must be approached differently in a location (or context) aware system, in which the location information could determine important decisions at application level.

A middleware for SitComp systems would provide support for (Hung et al., 2003) the acquisition of situational information, the combination and interpretation of this information (deriving higher-level contexts from low-level sensed contexts by using inference and learning mechanisms), and information delivery at an appropriate level of abstraction for situated applications. There are two-main forms of situation information offering. In publish-subscribe model, the information is posted to all the interesting applications that subscribed to it. This model allows defining context triggered events and using notifications to invoke all subscribers when a specific context becomes valid. As an alternative, a query interface can be used to allow applications interrogate the current situation when they need it (Hull et al., 1997). In both cases, a common model of context must be defined, together with a unique associated semantic of contextual information (Ranganathan and Campbell, 2003).

Several other projects have contributed to the development of the field by focusing on separate aspects of SitComp. XMIDDLE (Zachariadis et al., 2002) is a middleware that concentrates on the need of mobile device applications to establish ad hoc peer-to-peer networks. These systems suffer from a low bandwidth and non-continuous network connection, limited memory and processing resources, short battery life, and the non-continuous remote access to information located on different hosts. To respond to these requirements, XMIDDLE has been developed to be very lightweight and fast. It allows applications to share XML encoded data with other hosts, to access the shared data even when disconnected from the network, and to ensure that all hosts have a consistent version of the shared data.

Reconfigurable context-sensitive middleware (RCSM) facilitates the development and runtime operations of context-sensitive pervasive computing software (Yau et al., 2002). RCSM is an object-based framework that models the application as context-sensitive objects, with two parts: a context-sensitive interface and a context independent implementation. It provides object containers (ADCs) that perform the context data acquisition, monitoring, and detection. Also, a context-sensitive object request broker (R-ORB) hides the details of ad hoc networking, and performs device and service discovery on behalf of context-sensitive objects.

MiLAN, a middleware linking applications and networks (Heinzelman et al., 2004) allows sensor network applications to specify their quality needs and adjust the network characteristics to increase application lifetime while still meeting those quality needs. MiLAN receives information from applications (QoS requirements), the overall system (the relative importance of different applications), and the network (available sensors and resources). Based on this information, it adapts continuously the network configuration to meet the applications' needs while supporting the dynamic availability of information sources.

Gaia (Bresler et al., 2004) distributed middleware supports the construction of general-purpose active spaces. It has a two-layer architecture. The lower layer includes the functionality for component management to which event manager, context service, context file system, security, space repository, and presence service components are added. The higher layer is a framework that facilitates the creation and management of typical ubiquitous applications by using six patterns that govern, respectively: multi-device support, user-centrism, run-time adaptation, mobility, context awareness, and environment independence.

More recent projects were developed at MIT Media Lab to explore the future of digitally augmented objects and environments (MIT, 2010). They cope with Smart Cities (the CityCar project), new paradigms of collaboration (the CollaboRhythm technological framework), and new interfaces between humans, digital information, and the physical environment (FaceReader, G-stalt, and SixthSense projects).

Previous surveys refer to partial issues of SitComp architecture and middleware such as context awareness, ubiquitous computing, etc. Chen and Kotz (2000) present a research survey in context aware mobile computing focusing on context processing and context aware applications. Gaddah and Kunz (2003) describe different middleware paradigms for context aware mobile computing systems. In his survey on context aware systems, Baldauf (2007) refers to a layered architecture and focuses on context-aware middleware and frameworks. Demiris (2007) makes a brief survey of the characteristics of context-aware systems with focus on multimedia applications. Kjr (2007) provides a survey of context-aware middleware systems and describes the particular properties of each surveyed system. Cummins et al. (2003) gives an overview of current middleware for ubiquitous computing environment, while Hung et al. (2003) surveys relevant literature and projects in ubiquitous computing and context-aware applications.

Our paper is an attempt to survey the research in situated computer systems by gathering the issues in mobile, wearable, ubiquitous, context-aware, embedded, and augmented computing. We present a general software architecture that includes the main logical components and their relationship for space-based and SitComp systems, and we show how this architecture is instantiated as service oriented, event driven, agent-based, and peer-to-peer system architectures. Also, the main concepts, research results, and future trends in middleware for SitComp are discussed. Middleware challenging issues for SitComp, embedded devices, autonomic space management, proactive services, context awareness, and smart spaces are tackled as well.

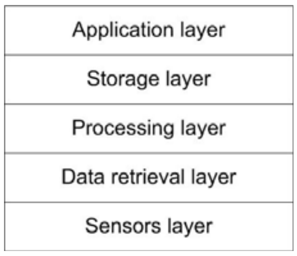
The rest of the paper is organised as follows. Section 2 presents the SitComp system architecture characteristics and discusses the main architectural types. Section 3 discusses in more details the key requirements addressed to the middleware for space-based and SitComp systems. Section 4 presents middleware issues for embedded devices with emphasis on service oriented middleware. Section 5 is devoted to autonomic smart space middleware and presents important issues related to interoperability and integration challenges. Section 6 treats event-based services for context-aware applications that use sensing services as sources of events that carry context data as event parameters. Section 7 puts a focus on context frameworks and middleware. Database access middleware for smart objects is discussed in Section 8. Section 9 is dedicated to conclusions and future trends.

2 Space-based and SitComp architecture

The SitComp systems architecture depends on the locations of sensors, the number of users, the available resources, the method of context-data acquisition, and other factors (Baldauf, 2007). It frames distributed components whose role is to adapt systems' actions to changes occurred in the very dynamic environment in which they operate, and to offer a natural and personalised interaction with users. Many distributed systems adopt a layered architecture (Figure 1).

The sensor layer includes any (hardware or software) data source that can provide situational information. Components at higher layers (which belong to the middleware) retrieve information on the current situation, process it by interpreting, aggregating and composing the raw data, organise the results, store them and offer situational information to applications, which will execute appropriate actions. In some cases, the middleware decides on the actions to be executed by applications or directly calls the corresponding functions in the applications. It is important to note that applications themselves can be sources of situation changes.

Figure 1 SitComp system architecture

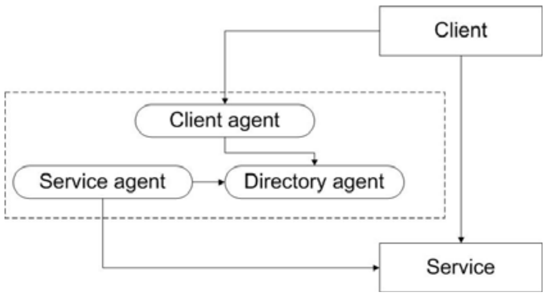


The SitComp system architectures based on this model use different paradigms that characterise their orientation. In object oriented system architectures, the context-sensitive object is the building component, the entire system being assembled of a collection of interconnected objects. In addition, an object request broker (ORB) is used for communication among objects situated in different system's nodes. Specific mechanisms are used to detect application-specific contexts, activate context sensitive objects, invoke object methods, and support objects' adaptation to context. All these facilities can be included in the middleware environment that facilitates the application development and execution. An example is the RCSM, which uses a context-aware interface definition language (CA-IDL) for the specification on context-sensitive object interface, and has components to support the other facilities mentioned above (Yau et al., 2002).

Service-oriented architecture (SOA) systems are built as a collection of distributed services that encapsulate specific functionalities. Services are offered via published interfaces, which are discovered and used by other services to build service invocations. Services can be advertised by service agents and can be discovered by user agents on behalf of clients. In this respect, a directory agent registers (temporarily) the advertisements and responds to user agent requests when it finds a match between the advertised services and the requests. Several works have been dedicated to the subject. A reconfigurable middleware solution for the dynamic service discovery is presented in (Carlos et al., 2007). It includes solutions for service advertising that are based on the use of heterogeneous protocols adapted to various environments. In Le Sommer et al. (2006), the design of a service oriented middleware platform for context-based mobile computing is presented. This uses the environmental context introspection to provide the services they host with an abstraction of their running context. Also, the asynchronous communication is used to support the frequent and unpredictable connection disruptions. The integration of web services in small devices and wireless networks has been facilitated by the definition of an universal plug and play (UpnP) architecture for direct device interconnections in home and office networks (OMA, 2006).

