11 The DiY Smart Experiences Project

A European Endeavour Removing Barriers for User-generated Internet of Things Applications

Marc Roelands[1], Johan Plomp[2], Diego Casado Mansilla[3], Juan R. Velasco[3], Ismail Salhi[4], Gyu Myoung Lee[5], Noel Crespi[5], Filipe Vinci dos Santos[6], Julien Vachaudez[6], Frédéric Bettens[6], Joel Hanqc[6], Carlos Valderrama[6], Nilo Menezes[7], Alexandre Girardi[7], Xavier Ricco[7], Mario Lopez-Ramos[8], Nicolas Dumont[8], Iván Corredor[9], Miguel S. Familiar[9], José F. Martínez[9], Vicente Hernández[9], Dries De Roeck[10], Christof van Nimwegen[10], Leire Bastida[11], Marisa Escalante[11], Juncal Alonso[11], Quentin Reul[12], Yan Tang[12], Robert Meersman[12]

1 Alcatel-Lucent Bell Labs, Antwerp, Belgium

2 VTT, Helsinki, Finland

3 UAH, Madrid, Spain

4 ENSIE, France

5 Institut Telecom SudParis, Paris, France

6 UMONS, Mons, Belgium

7 Multitel asbl, Mons, Belgium

8 Thales, Paris, France

9 UPM, Madrid, Spain

10 CUO, Leuven, Belgium

11 ESI, Bilbao, Spain

12 STARLab, Brussels, Belgium

Abstract In this chapter we discuss the wide range of challenges in user-generated Internet of Things applications, as being worked on among the large consortium of the *DiY Smart Experiences* (DiYSE) project (DiYSE, ITEA2 08005). The chapter starts with a discussion on the context of 'DiY' as a phenomenon to be leveraged, and eco-awareness as an example application area. The main body of the chapter is devoted to the technical outline of the DiYSE architecture, starting at the lower Internet of Things layers of sensors, actuators and middleware, over the role of semantics in device and service interoperability, up to requirements for the service framework and the application creation process. Furthermore, the chapter adds

D. Uckelmann et al. (eds.), *Architecting the Internet of Things*,
DOI 10.1007/978-3-642-19157-2_11,

considerations concerning tangible interaction in the smart space, assumed in DiYSE both for the context of experiencing as well as shaping the user experience. With the chapter, we thus take a holistic view, sampling the range from lower-layer technical implications of enabling DiY creation in the Internet of Things, up to the human-level aspects of creative communities as well as tangible interaction.

11.1 Drivers, Motives and Persona in the DiY Society

With the 'DiY society' (Von Hippel 2005) a world is imagined where anybody could become a creator of objects. With the DiYSE project taking the Do-it-Yourself (DiY) phenomenon as a starting point, we discuss its broader context in this section.

At first sight, the idea of creating objects might seem like nothing new. People have been creating things from the very start of civilisation, dating back to the prehistoric ages where people created very basic tools out of materials at their disposal. Ever since, the process of creating things has evolved and has become more complex, as the world and society itself became more complex (Sterling 2005). If we make a time warp to today's modern world, we see that the introduction of technology into our lives is at least one of the aspects that have influenced the way we create, use and perceive objects. Computerised systems are nowadays allowing us to create very complex products that not everyone is capable of creating from scratch anymore. In order to incorporate a computerised, electronic system into an object a certain amount of expertise is needed for programming the system or to integrate the various hardware and software elements.

So, a major challenge to make the DiY society possible is to make people more capable of creating meaningful objects again in the context of today's object complexity, beyond the intended use as driven and orchestrated by solution vendors, opening up e.g. the physical and electronic customisation possibilities. In an optimal *utopian* scenario, this means that the creation of technological and purely physical 'analogue' products should be a seamless activity, allowing people to create things that enhance their lives in a pervasive world. The way this process of creation is done by someone is inherently linked to characteristics such as personal background, intention, expertise and motivation.

Of course, all this is to be seen in the context of a complete next generation 'manufacturing' ecosystem, of which the viability depends on finding a sustainable balance, a multi-sided 'win-win', between the various involved actors. This will eventually determine the economical, next to the evident societal impact.

11.1.1 Evolution of DiY

Recently, Do-it-Yourself as a phenomenon has, again, started to take the central stage in research and development. To understand why this is the case, several things can be learned from the history of DiY and can be projected onto how the phenomenon could be perceived in the future.

11.1.1.1 The Past

Looking at the past, DiY can be seen as a variety of activities. Obviously people have been making objects themselves since the prehistoric ages, but looking at the evolution in history regarding the creation of objects one can observe several elements that had significant impact in how we approach DiY today. A good illustration of this is the way objects were created in the Middle Ages, as at that time people started having a *marketplace* to buy and sell things and there were established communities to share and learn new skills (Sennet 2008). In the Middle Ages, the creation of things was mostly done by *skilled craftsmen* who grouped together in *guilds*. In order for someone to 'learn' how to create something one had to go through a learning process in which a 'master' taught his skills to one or more apprentices.

Since then, we have evolved into a society where skills knowledge is more distributed among people and is less confined to one person or group. When nowadays the term DiY is coined, often the first associations made are about shops selling home improvement materials and people refurbishing their houses themselves.

11.1.1.2 The Possible Future

With the advent of computers and in particular of the Internet, the notion of DiY has taken new dimensions. First of all, it is now a lot easier to share and talk about DiY activities of all kinds through dedicated online community platforms (Dormer 1997). Secondly, more and more people are creating their own electronics, both hardware and software. Both these facts result in an increasing *accessibility* of technology, making tools available for people to enhance the quality of their lives on other levels than the purely functional.

11.1.2 Why Do People Build Things Themselves?

A central question that still remains is why people would at a certain point decide to build something themselves. There are at least two ways to approach this ques-

tion. On the one hand it can be seen as part of a *motivational psychology*, where a person does something based on intrinsic motivation. This means that the motivation comes from the person himself wanting to solve a problem in his own life for example, possibly but not necessarily with cost savings in mind[121]. On the other hand, DiY can be interpreted on various levels depending on the background of the person or people involved in the DiY activity, the so-called types of 'people logic'. For instance, a person customising his bought shoes is on a different level than a person who is creating shoes from scratch.

11.1.3 People Motivation as Driver

A major driver behind the reason that people at some point decide to create something themselves instead of buying a ready-made solution from a shop is the relationship they create with the thing they created themselves. In the context of interior decoration, Elizabeth Shove describes this type of motivation as follows: "The house objectifies the vision the occupants have of themselves in the eyes of others and as such it becomes an entity and process to live up, give time to, and to show off. What is important are end results, not the actual physical involvement in the tasks and projects of 'doing it yourself'" (Shove et al. 2007). Doing something yourself allows people to identify and relate to objects on a much deeper level than merely the functional. Von Hippel (2005) also states that "A thing is not merely a material object, but a frozen techno-social relationship", which points out that the relation between a person and an object is something quite delicate. This emotional link between a person and an object is what defines the meaning a person gives to something. It is this process of giving meaning that is highly stimulated through DiY activities.

11.1.4 People Logics, Distinguishing Motivation Levels

With regard to the previously mentioned motivational aspects, it should be nuanced that it does not work the same way for all people. The concept of DiY can be approached and understood by various people on different levels, depending on their personal background, personal skill or experience. To understand and comprehend these levels better, it should be made clear that what really matters in a

[121] Note that, while one would expect DiY activities to *save* costs to the one performing it, as he is the one investing time, effort and creativity, the modern DiY in many cases rather is motivated by feelings of 'ownership', 'passion for creating' and other psychological motives as discussed here. So DiY often implies a willingness to pay which is higher as compared to buying off-the-shelf products solving the same problem. Both *cost saving* and *spending* have a place in the DiY societal phenomenon.

DiY activity is the *mindset of a person*. Depending on the way people think about a subject, they will interpret it as being something, DiY or not. We here use the concept of 'people logics' introduced by Mogensen to illustrate this (Mogensen 2004):

- **Industrial logic:** This way of thinking is mostly straightforward, no-nonsense. In order for people of this kind to have a drive for DiY, a very small action would be needed. For example, mounting a device on the wall may give such a person a feeling of satisfaction.
- **Dream society logic:** In the dream society, people do things in order to show themselves to the outside world. Thinking about DiY from such perspective, a deep customising of a product could suffice to trigger the feeling of 'I did this myself'. This could be, for example, choosing the colour and materials of a pair of shoes.
- **Creative man logic:** The creative man wants to create things from scratch by himself based on his own personal needs. Starting from this point of view, this person could follow an *instructable* to create his own windmill to provide power to his house, as an example.

The logics presented here may need to be extended to cover every possible aspect of DiY, but the main point is that approaching people based on their mindset may proof to be the key to getting the masses to engage in (Internet of Things) DiY activities.

Next to the DiY mindset of people, as the main and basic driver, we mentioned before that this is to be seen in the context of overall *ecosystem dynamics*, where *economical constraints* are into play. DiY for the practitioner in fact can mean a cost saving or can be rather a higher spending for the same problem solved, depending on the degree and level of motivation as discussed. This last case is an obvious opening to business opportunities, as in fact leveraged for years already in the creative and *hobby crafting* sector as well as by vendors of high-end *modular* systems in various domains.