In SitComp, SOA can be coupled with the event paradigm in a natural way (Gaddah and Kunz, 2003) since the situational information can be incorporated in events. Events can advertise a situation change, a user entering a specific context, etc. The communication of events uses the publish/subscribe mechanism, which is an asynchronous communication mechanism with several advantages: the situational information is transmitted multicast to several consumers; producers and consumers are decoupled so that no entity has to know the others; it simplifies the distributed processing of the situational information to obtain situations at different abstraction levels. Publish/subscribe facilitates the use of content-based routing algorithms to deliver the published events to all interested subscribers, such as in Hermes (Gaddah and Kunz, 2003) and supports event-based services (Meier and Cahill, 2002) and adaptive applications (Engelstad et al., 2004) for ad hoc networks. The loose coupling characteristics of event-based architectures respond to the high level dynamism of ad hoc networks and simplify the application development.

Figure 2 Service oriented architecture



Multi-agent system architecture has important characteristics that favour its use in SitComp. Agents are reactive, proactive, autonomous, adaptive, communicative, and mobile. Agent architectures are distributed, robust, and fault tolerant. Several agent-based middleware platforms that will be discussed further in this paper propose partial solutions for SitComp: SIFF (Priyantha and Hirakawa, 2000), MESHMdl (Herrmann, 2004, 2006), LAICA (Cabri et al., 2005), SALSA (Rodriguez et al., 2004) and others.

Peer-to-peer architectures are also used in SitComp systems due to their capability to self-organise in the presence of failures and fluctuations in the environment. They have the advantage that ad hoc administration and maintenance are distributed among the users, thus reducing the cost of collaboration, communication, and processing. Look-up and locating of peers are supported through various mechanisms, which use an overlay network built on top of the physical computer network. In some mobile environments, such as intelligent transportation systems, peers have only a few seconds of connectivity and very limited bandwidth (Zaera, 2008). To avoid the loss of high priority events, such as safety-critical driver warnings, solutions have been proposed such as using a combination of distributed hash tables (DHTs) with Aspect-oriented space containers (Kuhn et al., 2009). Similarly, a notification service for mobile environments in which producers are able to send events even to subscribers that are in permanent move has been proposed (Schwiderski-Grosche and Moody, 2009). The service is able to store all events while the subscribers are offline. Once the interested peer is reachable again, the data associated with the saved events are delivered to its mobile environments.

This presentation of architectural models captures the essential features of SitComp systems. Actual implementations can bring new elements that respond to applications' particular demands. They will be discussed in the following sections, with more insights on middleware components and interaction mechanisms.

3 Middleware requirements

In this section, we discuss in more details the key requirements addressed to the middleware for space-based and SitComp systems. Since middleware is placed between the distributed platform and the applications, requirements come from both sides. The requirements coming from the lower layer refer to the enhancement of the capabilities and performance of the platform and address common issues of traditional distributed systems, such as heterogeneity, fault tolerance and network disconnections, scalability, and mobility. We can add also the protection of users' personal information (e.g., location and preferences) against security attacks, and the support for responding to high performance requirements.

SitComp systems use very diverse hardware components, from sensors, actuators, mobile handheld devices, and wearable computers to high-performance servers. Since mobile devices can have heterogeneous multiple network access (Engelstad et al., 2004), the networking interfaces can also be diverse. Mobile devices can benefit from the capability to connect to the internet in the traditional way as IP hosts, or to connect directly to each other and form ad hoc networks that use different forms of routing. In these cases, middleware should support applications' adaptation to the great variations of resources and services availability, by monitoring and pro-actively discovering resources, controlling the session, and changing the network connections or corresponding nodes. Also, context information may migrate with context-aware components (Henricksen et al., 2005).

Situated system general requirements are very well synthesised in Hull et al. (1997) with emphasis on the SitComp service, which is a middleware component aimed to provide situational information to applications. First, middleware must ensure the transparent access to situational information. For example, the application asking for the identification of a user's companion needs not know the source of this information, be it an active tag, speaker identification, or face recognition. In addition, different aspects (or dimensions) of a situation must be managed separately. For example, a long process needed for the determination of user's accurate location in a route finding application should not block a situated reminder application based on companion information in which the user is involved. Also, situation information must be delivered with low latency since late information might be useless. The system must make a trade-off between the response time and the accuracy of the situational information to ensure in time delivery. Finally, the system should be high throughput (be able to process a large number of situational events) and extensible (allow more capabilities, new situational dimensions, new equipment, to be added).

Adopting the asynchronous interaction between components resolves the problems of high latency and disconnected operations that are likely to produce in mobile systems. An asynchronous client-server interaction does not ask the two-components be running concurrently: a client can launch a request, disconnect from the network and retrieve the result later on (Gaddah and Kunz, 2003). Shared spatial data structures provide another solution to handle the problem of disconnected operations.

Middleware should ensure high QoS despite the various operational conditions such as resources with different structures, interfaces, and controls. For example, the military combat and command/control system described in (Loyall, 2003) incorporates different aerial vehicles and ground and air command and control nodes, which include a variety of processing nodes and network links. Some links use statically allocated and preconfigured time slots; others use dynamic reservation-based and priority-based network management. One important problem is to obtain high performance despite the discrepancies among components. A QoS adaptive middleware designed by authors was specifically oriented towards supporting QoS requirements related to mission, measurement of system conditions, delivered QoS, and resource availability and constraints, control of resource mechanisms, adaptation to system changes, failure treatment, and others.

A specific demand addressed to SitComp middleware, by contrast to the middleware for traditional distributed systems, is to not hide the information about heterogeneity and distribution, which can be offered as contextual data to be used for adapting the applications to particular situations (Kjr, 2007).

Fault tolerance, security and scalability are general requirements also found in general distributed systems.

Two special issues are the easy deployment and configuration, which would meet users' requirements and would facilitate the adaptation to specific environments, and traceability, which would offer enough information to users for understanding, debug, and control the system (Henricksen et al., 2005).

Concerning the requirements coming from the upper layer, the middleware should implement the functions that are common to situated applications: collect, process, and offer information about past and present situations. Possibly, it can extend these functionalities with predicting future probable situations. SitComp middleware should support (Hung et al., 2003) gathering of situational information from different sources and deliver the aggregate information to different agents. Usually, the middleware hides the information sources (consumers of contextual information need not know details about producers) and information consumers (producers of context information need not be concerned about who is using the situational information).

Very often the collection process is complicated by the source of the situated information, which could be noisy, unreliable, redundant, and could describe the local environment at a too low level of detail (Hull et al., 1997). Typically, raw data must be interpreted and processed in order to infer higher-level contexts from low-level sensed data. This processing can be associated with two important mechanisms: rules written in different types of logic (first order logic, temporal logic, fuzzy logic, etc.), and learning mechanisms (like Bayesian networks, neural networks, and reinforcement learning). The problems of interpretation can be even more difficult in the case of middleware that must support dynamic configuration and unpredictable kinds of applications.

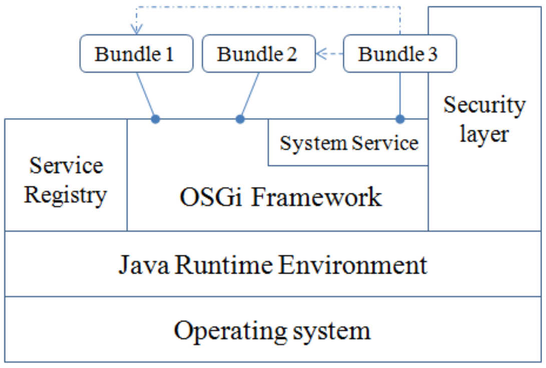
SitComp middleware should permit applications to easily specify the context aware behaviour. This asks for a uniform and platform-independent interface that allows applications to express their need for different context data. These facilities are also useful for enrich-able SitComp in which users add information that augments the context and is then made available to applications (Brown, 1996).

Apart from solving platform interoperability problems, middleware should support the interoperability between heterogeneous application agents (users). This includes the syntactic and semantic interoperability, and the ad hoc communication among different agents (Hung et al., 2003). In addition, a contextual system should accommodate many types of interaction and be user-centred. Social network tools might help understanding the changing relations between users, resources, and contexts. They can be used not only by the application layer to adapt the services according to user requirements but also by the platform layer to dynamically configure the infrastructure for saving the cost and power consumption (Hull et al., 1997).

4 Service-oriented and component-based middleware for embedded devices

Mobile and embedded device platforms have to support users with no abilities to configure the devices they work on, and help them to easier access the resources and services offered by small devices. The middleware has some additional requirements such as green operation, self-configuration with adaptation to the changing network topology, and self-protection to both intended threats and random damages (Bottaro et al., 2007). It generally relies on a subset of the general devices profile for web services – DPWS specification (OASIS, 2009) for enabling plug-and-play for networked devices. In response to the demand for a standardised software management platform, the open service gateway initiative (OSGi) released an open business standard for the management of dynamic modules in Java (Marples and Kriens, 2001). The OSGi platform offers a standardised and integrated environment to facilitate the development, discovery, tracking, communication and cooperation of applications following the SOA and the service oriented device architecture (sensors as utilities in enterprises) (Mauro et al., 2010). Key features of the OSGi service platform are the management of locally or remotely installed software components, secure execution of software components, cooperation and interoperability assurance of software components, etc. Figure 3 illustrates the four-level software architecture of the OSGi service platform.