But with the evolution to cheaper and more accessible electronics, and the potential to easily connect wirelessly and ubiquitously to the internet, fuelling the Internet of Things as a grassroots economic platform, DiY may also become a *game changer*, forcing many product and solution vendors to reconsider opening up to their products to customisation and interconnectivity as a quality, rather than pursuing 'locking in' consumers into a single-vendor buying track. In fact, this is what the *Institute for the Future* (IFTF) predicts in their map on *'The Future of Making'* (IFTF 2008). As with other market evolutions, commercial actors anticipating taking a strategic role in such a new ecosystem – a 'Web 2.0 of the Internet of Things' – may develop a clear first-mover advantage, comparable to what happened with the *Apple iPhone App Store*.

One particular theme, driven by economical but also broader societal choices, is *eco-awareness*. The following sections elaborate on this as an example area of DiY Internet of Things activities.

11.1.5 Eco-awareness, an Example Application Theme in DiYSE

One example area where new DiY user-generated applications could have a large socio-economic impact is the theme of *eco-awareness*, including but not limited to energy-efficient infrastructure. One scenario cluster in the DiYSE project considers leveraging user-generated pollution data and possibly also safety-related data in the city for community-building of mass-consumable applications, supporting this societal awareness. Another set of applications considered is about energy-efficient comfortable living, with energy consumption monitoring and control using smart objects. In this section, we discuss the requirements for applying the DiY concept to this example area.

11.1.5.1 Energy Consumption in a DiY Internet of Things

With the emergence of the Internet of Things, everything is becoming connected, and so, networks have evolved from primarily a source of information to the most important platform for many types of applications, involving all kinds of devices and objects. Likewise, connected communities of people using the 'connecting to anything' capability of the Internet of Things are also expected to grow more and more. Therefore, the need is emerging for solutions for interdisciplinary fusion services that combine Information Technology (IT) with other technologies.

Among several applications for interdisciplinary fusion services in relation to ecological themes, aspects such as *energy harvesting* and *low power consumption* are also quite important elements for Internet of Things smart experiences to become a reality. Current technology seems inadequate for the emerging low-power processing requirements. The development of new and more efficient and compact energy storage like fuel cells, printed/polymer batteries, etc; as well as energy generation devices, coupling energy transmission methods or energy harvesting using energy conversion, will be pivotal for implementing autonomous wireless smart systems.

In the DiYSE project we address the challenge of eco-awareness for energy efficiency by making it more tangible to people, and more 'DiY' in people's mindset by introducing network-connected smart objects in the setting. Taking advantage of this paradigm, one can indeed imagine that *consumers start monitoring* their energy consumption and thus better understand how their habits relate to their energy consumption. This not only provides a more fine-grained picture of the energy consumption in houses, buildings and vehicles to the *energy suppliers*,

but ultimately, with DiY involvement, it also provides citizens with impactful participation means, *sharing good practices* and energy saving 'tricks' using self-made hardware or software enhancements, fuelling a collective green society mindset. Also, energy suppliers would be able to interact with their customer households in a less 'black-and-white' fashion, for example activating appliances that consume much energy, such as washing machines or laundry dryers, at times when energy can be produced and provided in ways that is environmentally friendly and well-priced. With service creation technology around connected smart objects, as researched in DiYSE, a lot more could be offered, like tracking energy consumption peaks, providing consumer notifications on appliances still running possibly inadvertently, or other, more complex applications for which personalisation is an essential factor in mass market acceptance.

11.1.5.2 DiY Engagement in Eco-aware Applications

In the *home environment*, the big paradigm shift could come when every smart object knows the interoperable protocols, removing the need for the dedicated systems developed independently today. Even beyond that, without putting any pre-established 'high-end' solution in place for which – a priori – cross-system interoperability standards would have been established, building intrinsically more *open* systems – in a DiY Internet of Things fashion – would encourage inhabitants to participate effectively in an ecological engagement. For example, maintaining a comfortable temperature and heating of water are the most energy consuming tasks in a typical house, with a dramatic potential for energy conservation and as a consequence a potentially significant positive impact on the environment. With the family engaging in more elaborate 'self-configuration' as a DiY activity, the house could become so fine-tuned, that the comfort of each of its inhabitants is simultaneously maximised through learning the individual preference profiles, while keeping energy consumption within desired limits. In the *vehicle environment*, smart objects in the car will be able to manage better the energy needed. Optimal route planning will reduce the distance driven, and better control systems for the car itself will make the ride more energy efficient, all combined contributing to reduced emissions and less pollution. Here also, awareness among citizens can be amplified by giving them the means to customise the experience, and even contribute data and measurements to related electronic communities.

Among several applications envisioned in DiYSE, Figure 11.1 shows an applications overview on the theme of eco-awareness and energy efficiency. For the home and building environment, objects such as energy saving controllable sockets, smart metering, and home automation controllers are used for energy management. In vehicles, devices taking part in the navigation control, and devices for safety, can be used for energy saving.