Figure 3 The OSGi software architecture (see online version for colours)



The OSGi framework includes a run-time platform where services are deployed and maintained via the OSGi lifecycle model. The main components (named bundles) are OSGi applications in the form of deployable web service packages that can be imported and exported dynamically. Bundles can be added, removed, and replaced at runtime while still maintaining the relations and dependencies among them.

Various prototypes of SOA middleware systems for embedded devices have been developed. In MORE (Schmutzler et al., 2009), the runtime is implemented as a DPWS-compliant stack (WS-event is used to implement a publish/subscribe functionality) which is installed on small devices as a regular OSGi bundle. The middleware requires the use of CLDC-compliant platforms to run. The authors made several tests of the prototype with a gas sensor equipped with a GSM/GPRS wireless module. The results prove, as the authors say, that the DPWS standard is too complex to be ported on highly restricted embedded devices, unless further adaptation is performed.

The middleware developed in HYDRA (Eisenhauer et al., 2009) allows creation of ambient intelligence applications based on wireless devices and sensors. HYDRA uses SOA to provide interoperability at a syntactic level, and the semantic model driven architecture (semantic MDA) to facilitate application development and promote semantic interoperability for services and devices. The semantic MDA of HYDRA includes a set of models (ontologies), and describes how they can be used both at design-time and at run-time. The basic idea behind HYDRA Semantic MDA is to differentiate physical devices from application's view of devices. HYDRA also introduces the concept of semantic devices, which represents the model of the real devices, serves as logical units that can be semantically discovered, and provides information about device capabilities and services. The middleware is, however, not yet completely implemented. The project is scheduled to end with a set of demonstration results for intelligent housing and afterwards an implementation will be made available.

The SOA-WSN prototype (Leguay et al., 2008) implements the SOA model for sensors that run tinyOS, together with an OSGi framework. HELLO messages are sent periodically to announce the available services. Routing is performed by implementing a distance-vector multi-hop routing protocol that uses periodic beacons. DPWS stack is installed on powerful gateway nodes that translate (bridge) between WSN-SOA and DPWS and also implement WS-discovery, WS-topics and WS-eventing in order to achieve data dissemination via appropriate event handlers. The tiny nodes and the bridges can communicate according to the publish/subscribe model. Still, the middleware fails to address problems related to fault tolerance, security and scalability. The routing can experience slow convergence and routing loops. The gateway nodes can easily become single-point-of-failure. The tiny nodes are not present with trust guarantees for the data coming from the gateway nodes. The hello messages are disseminating to nodes using an epidemic mechanism, which can lead to congestions.

Despite the important progress in service oriented middleware for embedded devices, some problems are still to be solved in future research. Minimisation of energy consumption through the use of a distributed network-channel coding and network coding division multiplexing strongly tailored to the characteristics of embedded systems, and virtualisation with emphasis on the design and development of a distributed multi-layer and service-oriented middleware aiming to bring SOA's benefits to low capacity nodes are just two of them. They are approached in ongoing research projects such as VITRO, which focuses on supporting the collaborative and multi-purpose virtual sensor networks (Akribopoulos et al., 2010).

5 Autonomic space management

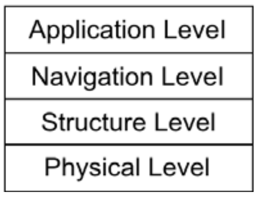
Smart space is a typical pervasive environment in which computation embedded in its surroundings is integrated with human user space, allowing for high dynamicity and context-awareness. Due to this increased complexity, a management framework suited for smart spaces is faced with important interoperable and integration challenges when combining adaptive user services with adaptive resource management in a heterogeneous environment. (Feeney and Frisby, 2006) summarised the challenges specific to smart space design and the issues related to their management: services executing within the smart space must be defined and managed so that their adaptation and composition can be achieved by operators; users should be allowed to create and manage procedures; information needs to be collected, structured, processed and made available as context to users. They are similar to the goals of IBM's initiative on autonomic computing described by (Kephart and Chess, 2003). To achieve these goals, one needs a translation from traditional autonomic models (usually relying on decomposition) to specific spatial management models.

The key requirements for autonomic space management were introduced by Manzalini and Zambonelli (2006) and are mainly focused on situation awareness. Situatedness means that the activities of the autonomic components are strongly related to their location in either a physical or a virtual computational environment. Since services autonomously adapt to the context from which they are requested and in which they execute, the supporting technologies must be able to capture situational data and effectively exploit it.

For situation awareness, the nodes of a network within a smart space must be able to autonomously connect with each other, set up some sorts of common coordinate systems, and position themselves in the space. In addition, the nodes of the network should be able to self-reorganise their distribution in the virtual space and to make room for new nodes joining the network (i.e., allocate a portion of the virtual space to these nodes), fill the space left by nodes that leave the network, and change the spatial distribution of nodes to react to node mobility. With this approach, a system can dynamically tune the structure of the space and enforce some sort of autonomic management of the network, transparently to the higher application levels.

When implementing a specific autonomic management system, one needs to provide computational models to wrap each of the components and to model the interfaces and the interaction among the components and between the system and its environment. Zambonelli and Mamei (2005) introduced such a model based on an autonomic spatial computing stack. As shown in the Figure 4, the autonomic mechanisms are structured according to a 'space-oriented' stack of layers.

Figure 4 The autonomic spatial management stack



For each layer in the model, the authors identify several mechanisms, exploited in different scenarios that target common goals and provide similar 'spatial services'.

The physical layer models the initial interaction between the elements of the framed system. Since at this level the components are characterised by a high churn, only very basic forms of autonomy are present. However, these are essential for the complex expressions of autonomic behaviour needed at higher levels. Usually, the most used technique in this case is the broadcast.

In Zimmermann et al. (2006), an illustration of this approach is presented for the autonomic management of a group of collaborating base stations to provide effective wireless network access in dynamic environments. New base stations that join an existing wireless network integrate themselves seamlessly, while the rest of the system adapts to their presence dynamically. However, a more complete system evaluation is still needed in order to evaluate the intrusiveness of the monitoring component added to the wireless stations to support their autonomic behaviour. Also, the system may benefit from additional functionalities like improved channel allocation, load balancing, rogue detection or location tracking.

The spatial structure overlay is created by the elements of the underlying physical network at the structure level. This process usually relies on the existing techniques for spatial localisation and look-up. The ability to build and manage a durable spatial structure independent of the physical network dynamics and self-adjusting in real-time to the sensed conditions of the network represents an important expression of autonomic behaviour. In addition to the mechanisms of geographical localisation, pervasive computing systems also rely on logical spatial structures (e.g., overlays) that define abstract spatial relationships within the framed physical computing space.

An example is presented in Borcea et al. (2004), which introduce spatial programming (SP), a space-aware programming model based on smart messages, for outdoor distributed embedded systems. Central to SP are the concepts of space and spatial reference, which provide applications with a virtual resource naming in networks of embedded systems. SP defines two-abstractions to support outdoor distributed programming: space regions and spatial references. A space region is a virtual representation of a given physical space. A spatial reference is defined as a space:tag pair which is mapped to a system embedded in the physical space. The space is a region that represents the geographical scope of this system, while the tag is the name of a property or service provided by the same system. Using reference consistency and access timeout, this approach offers fine-grained, network-transparent access to systems embedded in the physical space. The main benefits of SP are the flexibility and simplicity to program user-defined distributed applications in highly volatile outdoor computing environments. Anyway, SP is an early attempt to design and implement a space-aware programming model and its potential was not fully explored.

The navigation level is focused on the techniques used to guide node activities in the spatial structure and to track the local properties of the space that make exploitable the underlying spatial structure. Usually, this is achieved through geographical routing that targets specific physical coordinates, multi-hop routing using spanning trees or some metric-based routing techniques. However, when the spatial structure lacks well-defined metrics, the only navigation techniques remain flooding and gossiping (random navigation).