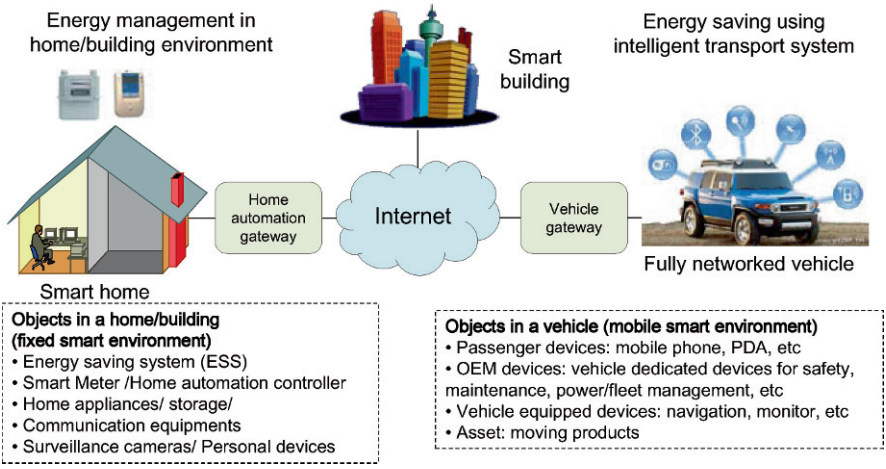

Fig. 11.1 Applications Using Eco-awareness for Energy Efficiency

The smart home and smart building in fact may cover a wide range of services, applications, equipment, networks and systems that act together in delivering the *'intelligent' environment* for domains such as security and control, communications, leisure and comfort, environment integration and accessibility. Particularly smart building entails a suite of technologies used to make the design, construction and operation of buildings more efficiently, applicable to both existing and newly built properties (GeSI and The Climate Group 2008). Example systems are building management systems (BMS) that run heating and cooling systems according to occupants' needs, or software that switches off all computers and displays after everyone has gone home. BMS data can be used to identify additional opportunities for efficiency improvements.

So, various concepts and approaches are possible in optimising the energy efficiency of buildings and homes, leveraging the intelligent building control in an attractive cost-benefit ratio. A few example applications from this view are:

- *intelligent/automated light control*, allowing users to lighten their homes before entering, both for safety and to create a welcoming environment, or to mimic activity while away – even reconfiguring activities remotely when away from home / office;
- *auto-regulation of heating based on non-occupancy detection*, maximising energy savings, while remote temperature controls still allow for adjustments; and
- *media/entertainment control*, integrated more comfortably with home activity compared to stand-alone entertainment systems, and remotely accessible.

In these examples, the home or office becoming a 'smart space' with DiY Internet of Things capabilities would allow much easier 'programming by doing' configuration of the otherwise (semi-)professional home automation configura-

tion, and allow for originally unforeseen improvements from community 'wisdom of the crowd'.

Automotive transport, as the other area of applications mentioned, represents one of the main sources of green house gas emissions, but with the generalised availability of ultra-high-speed broadband access and the ubiquitous provision of next generation mobile telecom services, many tasks and movements could be coordinated much better, for minimising power consumption. While the main focus of applying ICT to transport through the development of *Intelligent Transport Systems* (ITS) is *safety*, the efficiency management of transport systems through ITS can also reduce the *environmental impact* of transportation. Example applications for this area currently considered by industry (ITU-T 2008) are:

- *enhanced navigation or vehicle dispatch*, considering alternative routes, possibly proactively, reducing journey time and energy consumption;
- *parking guidance systems*, additionally reducing engine time;
- *road pricing schemes*, such as the congestion charge applied in London, encouraging use of public transport during congestion periods.

Furthermore, *vehicles* can serve as mobile environmental pollution *sensors*, and electrical vehicles can play an important role as *energy storage*, production or consumption elements in smart energy grids.

Here again, the engagement of people can be dramatically improved, by providing the flexibility of high (DiY) personalisation, and even having them actively contribute to the roll-out of the mobile and fixed pollution sensing infrastructure by sharing personal data and devices. A car vendor, for example, opening up the car information system to community driven sensor (and other) applications could have a unique selling point in enhanced in-car navigation, with this maybe even becoming a must-have feature when the market evolves.

11.1.5.3 Requirements for Enabling DiY in Eco-awareness Applications

In light of the range of possible eco-awareness applications, a number of further requirements on capabilities at various levels need to be considered, in order for such DiY, sometimes crowd-sourced value to become possible, for example with respect to:

- *easy installation and integration of everyday objects* in our home environment, locally as well as remotely, for monitoring and control purposes, in particular for intelligent energy consumption monitoring and control functions;
- *public IP network data connectivity and access*, directly or indirectly, to all objects involved in the service or application;
- *device* virtualisation, installation and *provisioning*;

- *meaningful and permanent, globally unique identification of objects*, allowing for the linking of devices into associated functions, and the unambiguous derivation of meaningful information from raw sensor data;
- *search and discovery* of suitable, available appliances according to properties and capabilities;
- *web-based* information processing, notification and *visualisation*;
- *personalisation* of associated services, considering *context information,* such as in personal home energy profiles and scenes;
- support for a *creation workflow* entailing activities involving all the above;
- *secure access* limitation or sharing of personal or household data across Internet; and
- *identity-based user management*.