GeoDiscovery is an application aiming at node orientation in the smart space, by integrating content with GPS and with a location-oriented metadata language, called GeoXML (GeoDiscovery Website, 2010). Based on XML, this language handles attributes and other information associated with pure coordinate values for location. GeoXML is agnostic about presentation methods; hence, it allows publishers to have a single data model but to deliver content in a variety of media. Several formats are supported including photographs, video, drawings, models and maps. For instance, the Google Maps API is able to handle GeoXML markers. However, in order to gain a large adoption in real life scenarios the system needs an adapted version for mobile devices – the main sources for its input data. Also, in the current context of social networks expansion, the system lacks important features like sharing among users of the location-based information captured.

The application level uses the underlying navigation techniques to coordinate tasks within the application components. With this approach, a self-adaptive feedback loop can be used to develop applications, as in the model suggested by Kephart and Chess (2003). Components can navigate in the space and use the sensed structure and properties of the space to differentiate their tasks, while at the same time they can adapt the existing structure to the tracked conditions and detected events within their activities.

Cascadas (Manzalini and Zambonelli, 2006) is an emerging solution that approaches the problem by conceiving a form of application level overlay network composed of service components, supported at the execution level by mid-level components. These can enforce in an autonomic manner important features such as security, QoS, self-supervision, self-survivability and knowledge-based self-adaptation. Cascadas considers developing and deploying each of these components, both at the application level and at the mid-level, in terms of self-similar ACE components or ACE aggregates. These components are then able to dynamically self-organise as needed with each other and with entities that are already deployed, and to interact so as to provide the desired functionality in a situation-aware manner with minimal configuration effort. While a proof of concept scenario that refers to a pervasive behavioural advertisement application was developed, a potential future industrial application is needed to demonstrate the effectiveness of the platform in a fully distributed fashion.

Policies represent another mechanism to provide the runtime configuration of component behaviour needed in adaptive smart space management systems, at the application level. The Gaia project (Roman and Campbell, 2000) uses policies to implement a model for security and access control using roles and different authentication methods with variable confidence levels. In Gaia, all resources in the system are associated with policies. Users present credentials (closely linked to roles), which are compared with policies in order to validate particular resources within the smart space. One of the main achievements of Gaia resides in its ability to simplify the development of portable applications that can be dynamically partitioned and mapped to a variety of devices, can be customised based on the space context, are bound to users, and can move across different spaces. Gaia encapsulates the heterogeneity of active spaces, and presents them as a programmable environment, instead of a collection of individual and disconnected heterogeneous devices. While this work demonstrates the flexibility and power of policies and role-based access control in smart spaces, there are still several areas that have to be addressed in the context of access control-based autonomic space management: policy refinement, role delegation or collaborative management are just a few examples. Moreover, Gaia lacks services that support the user virtual space abstraction. Finally, we note that the system could benefit from the support for federating several Gaia services in order to aggregate different active spaces.

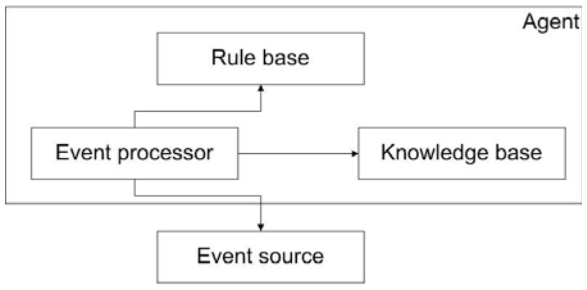
The new communication/networking paradigms employed for the autonomic space management can be situated in multiple and dynamic contexts (ranging from sensor networks to virtual networks of humans), and are expected to be autonomously controlled, self-organising, radically distributed, technology independent and scale-free. While this facilitates advances both in the architecture and functionality of the management systems, some challenges still remain to be addressed: overall stability and resilience of the smart space's underlying network as it evolves, trustworthiness and the interaction of new communication paradigms with human, social or commercial aspects, in relation to ambient intelligence and environments' sensors.

6 Event-based and proactive services and middleware

Event-based services are at the aim of context-based applications that use sensing services as sources of events that carry context data as event parameters. Event processing is done according to specific rules and consists of event aggregation, classification, and abstraction, which are used for generating higher level complex events. Event combinations are obtained by disjunction, conjunction or sequencing of two-events, the occurrence of an aperiodic event, periodic occurrence of an event, non-occurrence, and temporal (Chakravarthy and Adaikkalavan, 2009). Several models have been proposed for complex event detection. One of the most used models is event, condition, action (ECA) paradigm (Chakravarthy and Adaikkalavan, 2007), in which event elements are described as a rule with three parts: the event detected, the relevant context (condition) in which the event occurred, and the action triggered by the event occurrence. For implementation, the model can use WS-eventing and WS-ECA technologies for embedded services (Jung et al., 2007). The condition part is implemented as an XPath expression, while the action part is a conjunction or disjunction of several primitive or composite actions, which can be service invocations, creation and publishing of events or triggering other rules.

Agents add pro-activeness to event-based services. Agent general architecture (Figure 5) includes an event processor (EP), a rule base (RB), and a knowledge base (KB). The occurrence of an event is detected by the EP and is stored in the KB, which then activates a specific rule or set of rules from the RB. Intelligent pro-active agents have capabilities for interpreting the context, draw inferences, take decisions, plan actions, and execute them to determine changes in view of fulfilling the goal for which they have been designed.

Figure 5 General agent architecture



The concept of agents fits well with the requirements of SitComp: agents of different categories can be spread in the environment and can collaborate with each other to offer services that are adapted to the current environment situation. In Cabri et al. (2005), the authors present their experience in the LAICA project, in which an agent infrastructure for ambient intelligence in an urban scenario has been developed. Agents can capture situational information coming from a variety of computer-based devices and sensors, and, due to their proactively and autonomous characteristics, they can act upon the environment, through actuators, in a distributed way. To this, we can add the social abilities of agents: since a device can be accessed by several agents and an agent can access several devices, agents can exchange information and get a more precise image of the environment; also, they can coordinate their actions in order to tolerate the dynamic behaviour of the environment and offer more dependable services. Two other important characteristics can be mentioned. One is agent's mobility that can be exploited for capturing situational data during agent migration and for dynamic agent updates at run time. The second is the ability to extend the system functionality by the addition of new agent categories. The LAICA urban scenario uses several components like traffic lights, video cameras, motion sensors and dark vision sensors. The data sensed by the devices are pre-processed and delivered to the middleware level. The middleware includes the basic services to support agent execution, more specific discovery and event services.

There are two main agent classes in LAICA: sensor and effector agents (Cabri et al., 2005). The sensor agents are attached to the environmental devices which they control locally. Effector agents can be associated to users and offer them adapted services by interacting with other agents on users' behalf. However, LAICA's ambient infrastructure is not very rich and the experiments are limited to a reduced number of agent classes. In addition, this middleware lacks an API for creating agent-based situation aware systems.

Some applications need to self-adapt to users' tasks, in other words, they must be activity-aware. The support frameworks of these applications need components with intelligent capabilities for learning from users' behaviour, modelling computational activities, and gathering context for inferring and representing activities. Agent-based frameworks fit well the above mentioned requirements due to agents' characteristics such as reactivity, pro-activity, learning capabilities, etc. An extension of SALSA – simple agent library for smart ambient (Rodriguez et al., 2004) incorporates customisable activity mechanisms to achieve operations related to activity-awareness. In SALSA, the computational representation of a human activity takes the form of e-activity, which is characterised by attributes such as persons performing the activity, her location, artefacts used for activities performance, frequency and schedule of execution. SALSA middleware includes the ingredients to manage e-activities' life cycle, which includes three main phases: initialised, activated, and accomplished. SALSA middleware facilitates the creation of a representational model of users' activities, and supports specialised queries for retrieving different modules of the model or activity histories. However, SALSA has a limited usage as a research test bed for agent-based ubiquitous computing systems, and for teaching purposes.

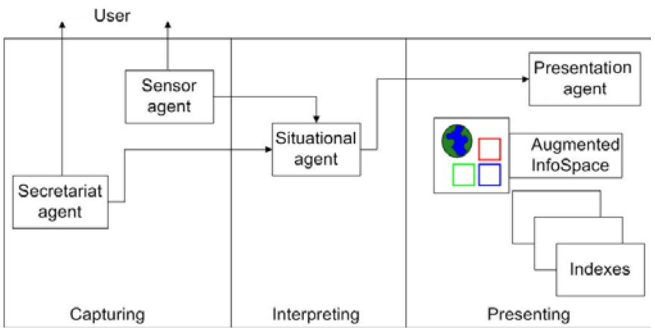
Other works have been devoted to activity-aware computing. The activity-based computing framework provides components for activity and state management that uses the generic activity markup language, AML activity ontology (Bardram and Christensen, 2007). In Mahmud et al. (2009), a high-level guideline for developing activity-aware applications is proposed, with emphasis on systems for activity recognition.