Such groups of requirements may be extrapolated to get to *generic* requirements as need to be covered for *any* creation architecture on top of the Internet of Things as is aimed at in the DiYSE project. Further in this chapter, specific solution parts of DiYSE are discussed, addressing these requirements as relevant for diverse use cases.

11.1.5.4 Technologies and Standards Relevant for DiY Eco-awareness

While energy efficiency in buildings clearly would benefit environmental sustainability, there is still a technological barrier to DiY creation by the masses on this theme, or, as a start, just even for involvement of all related non-IT professional parties. Therefore, a way must be found to *disseminate* and promote technological *good practices* for energy efficient buildings – a typical ingredient of a DiY community phenomenon – and to increase the accessibility for non-technical experts to a range of available technologies.

As was introduced already in the eco-awareness examples in previous sections, this could be highly accelerated by the availability of properly standardised, general and domain-specifically profiled, *interoperable protocols* for smart objects and associated applications, in order to keep users agnostic from the underlying technology.

A number of existing technologies and standards should therefore be taken into account, in a first step to enabling DiY creation on top of the Internet of Things:

- *Web technologies*, including information processing, notification and visualisation, but also so-called *mash-up* technologies, should be easily usable in combination with the Internet of Things, as a basic communication platform like needed for example in energy efficiency in building automation and smart homes. In fact, the leveraging of this technology has led to early definitions of a *Web of Things*, in which a REST-based convention is taken as a first step to realise *physical mash-ups* (Guinard et al. 2009).

- *Internet of Things and device application programming interface (API)* workgroups in standards bodies like Internet Engineering/Research Task Force (IETF[122]/IRTF[123]), International Telecommunication Union - Telecommunication Standardisation Sector (ITU-T)[124] and World Wide Web Consortium (W3C)[125], play a pivotal role.
- *Service deployment, service life cycle management and device management* standards alliances like the Open Service Gateway initiative (OSGi) Alliance[126], the Universal Plug-and-Play (UPnP) Forum[127], and the Digital Living Network Alliance (DLNA) for home devices configuration and functional abstraction.
- Beyond this, the Internet of Things Research Cluster (IERC)[128] is making an effort to concertise standardisation and interoperability activities among the many European projects working around the Internet of Things. In this context, subject of debate is, for example, the *unique identifiers for objects*, which, while mainly stemming from early Internet of Things applications in logistics and supply chain management, are also required when extending the eco-awareness theme into the DiY realm. Related especially to naming and addressing for the Internet of Things also is the work of the European Telecommunications Standards Institute (ETSI) Technical Committee on Machine-to-Machine Communication (TC M2M).
- *Optimisation techniques from Cloud Computing* could particularly be applied and combined with the notion of Internet of Things in the context of smart energy grids, as power needs to be 'routed' according to the distributed fluctuations in energy capacity and consumption needs, requiring bidirectional, real time information exchange among customers and energy management operations.

11.2 Sensor-actuator Technologies and Middleware as a Basis for a DiY Service Creation Framework

For applying the freedom of creativity of Web 2.0 to the Internet of Things, as aimed at in the DiYSE project, it is essential that non-expert users are enabled to easily search for public devices or share their own, privately bought or DiY-built

[122] http://www.ietf.org/
[123] http://www.ietf.org/
[124] http://www.itu.int/ITU-T/index.html
[125] http://www.w3.org/
[126] http://www.osgi.org/
[127] http://www.upnp.org/
[128] http://internet-of-things-research.eu/

devices, and as such personalise the physical environment by combining and 'mashing-up' device functions, regardless of whether the system has prior knowledge about the devices or not. A large, heterogeneous set of device types needs to be considered, with devices 'speaking' a wide range of 'languages', having varying specific constraints in terms of mobility, battery, computation, etc., and serving different usages, the same device even having different purposes in different contexts for different users, all in a constantly evolving manner.

Traditional computing approaches are not intended to cope with such complexity. Therefore, this section explores how DiY service creation environments, as envisioned in the DiYSE project, can deal with *plug-and-play* connectivity of heterogeneous device types, how the function of appearing devices can be understood, and how the data generated by these devices can be interpreted.

11.2.1 Device Integration

In the following subsections we introduce the notion of enhanced device drivers, as a means of first-level abstraction for heterogeneous device types, and the DiYSE Gateway, serving as a proxy for resource-constrained devices. Finally, we discuss ways to identify and address the discovered devices.

11.2.1.1 A first Level of Abstraction Addressing Device Heterogeneity

As a DiY creation system needs to support legacy devices, and cannot assume that future devices will respect any specific standard, the only viable solution is to make the system accept *any* kind of device interface and *describe* it using a common 'meta-language' understandable by a machine.

A similar abstraction mechanism is commonly used for peripherals in every computer operating system, and is known as a *device driver*. It contains only the programming interfaces required for the system to communicate with the device, while hiding specific implementation differences within one device class. However, the device driver does not contain any information about the different ways of using the device.

As an example, a user may want to control his motorised pan-tilt-zoom camera using a *WiiMote* controller and gestures. This interaction may seem conceptually straightforward for a human being, but technically it is unfeasible for a non-expert user unless the specific software exists. It may seem obvious to a human that both devices could 'talk' about pointing a given direction, but machines need additional knowledge to achieve it. The information about the *meaning* of actions such as *'get pointed direction'* or *'turn to direction',* required for their automatic mapping, needs to be provided by a human in every case because there is no computer algorithm enabling to find a logical relationship between those.

One solution as investigated in the DiYSE project is to *embed* the knowledge about the capabilities of a device in an *enhanced* driver, so that it is understandable by machines without low-level programming intervention. In the example, the *WiiMote* driver would be augmented with information such as *'can control direction'*, whereas the camera would have *'can have its direction controlled'* as a property exposed by the enhanced driver. Semantic reasoning mechanisms, as discussed later in this chapter, would use such conceptual information to assist the user in describing device interactions that make sense conceptually and are at the same time technically feasible and well-described, so that the desired device interaction is executable without the need to develop dedicated software. For instance, in a most basic scenario not even considering the higher layer semantic reasoning capabilities of an application creation environment above it, a straightforward request to the system to link the *WiiMote* and the camera can already default to the automatic realisation of the intuitively expected interaction of controlling the orientation of the camera by means of the *WiiMote*. So, for such basic scenarios, behaviour creation can be as simple as defining a *Lego* crane control, without such control being predesigned as a fixed function, only relying on the basic semantic annotations obtained from the enhanced device driver.

With this approach, the problem of complexity due to the heterogeneity of devices is solved at a low-level stage. Even at this basic, not further enhanced level, non-expert users will not experience a barrier of low-level technical details or compatibility issues anymore.

Beyond this, in the context of DiY creation of Internet of Things applications, as an important potential enabling element for the DiY Internet of Things ecosystem, web communities are envisioned to emerge in which experts can publish and enrich enhanced drivers, so that the spectrum of possible applications constantly broadens, including newly supported devices as well as new ways of applying existing device features.

11.2.1.2 Achieving Device Data Connectivity for Resource-constrained Devices

Despite the progress in leveraging the IPv6 protocol for connecting smart objects into the cloud, in the foreseeable future many, in one or multiple aspects *resource-constrained* devices will remain supporting only dedicated – but nevertheless often standardised – protocols specifically designed for the resource-constrained nature of the devices. Examples are *Wireless Sensor Network* nodes, or *Zigbee* or *Bluetooth* peripherals, for which hardware cost, related to memory and processing power, but especially also energy consumption are important factors.

As such resource-constrained devices are also considered *key* in the DiYSE context, the project considers an *intermediate gateway function* for exposing also these devices in a uniform way to the overall framework and make them IP-addressable, to connect them into local or global IP networks for data retrieval,

control, and device management. Such a *DiYSE gateway*, as we named this function, requires a flexible abstraction layer hiding the underlying network technology heterogeneity, while supporting fast and seamless device deployment. Also, this layer should relay unique identification of the devices, for transparent interoperability and remote device querying, control and monitoring. Figure 11.2 shows the main modules of a DiYSE Gateway, distinguishing:

- a discovery module for devices being plugged in,
- means to install and execute enhanced drivers, and
- the bookkeeping of connected devices both for keeping track of local execution and for southbound device exposure.

For devices that directly connect into the cloud via IP, equivalent functions for proper exposure to the middleware may be provided also, according to a notion that we could call a *cloud DiYSE gateway*.

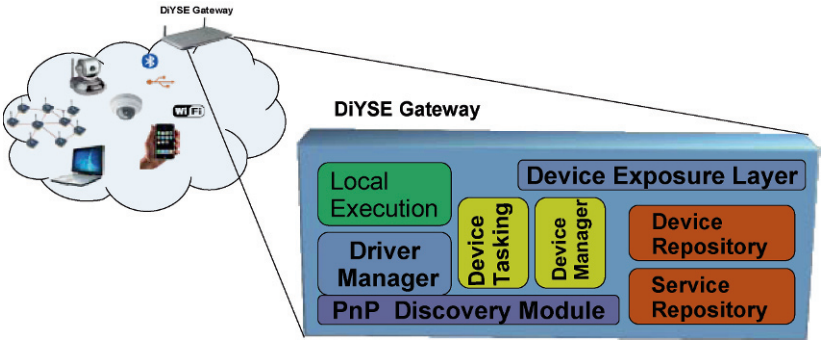

Fig. 11.2 Main Modules of a *DiYSE Gateway* Function

In the next subsection we discuss some further aspects of the installation, registration, and integration of these resource-constrained devices, or *nodes* as we call them.