7 Context frameworks, databases and middleware

Several significant issues are associated with space and SitComp and must be included in any discussion about this concept. They are the location information (or spatial identity) that characterises a situation, the time or time interval (temporal identity) of an activity or event, the activity identity, descriptive information about the activity (filing information), links to information entities related to the situation, filtering information based on actual context and user interest, and visualisation. They can be found in the augmented album (Priyantha and Hirakawa, 2000), a situation-based application developed to demonstrate how the user-situations transform the management and retrieval of digital images (and video clips as well) into a more natural and user friendly process. It was developed on top of the situated information filing and filtering – SIFF middleware in which the location, time, and user's social events are captured as contextual information when a picture is taken. For retrieving digital images, the user can interact with the application by three components: map component, time frame component, and events component.

SIFF supports this and other similar applications presented in Priyantha and Hirakawa (2000). It is based on the situation metaphor according to which the computer mediated and non-mediated activities are included in a situation space. This space integrates the physical context, user activities and information entities, and describes all identified past user-situations that could be used in providing appropriate system behaviours and functionalities. SIFF has a multi-agent-based system architecture organised as three functional divisions (Figure 6). The context capturing division uses two-agents (secretariat agent and sensor agent) that capture the user's context and perform activities for achieving her goals in the workspace. The interpreting division includes the situational agent, a RB, and a KB. The presenting division includes the presentation agent, three index files, and the augmented information space.

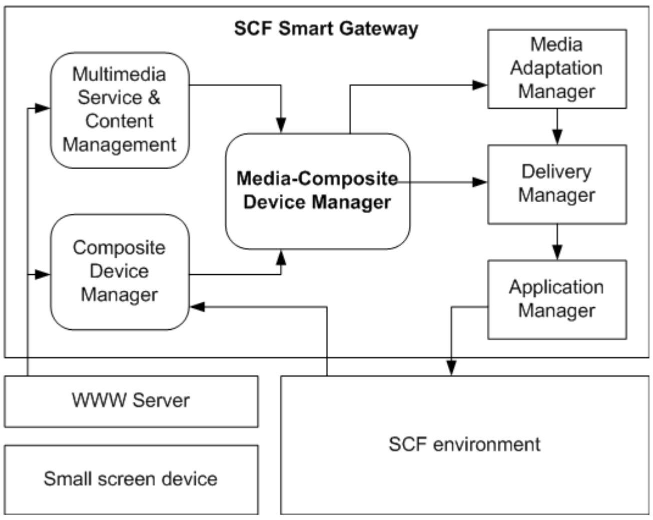
Figure 6 SIFF architecture (see online version for colours)



The augmented information space contains the data to which situational information is added as metadata. Each information entity is a logical collection of multimedia data sources with respect to an initial situation. An entity has two parts: an interface and a data segment. The interface contains attributes that represent the situational information. Attributes may take multiple values based on which multiple hierarchies can be built. The data segment contains one or more objects. The key components in this division are the index files that describe the hierarchical information organisation in the situation space and are used to support the functionality of the situation metaphor, especially situation-based data search and situation-dependent filtering of the information for presentation. This organisation of the information space plays an important role to achieve functional features in the situation metaphor. It is adequate to the functional needs of the applications and better responds to the requirements of an augmented interface for the information space. It represents an improvement (from conceptual and physical points of view) of the traditional data file organisation, which is closer to the physical storage structure than to application's requirements.

In Ranganathan and Campbell (2003), a context-aware middleware for a ubiquitous computing environment is presented. It is part of Gaia (Roman and Campbell, 2002), an infrastructure for smart spaces. This middleware is based on a predicate model of context. Each predicate has a name, which represents its type, and a description whose format depends on the type. For example, a location context predicate can indicate that Chris (a person) enters (a verb) the room 3231 (a location) in the form location (Chris, entering, room 3231). The context predicates and their arguments are specified in an ontology, which defines the semantics of different contexts. This can be used for checking the predicates and for supporting the interoperability of different computing environments with different ontologies. The predicate model supports the use of different reasoning mechanisms. The current implementation is based on many-sorted first order logic, propositional linear-time temporal logic or probabilistic propositional logic. It also allows agents to learn by using Bayesian methods or reinforcement learning. Context synthesis provides higher-level contexts starting from simpler sensed contexts. Rule-based synthesis use pre-defined rules to infer different contexts. For example, the rule #People (Room 2401, ‘‘>=’’, 3) AND Application (PowerPoint, Running) ‘‘=>’’ RoomActivity (2401, Presentation) can be used to deduce the current context in a room based on the number of people in the room and the applications currently running there. Rules can also be used to express context sensitive behaviour. Despite the important innovations proposed in Gaia, it requires important resources for computing intensive activities, which does not fit with limited resources of mobile terminals. Gaia does not consider the scalability, privacy, and user monitoring and control. Also, limited forms of heterogeneity, mobility, and component configuration are supported.

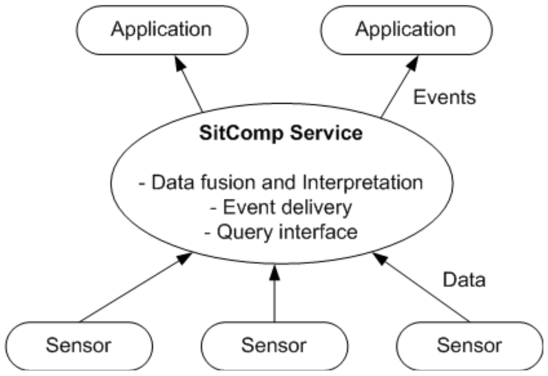
Figure 7 SCF System architecture



SitComp can be used to extend the capabilities of scarce resource devices by using more powerful equipment found in the user’s local context. In the SitComp framework, SCF (Pham et al., 2000) this extension is based on ubiquitous computing infrastructure and makes possible nomadic users to access rich multimedia contents (that require powerful equipment) from a small-screen device. The idea behind SCF is to use an innovative distributed architecture, called small screen-composite device (SS/CD) to ensure gathering the available resources existing in user’s current location proximity into a more powerful device and make it available to mobile PDA users. SCF has been designed to be user-centric, pervasive, full multimedia capable, and to discover and exploit composite devices. The approach eliminates the need to adapt the multimedia content to small devices and avoids users wearing heavy equipments.

The smart gateway is the key component of the SCF since it adds adaptation and flexibility to the middleware. So, the media-composite device manager identifies appropriate devices for the performance of selected media, based on the information stored in a database of available composite elements that is maintained by the composite device manager. If the characteristics of the available composite devices are not proper to handle requested contents and services, the media is adapted to the capabilities of the actual composite devices by the media adaptation manager. In addition, the delivery manager determines the appropriate delivery order of services. Finally, the application manager invokes processes running remotely on available output devices and interacts with the application. Some of the ideas proposed in this work have been enriched by other authors, For example, Elting (2005) improves the idea to connect a PDA to different computers that export an HTML interface or stream media to the PDA, to make possible the combination of the display of the PDA with other displays, and show different parts of the HTML page simultaneously on different devices.

Figure 8 SitComp service as middleware



Different issues are encountered in wearable computers that support situated applications. The work in Hull et al. (1997) is based on the experience gained in the design and development of a pilot ultra-portable computing (UPC) platform with three main components: the first does sensing places and people, the second interprets sensor data, and the third is a situated application. The first component is used to detect the location and companions, which are marked with active tags. The tagging system also includes a small low-powered detector, which fits into a wearable computer and communicates with tags by near-field radio. Sensor data interpretation is done by the SitComp service, a middleware component. It receives raw data from sensors and interprets them to provide a “stable, reliable and abstract view of the status of the current situation, and changes in that status” (Hull et al., 1997). SitComp was designed to respond to difficult problems that sensor data interpretation could face, such as noisy or missing data, redundant data coming from several sensors, and the use of heuristics. In addition, SitComp handles and delivers events, and offers a query interface to the application.

Despite the adoption of simple solutions for the prototype implementation, the experience with the UPC platform allowed authors to highlight the value of SitComp ant its applicability to a large spectrum of application areas such as augmented reality, localised information, context-based retrieval, situated reminders, appropriate behaviour of devices, and monitoring.

Users can not only consume contextualised information but also produce and enrich the environment with contextualised information. An example is given in Brown et al. (1997), which refers to a new form of document, called a stick-e document. This is composed of smaller components, named stick-e notes. Each stick-e note is the digital equivalent of a post-it note, which can be created and posted by using a hand held device like a PDA. The stick-e has attached a context, so that it is triggered (the user can decide what stick-e documents can be triggered) whenever the user situates in the same context. The platform supporting stick-e documents is distributed and contains components that allow note and document preparation, document management (e.g., making them triggerable), trigger them, and present the triggered notes to the user. A useful feature for the note presentation is to prefix the content of a note, the motivation for triggering it, which makes users aware that they entered a new context.

Aggregation and interpretation of sensor data is approached in many other projects. Context fusion networks (Chen et al., 2004) provides data fusion services based on an operator graph model that allows application developers to specify the processing of context information. It has been implemented as a peer-to-peer network named solar, which is scalable, reliable and supports mobility. Anyway, solar does not address heterogeneity, privacy, monitoring, and control of the system by users. The PACE middleware for distributed context-aware systems (Henricksen et al., 2005) provides aggregation and storage of context information, performs query evaluation, and assists context-aware applications with making context-based decisions. The middleware includes also a messaging framework and a set of tools, which produce custom components for developing and deploying context-aware systems. Several problems have not been adequately addressed by PACE, such as scalable deployment, configuration and management of sensors, and caching of context information to support mobility.

8 Database access middleware for smart objects and spaces

A smart object is an item equipped with sensors and/or actuators to interact with the physical world, a tiny microprocessor to process the data captured from sensors and execute proper actions, a communication device to interact with other objects, and a power source. Smart spaces are ordinary indoor or outdoor spaces, such as parking areas or classrooms, equipped with sensors, actuators, communicators, cameras, microphones, speakers, displays, RFID tags, etc. So, a smart space consists of multiple smart objects designed to make the ordinary space become ‘smart’. Such spaces are capable of perceiving and reacting to their inhabitants’ behaviour. They can provide proactive notifications or facilitate the information exchange between multiple artefacts that compose an intelligent living room (Kawsar et al., 2008). The smart objects have several characteristics (Kawsar et al., 2009): affordances that affects how people use it (e.g., a smart table can play the additional roles of ambient display and proximity detector), keeping the original physical appearance while providing perceptual feedback of their states, support both pushing and pulling the environment phenomenon, and maintaining a persistent memory. Sophisticated objects such as AwareMirror maintain locally state history (Fujinami et al., 2005). The AwareMirror presents information relevant to a person in front of it by super-imposing his/her image. A toothbrush has been chosen as an identification tool while proximity sensors have been utilised to detect a person’s position (in front of the mirror).

A well-known middleware for smart objects is Prottoy (Kawsar et al., 2009). The framework is composed of an artefact wrapper and a virtual artefact. The former encapsulates smart objects and the latter is responsible for the manipulation of such objects. For each smart object, application developer’s instantiate a virtual artefact in the application space to interact with the corresponding smart object. Similar to the OSGi approach, artefacts and features are dynamically inserted on top of a generic core. This layer provides a communication module with support for the transport layer, a discovery module that allows service advertisement, and a notification module that enables other modules and artefacts to indicate their status. The framework was successfully tested in several experiments such as the design of a wearable teddy that acts as user interface for information services, monitors user’s physical activity state, and notifies personalised information in a contextual manner. The experiments also show Prottoy’s limitations. There is no location model in Prottoy, so e.g., the artefacts location has to be statically updated. Another issue is the query support. Currently, Prottoy provides only AND operation, that is artefacts properties and capabilities can only be concatenated for artefact searching. Prottoy cannot handle other combinations (like OR or XOR, etc.). Also, the proxy and security components of Prottoy provide just basic functionalities with little place for extensions.

The Parlay/OSA framework provides a standard way for applications to access information such as user location or account balance from the secure servers using telecommunication networks (Cummins et al., 2003). The framework provides controlled access to service capability features (SCFs), defined as collections of interfaces that provide specific functions, which in combination with distributed technology supports flexibility in application location and business scenarios. It consists of a core part having a trust and security management component (for authentication of domains), a SCF for registration of new SCF to the framework, a SCF factory (for creation of new SCF instance), and a SCF discovery (discovery of SCFs provided by the operator). In addition, there are also framework interfaces for integrity management (for load balancing, fault management, heart beat), event notification (for notifications for specific events, such as the registration of new service capability servers, or SCS), and contract management (for management of contracts between different domains, such as between application provider and network operator). The service capability servers (SCS) are logical entities implementing a Parlay API that are capable to interact with core network elements. Examples of SCS are as home location registry (HLR), mobile switching centre (MSC), service switching point, etc.

The set of APIs developed within Parlay/OSA has grown gradually. Consequently, the resulted set of specifications is rather complex, which decreases its accessibility. Many authors argue that currently a high-level description is needed and the Parlay/OSA data-types are especially rather complex. A model is missing, which describes the relations of the different data-types related to registration and discovery. However, solving these problems is under development.

Smart objects and spaces are intended to provide rich ambient intelligence environments. Frameworks supporting such paradigms are built around ubiquitous computing and human-centric computer interaction designs and include technologies that are embedded, context aware, personalised, adaptive and anticipatory. Such frameworks introduce pro-activity to the world of ubiquitous computing. However, a current problem is that a proactive middleware needs to interfere with the everyday life of humans, while respecting the invisibility imposed by ubiquitous computing. A research trend is, therefore, towards finding adequate models that balance between invisibility and pro-activity.

9 Conclusions and future trends

We discussed SitComp, an actual research area that aims to support people interaction with computers in a more natural way than the traditional office context computing. SitComp is directly related to mobile, ubiquitous, and wearable computing, augmented reality, computer-supported cooperative work, multimodal interaction, and user-centred design. Different architectures and middleware platforms have been developed for SitComp systems, which are inherently distributed and support a large variety of computing resources (sensors, mobile equipment, etc.) to provide functionalities that respond to high demand applications.

Several projects have been presented, which approached the development of SitComp middleware and applications. They incorporate new innovative models, techniques, mechanisms, and intelligent solutions that recognise the context, user activities and other situational information to support advanced systems' characteristics such as the adaptability, flexibility, learning ability, etc.

Despite the significant achievements, important aspects are still to be studied in future works. A very important issue is related to SitComp systems' scalability. While past and actual projects focus on new methods for sensing and using situational information, projects to build large systems and experiment these methods at larger scale would help to check their applicability and discover new issues that must be considered and eventually solve them (Mills and Scholtz, 2001). Research will be devoted to various aspects of large-scale ubiquitous computing, small portable or wearable devices and wireless networks that run in an information rich environment. Ensuring their autonomy and adaptability are two important issues that could help not only to support situation awareness but also in facilitating their management and administration. The next distributed networks of cooperating objects (including sensors and actuators) will respond to requirements regarding scale, heterogeneity, integration with services, and will enable person/object and object/object seamless communication. The objective is to provide the architecture and technology for developing context-aware, reliable, energy-efficient and secure distributed networks of cooperating smart devices and objects, opening a new range of internet enabled applications.

Smart systems will be able to sense, describe, and qualify a given situation, and will mutually identify and address each other. They will predict, help to decide, and to interact with their environment by using highly sophisticated interfaces between systems and users. Larger scale and more diverse tests will be used for validating the research results in conditions close to reality. Smart components and smart systems will integrate new functionalities. Accent will be put on the design, modelling, and operation of a number of a large number of independent, heterogeneous, and interacting embedded systems, and of autonomous interconnected systems (system of systems).

Other aspects not sufficiently studied are the security and privacy (Baldauf, 2007). They are related to the protection of sensitive situational data and must be approached by special components of the middleware. Solutions for different context encodings should be found together with new ways to identify and access context sources. The context information might be private and should not be visible to all applications (Hung et al., 2003). Similarly, the actions developed for applications should not violate the security policies in force. In addition, application services should be dynamically adaptable to security policies, but also to environmental situation and available resources. Privacy is important for the acceptance of technological developments in SitComp, since it could be in contradiction with gathering of information from observing people in their everyday activities (Demiris, 2007). This is just one of the usability aspects of context-aware applications (Hung et al., 2003). Other usability issues that should be studied in the future are related to users' implication in specifying rules for the context-sensitive behaviour of applications, and the acceptance of behavioural changes of applications in function of context modifications, which is different from the predictable nature of conventional applications. So, the social implications of

SitComp need a more thorough analysis (Mills and Scholtz, 2001), specifically, the implications SitComp has on the nature of our work and play, on interaction within and among organisations, and on reducing the digital divide.

The interoperability of context formats, communication mechanisms, and service descriptions is also important for the enhancement of SitComp systems (Baldauf, 2007). Standard protocols and associated technologies (WSDL, SOAP, etc.) are already known solutions to enhance the interoperability. More research is needed in ensuring the semantic interoperability based on ontologies and deriving new contextual patterns to aggregate context-aware services. Provision of adaptive services could also be based on composing atomic context-aware services into higher level services based on context information.