11.2.1.3 From Hardware to Device Description

As previously described, we consider the use of *enhanced drivers* to cope with the high heterogeneity of *nodes*. Typically for each new device, the device manufacturer, or an expert developer, can make available appropriate driver software, which eventually gets automatically installed on local DiYSE gateways, thus providing a description of device capabilities and an interactions-set that users, or other devices, may leverage.

For this automatic driver installation, a DiYSE gateway triggers a *driver-lookup* operation among the different repositories by using uniquely characterising

meta-data extracted from the new node, like type of device, vendor, hardware MAC address, etc.. On top of the base set of handling functions that devices from the same hardware family may have, specific additional handlers may be required for managing additional functionalities, like for instance a special night-capture function that only a specific type of pan-tilt-zoom camera may have.

A further essential requirement for the integration of large numbers of devices in the cloud is the capacity to individually *identify and address* them. Several solutions for that can be considered, such as the use of a generic syntax like *Uniform Resource Identifier* (URI, RFC 3986[129]) as a permanent and unique identifier included in the device description, or, alternatively, the association of an *IPv6 address* with every node, or still, the use of *application level identifiers* on top of the network addresses, like logical peer-to-peer (P2P) identifiers or Dynamic DNS. As many nodes may however not be able to store or compute their identifiers, such operation often needs to be performed on the connecting DiYSE gateway.

In the DiYSE architecture, also a device discovery service is foreseen, providing high-level descriptions of the capabilities and the services that the installed nodes can provide. While more and more devices today are discoverable via direct embedded support for communication protocols like UPnP[130], DPWS[131] or DLNA[132], most of today's devices in the surroundings remain to be incompatible or not equipped with such self-description mechanisms. Thus, DiYSE gateways use the enhanced drivers to map the node functionality and attributes into a *common description language* like DPWS for remote description of connected nodes, and offer the functionality to expose devices and services and send events beyond the local network domain boundaries across the internet.

11.2.2 Middleware Technologies Needed for a DiY Internet of Things

A middleware, being a software infrastructure that ties together hardware, operating systems, network stacks, and applications, should provide a runtime environment supporting functions such as multi-application coordination, standardised system services (e.g. data aggregation, control and management policies), and mechanisms for adaptive, efficient resource handling. As such, middleware support is essential for interworking with so-called *Reduced Functionality Devices* (RFDs), such as DiYSE *nodes,* which are by definition resource-constrained devices, and which moreover are using one out of a heterogeneous range of commu-

[129] http://www.ietf.org/rfc/rfc3986.txt

[130] http://www.upnp.org/

[131] http://docs.oasis-open.org/ws-dd/ns/dpws/2009/01

[132] http://www.dlna.org/home

nication standards, including *IEEE 802.15.4*, *ZigBee*, *Z-Wave*, *Bluetooth*, and *6LoWPAN*.

In the middleware that exposes RFDs as a set of generic services for applications, we distinguish in the DiYSE project:

- *Low Level Services*, containing the vital, intensively used functions always needed for interaction with the hardware, like *Real-Time Management*, *Communication and Context Discovery Management*,
- *High Level Services*, typically less critical, providing a first level of application support, more adaptable to different scenarios, with functions such as *Query*, *In-node Service Configuration*, and *Command*,
- *Cross-Layer Services*, providing mixed high and low level functions, such as *Reasoning*, *Portable Code Execution Environment*, and *Security*, and
- *Control Services*, providing the middleware's core functions of component deployment and lifecycle management as well as inter-component communication through event communication, by means of entities called *Software Component Container* and *Eventing Service Manager*.

So, one of the interesting research challenges as investigated in DiYSE with respect to middleware for RFDs is to *translate the service-oriented computing (SOC) paradigm to wireless sensor and actuator networks*. The SOC approach is promising for easy assembly and deployment of interoperable, platform and operating system independent services in such networks, but should also fulfil typical additional requirements for smart environments, such as lightweight business logic optimised for low computational overhead and low battery consumption.

In order to evaluate the performance in low-resources devices in the DiYSE framework in terms of processor power, memory size, bandwidth and battery lifetime, the RFDs-based approach has been implemented in a Wireless Sensor Network. As illustrated in Figure 11.3, in order to provide advanced sensor services to the envisioned end user applications, service composition and management tasks are performed in specialised nodes in the sensor network. To this end, sensor nodes with *Broker*, *Orchestrator* and *Trunk Manager* roles has been defined.