For augmented computing, besides the advances in enriching the user environment, new contributions are needed for augmented information visualisation (Demiris, 2007), more specific the multiple device deployment. For example, in a context aware application, the display of multimedia on a TV screen in a room is continued on a portable device when the user leaves the room.

Acknowledgements

The research presented in this paper is supported by national project 'DEPSYS – models and techniques for ensuring reliability, safety, availability and security of large-scale distributess, Project 'CNCSIS-IDEI' ID: 1710. The work has been co-funded by the Sectoral Operational Programme Human Resources Development 2007–2013 of the Romanian Ministry of Labour, Family and Social Protection through the Financial Agreement POSDRU/89/1.5/S/62557.

References

Akribopoulos, O., Chatzigiannakis, I., Koninis, C. and Theodoridis, V. (2010) 'A web services-oriented architecture for integrating small programmable objects in the web of things', in *Proc. of 3rd Int. Conf. on Developments in eSystems Engineering (DeSE 2010)*, pp.70–75, London, UK.

Baldauf, M. (2007) 'A survey on context-aware systems', *Int. J. Ad Hoc and Ubiquitous Computing*, Vol. 2, No. 4, pp.263–277.

Bardram, J. and Christensen, H.B. (2007) 'Pervasive computing support for hospitals: an overview of the activity-based computing project', *IEEE Pervasive Computer*, Vol. 6, No. 1, pp.44–51.

Beaudouin-Lafon, M. and Mackay, W.E. (2000) 'Research directions in situated computing', in *Proc. of the Workshop on Research Directions in Situated Computing, ACM Conf. on Human Factors in Computing Systems (CHI 2000)*, pp.369–372, The Hague, The Netherlands.

Borcea, C., Intanagonwiwat, C., Kang, P., Kremer, U. and Iftode, L. (2004) 'Spatial programming using smart messages: design and implementation', in *Proc. of the 24th Int. Conf. on Distributed Computing Systems*, pp.690–699, Tokyo, Japan.

Bottaro, A., Grodolle, A. and Lalanda, P. (2007) 'Pervasive service composition in the home network', in *Proc. of the 21st Int. IEEE Conf. on Advanced Information Networking and Applications (AINA-07)*, pp.596–603, Niagara Falls, Canada.

Bresler, J., Al-Muhtadi, J. and Campbell, R. (2004) 'Gaia mobility: extending active space boundaries to everyday devices', in *Proc. of the 24th Int. Conf. on Distributed Computing Systems Workshops*, pp.430–433, Tokyo, Japan.

Brown, P.J. (1996) 'The stick-e document: a framework for creating context-aware applications', *Electronic Publishing*, June 1995, Vol. 8, Nos. 2–3, pp.259–272.

Brown, P.J., Bovey, J.D. and Chen, X. (1997) 'Context-aware applications: from the laboratory to the marketplace', *IEEE Personal Communications*, Vol. 4, No. 5, pp.58–64.

Cabri, G., Ferrari, L., Leonardi, L. and Zambonelli, F. (2005) 'The LAICA project: supporting ambient intelligence via agents and ad-hoc middleware', in *Proc. of the 14th IEEE Int. Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise*, pp.39–44, Washington, DC.

Carlos, A.F-C., Blair, G.S. and Grace, P. (2007) 'An adaptive middleware to overcome service discovery heterogeneity in mobile ad hoc environments', *Distributed Systems Online*, June, Vol. 8, No. 7, pp.1–11.

Chakravarthy, S. and Adaikkalavan, R. (2007) 'Ubiquitous nature of event-driven approaches: a retrospective view', in *Proc. of the Dagstuhl Seminar 07191*, Position Paper, available at http://drops.dagstuhl.de/volltexte/2007/1150/pdf/07191.Chakravarthy Sharma.Paper.1150.pdf (accessed on 10 January 2010).

Chakravarthy, S. and Adaikkalavan, R. (2009) 'Provenance and impact of complex event processing (CEP): a retrospective view', in Buchmann, A., Darmstadt, T.U. and Koldehofe, B. (Eds.): *Special Issue of IT – Complex Event Processing*, Oldenbourg Publications, September, Vol. 51, No. 5, pp.243–249.

Chen, G. and Kotz, D. (2000) 'a survey of context-aware mobile computing research', Dartmouth Computer Sci. Tech. Report TR2000-381, Dartmouth College.

Chen, G., Li, M. and Kotz, D. (2004) 'Design and implementation of a large-scale context fusion network', in *Proc. of the 1st Annual Int. Conf. on Mobile and Ubiquitous Systems (MobiQuitous)*, pp.246–255, IEEE Computer Society, Cambridge, MA, USA.

Cummins, S., Davy, A., Finnegan, J. and Carroll, R. (2003) 'State of the art: middleware in smart space management', *M-Zones Deliverable 1, The M-Zones Project*; also as O'Foghl, M. (2003) 'Infrastructure requirements for smart spaces and managed zones', *First Int. Workshop on Management of Ubiquitous Communications and Services, MUCS 2003*, pp.28–34, Waterford, Ireland.

Demiris, T. (2007) 'Context revisited: a brief survey of research in context aware multimedia systems', in *Proc. of the 3rd Int. Conf. on Mobile multimedia communications (MobiMedia '07), ICST (Institute for Computer Sci, Social-Informatics and Telecommunications Engineering), ICST*, Article 66, p.5, Brussels, Belgium.

Eisenhauer, M., Rosengren, P. and Antolin, P. (2009) 'A development platform for integrating wireless devices and sensors into ambient intelligence systems', in *Proc. of the SECON Workshops '09, In 6th Annual IEEE Communications Society Conf. on Sensor, Mesh and Ad Hoc Communications and Networks Workshops*, pp.1–3, Rome, Italy.

Elting, C. (2005) 'Orchestrating output devices: planning multimedia presentations for home entertainment with ambient intelligence', in *Proc. of the 2005 Joint Conf. on Smart Objects and Ambient Intelligence: Innovative Context-Aware Services: Usages and Technologies*, pp.143–158, Grenoble, France.

Engelstad, P.E., Egeland, G., Bygdas, S.S., Geers, R. and Urnes, T. (2004) 'Middleware supporting adaptive services in on-demand ad hoc networks', available at http://folk.uio.no/paalee/publications/MW-over-AODV-paper-for-ICIN04.pdf (accessed on 20 January 2010).

Feeney, M. and Frisby, R. (2006) 'Autonomic management of smart spaces', in *Proc. of the 3rd Int. Workshop on Managing Ubiquitous Communications And Services (MUCS 2006)*, pp.108–119, Cork, Ireland.

Fujinami, K., Kawsar, F. and Nakajima, T. (2005) 'AwareMirror: a personalized display using a mirror', in Gellersen, H-W., Want, R. and Schmidt, A. (Eds.): *PERVASIVE 2005, LNCS*, Vol. 3468, pp.315–332.

Gaddah, A. and Kunz, T. (2003) 'A survey of middleware paradigms for mobile computing', Technical Report SCE-03-16, Carleton University Systems and Computing Engineering, July.

GeoDiscovery Website (2010) Available at http://www.geodiscovery.com (accessed on September).

Heinzelman, W.B, Murphy, A.L., Carvalho, H.S. and Perillo, M.A. (2004) 'Middleware to support sensor network applications', *IEEE Network*, Vol. 18, No. 1, pp.6–14.

Henricksen, K., Indulska, J., McFadden, T. and Balasubramaniam, S. (2005) 'Middleware for distributed context-aware systems', in *Int. Symposium on Distributed Objects and Applications (DOA)*, pp.846–863, Springer-Verlag.

Herrmann, K. (2004) 'MESHMdl – a middleware for self-organization in ad hoc networks', Technical Report, Berlin University of Technology, available at http://www.ivs.tu-berlin.de/ kh/publikati-onen/MDC2003.pdf (accessed on 11 February, 2010).

Herrmann, K. (2006) 'Self-organizing infrastructures for ambient services', PhD thesis, University of Stuttgart, Institute of Parallel and Distributed Systems (IPVS).

Hull, R., Neaves, O. and Bedford-Roberts, J. (1997) 'Towards situated computing', in *Proc. of the First Int. Symposium on Wearable Computing (ISWC'97)*, pp.146–152, Cambridge.

Hung, N.Q., Ngoc, N.C., Hung, L.X., Lei, S. and Lee, S.Y. (2003) 'A survey on middleware for context-awareness in ubiquitous computing environments', *Korean Information Processing Society Review, Processing Society (KIPS) Journal*, 3 July, pp.97–121.

Jung, J., Park, J., Han, S. and Lee, K. (2007) 'An ECA-based framework for decentralized coordination of ubiquitous web services', *Inf. Softw. Technol.*, Vol. 49, Nos. 11–12, pp.1141–1161.

Kawsar, F., Fujinami, K. and Nakajima, T. (2009) 'Prottoy middleware platform for smart object systems', *Special Issue on New Advances and Challenges in Smart Home, Int. Journal of Smart Home*, Sci. & Engineering Research Support Center (SERSC), Vol. 2, No. 3, pp.1–18.

Kawsar, F., Masum, M.A. and Nakajima, T. (2008) 'Applying commonsense to augment user interaction in an intelligent environment', in *Proc. of the 4th IET Int. Conf. on Intelligent Environment (IE08)*, pp.1–6, Seattle, WA.

Kephart, J. and Chess, D. (2003) 'The vision of autonomic computing', *IEEE Computer*, Vol. 36, No. 1, pp.41–50.

Kjr, K.E. (2007) 'A survey of context-aware middleware', in *Proc. of the 25th Conf. on IASTED Int. Multi-Conf.*, pp.148–155, Software Engineering, Innsbruck, Austria.

Kuhn, E., Mordinyi, R., Keszthelyi, L., Schreiber, C., Bessler, S. and Tomic, S. (2009) 'Aspect-oriented space containers for efficient publish/subscribe scenarios in intelligent transportation systems', in R. Meersman, T. Dillon and P. Herrero (Eds.): *Proc. of the Confederated International Conferences, CoopIS, DOA, IS, and ODBASE 2009, on On the Move to Meaningful Internet Systems: Part I (OTM '09)*, pp.432–448, Springer-Verlag, Berlin, Heidelberg.

Le Sommer, N., Guidec, F. and Roussain, H. (2006) 'A context-aware middleware platform for autonomous application services in dynamic wireless networks', in *Proc. of the First Int. Conf. on Integrated Internet Ad Hoc and Sensor Networks (InterSense '06)*, Article 9, ACM, New York, NY, USA.

Leguay, J., Lopez-Ramos, M., Jean-Marie, K. and Conan, V. (2008) 'Service oriented architecture for heterogeneous and dynamic sensor networks', in *Pro. of the Second International Conference on Distributed Event-Based Systems (DEBS '08)*, ACM, New York, NY, USA, pp.309–312.

Lieberman, H. and Selker, T. (2000) 'Out of context: computer systems that adapt to, and learn from, context', *IBM Systems Journal*, Vol. 39, Nos. 3–4, pp.617–632.

Loyall, J.P. (2003) 'Emerging trends in adaptive middleware and its application to distributed real-time embedded systems', in *Embedded Software, Lecture Notes in Computer Sci., 2003*, Vol. 2855, pp.20–34.

Mahmud, N., Vermeulen, M., Luyten, J. and Coninx, K. (2009) 'The five commandments of activity-aware ubiquitous computing applications', in Duffy, V.G. (Ed.): *Proc. of the 2nd Int. Conf. on Digital Human Modeling: Held as Part of HCI Int. 2009 (ICDHM '09)*, pp.257–264, Springer-Verlag, Berlin, Heidelberg.

Manzalini, A. and Zambonelli, F. (2006) 'Towards autonomic and situation-aware communication services: the CASCADAS vision', in *Proc. of the IEEE Workshop on Distributed intelligent Systems: Collective intelligence and its Applications*, pp.383–388, IEEE Computer Society, Washington, DC.

Marples, D. and Kriens, P. (2001) 'The open services gateway initiative: an introductory overview', *IEEE Communications Magazine*, Vol. 39, No. 12, pp.110–114.

Mauro, C., Leimeister, J.M. adn Krcmar, H. (2010) 'Service oriented device integration – an analysis of SOA design patterns', in *Proc. of the 2010 43rd Hawaii Int. Conf. on System Sci (HICSS '10)*, pp.1–10, IEEE Computer Society, Washington, DC, USA.

Meier, R. and Cahill, V. (2002) 'STEAM: event-based middleware for wireless ad hoc networks', in *Proc. of the Int. Workshop on Distributed Event-Based Systems (ICDCS/DEBS'02)*, pp.639–644, Vienna, Austria.

Mills, K.L. and Scholtz, J. (2001) 'Situated computing: the next Frontier for HCI research', *Book Chapter HCI in the New Millennium*, pp.537–548, Addison-Wesley.

MIT (2010) Available at http://ttt.media.mit.edu/research/research.html (accessed on January).

OASIS (2009) 'Devices profile for web services', Version 1.1, OASIS Standard, available at http://docs.oasis-open.org/ws-dd/dpws/1.1/os/wsdd-dpws-1.1-spec-os.pdf (accessed on 5 February 2010).

OMA (2006) 'OMA web services enabler (OWSER): overview', OMA-AD-OWSER Overview-V1 1-20060328-A, available at http://www.openmobilealliance.org/releaseprogram/ owserv1 1.html (accessed on 20 March 2010).

Pascoe, J. (1998) 'Adding generic contextual capabilities to wearable computers', in *Proc. of the 2nd IEEE Int. Symposium on Wearable Computers (ISWC '98)*, pp.92–98, IEEE Computer Society, Washington, DC, USA.

Pham, T-L., Schneider, G. and Goose, S. (2000) 'A situated computing framework for mobile and ubiquitous multimedia access using small screen and composite devices', in *Proc. of the eighth ACM Int. Conf. on Multimedia (MULTIMEDIA '00)*, pp.323–331, ACM, New York, NY, USA.

Priyantha, K. and Hirakawa, M. (2000) 'Situated computing: a paradigm to enhance the mobile user's interaction', *Handbook of Software Engineering and Knowledge Engineering*, pp.34–40.

Ranganathan, A. and Campbell, R.H. (2003) 'A middleware for context-aware agents in ubiquitous computing environments', in Endler, M. (Ed.): *Proc. of the ACM/IFIP/USENIX 2003 Int. Conf. on Middleware (Middleware '03)*, pp.143–161, Springer-Verlag New York, Inc., New York, NY, USA.

Rodriguez, M., Favela, J., Preciado, A. and Vizcaino, A. (2004) 'An agent middleware for supporting ambient intelligence for healthcare', in *Second Workshop on Agents Applied in Health Care (ECAI 2004)*, pp.67–73, Valencia, Spain.

Roman, M. and Campbell, H. (2000) 'GAIA: enabling active spaces', in *Proc. of the 9th ACM SIGOPS European Workshop*, pp.229–234, Kolding, Denmark.

Schmutzler, J., Rohde, S. and Wietfeld, C. (2009) 'Integration of wireless peer-to-peer sensor networks with embedded web services', ITG Fachtagung-Mobilkommunikation, Osnabrck, Germany, available at http://www.vde-verlag.de/buecher/ivz/ivz3164.pdf (accessed on 23 January 2010).

Schwiderski-Grosche, S. and Moody, K. (2009) 'The SpaTeC composite event language for spatio-temporal reasoning in mobile systems', in *Proc. of the Third ACM Int. Conf. on Distributed Event-Based Systems (DEBS '09)*, pp.1–12, Nashville, Tennessee, USA.

Yau, S.S., Karim, F., Wang, Y., Wang, B. and Gupta, S.K.S. (2002) 'Reconfigurable context-sensitive middleware for pervasive computing', *Pervasive Computing*, available at http://computer.org/pervasive (accessed on 20 January 2010).

Zachariadis, S., Capra, L., Mascolo, C. and Emmerich, W. (2002) 'XMIDDLE: information sharing middleware for a mobile environment', in *Proc. of the 24th Int. Conf. on Software Engineering (ICSE 2002)*, pp.712–718, Las Vegas, NV, USA.

Zaera, M. (2008) 'Wave-based communication in vehicle to infrastructure real-time safety-related traffic telematics', Master's thesis, Telecommunication Engineering, University of Zaragoza, August.

Zambonelli, F. and Mamei, M. (2005) 'Spatial computing: an emerging paradigm for autonomic computing and communication', *Autonomic Communication, Lecture Notes in Computer Science*, Vol. 3457/2006, pp.44–57, Springer Berlin/Heidelberg.

Zimmermann, K., Felis, S., Schmid, S., Eggert, L. and Brunner, M. (2006) 'Autonomic wireless network management', *Autonomic Communication Lecture Notes in Computer Sci., 2006*, Vol. 3854, pp.57–70.