Broker nodes represent the interface between the Wireless Sensor Network and external networks. They receive semantic descriptions of the simple services provided by each sensor using lightweight notation languages, such as *Service Mapping Description* (SMD), on one side and receive service requests from external networks on the other side. *Orchestrator* nodes are responsible for implementing a virtual sensor service paradigm. They perform the composition of the offered simple services into potentially complex and sophisticated composed services, using the semantic descriptions of those primitives. The service control plane is implemented in the *Trunk Managers*, which perform tasks associated with service state supervision, such as self-configuration, self-adaptation and self-recovery, in order to increase service availability and network resilience.

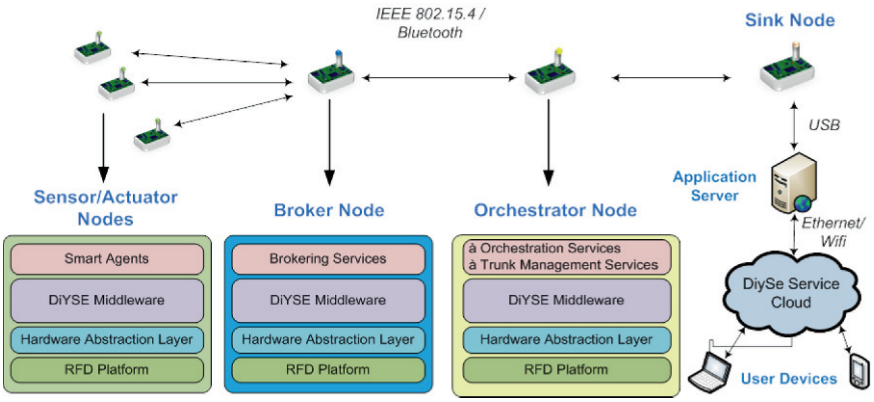

Fig. 11.3 Network Architecture and Middleware for WSANs in DiYSE

11.3 Semantic Interoperability as a Requirement for DiY Creation

As discussed in the previous sections, the interoperability among devices supporting different lower layer communication protocols is a crucial requirement that must be fulfilled to enable the ad-hoc mixing and matching of devices and sensor nodes in DiY applications, as is aimed at with the DiYSE architecture. As *ontologies* have been proposed as a solution not only to provide semantics for the data to be exchanged, but also for describing the devices themselves, they indeed provide the necessary means for compositions of devices in applications – by web-based composition of service-level exposure of the devices, or even beyond that, by locative, on-the-spot creation of applications in the smart space.

In this section, we first introduce the concepts concerning ontologies in computing science, and then describe through examples how the ontologies are envisioned to achieve semantic interoperability in an Internet of Things creation environment as researched in DiYSE.

11.3.1 Ontology

Ontology, as a discipline, is the branch of philosophy that is concerned with the nature of things that exist in the universe (Smith 2003). More specifically, it is the science that aims to provide an *exhaustive* and *definitive* classification of things based on their similarities and differences. By *exhaustive*, we mean that it provides

an explanation for everything that is ongoing in the universe. By *definitive*, we mean that every type of things should be included in the classification. In this sense, the ontology discipline tries to provide answers to the question: *What are the features common to all things?*

In computing science, an *ontology* is commonly defined as: *"a formal, explicit specification of a shared conceptualization"* (Studer et al. 1998). More specifically, an ontology is an engineering artefact composed of *(i)* a *vocabulary* specific to a domain of discourse, and (ii) a set of explicit assumptions regarding the intended *meaning* of the terms in the vocabulary for that domain. This set of assumptions is generally expressed in terms of unary and binary predicates, by which *concepts* and the *relations* between them are expressed. In its simplest form, an ontology defines a hierarchy of concepts related by their taxonomical relationships, whereas in more complex cases, additional relationships can be expressed between concepts to constrain their intended meaning. As an ontology captures knowledge about a domain, this needs to happen *by consensus* among a group of people, in order to reach a common agreement on its conceptualisation.

Different languages have been developed over the last two decades to represent ontologies. For instance, *Simple HTML Ontology Extension* (SHOE) (Luke et al. 1997) was developed to annotate web pages with semantics, whereas the *Ontology Exchange Language* (XOL) (Karp et al. 1999) was primarily developed to exchange ontologies in the bioinformatics domain. Since the World Wide Web consortium has published several recommendations to express ontological content on the Web. For example, the *Resource Description Framework* (RDF) (Miller and Manola 2004) allows users to describe the relation between different web resources, while the *Web Ontology Language* (OWL) (van Harmelenand and McGuinness 2004) extends on the RDF vocabulary to provide precise meaning through formal semantics.

11.3.2 Ontology Engineering Methodologies

Over the last two decades, several ontology engineering methodologies have been developed. Gruninger and Fox (1995) propose a method inspired by knowledge-based development using first order logic, starting by identifying a number of motivating scenarios from which a number of natural language competency questions are extracted. These questions subsequently lead to the identification and formalisation of the terminology and axioms that constitute the ontology. Finally, the ontology is evaluated by proving that the original questions can be answered. In this way, competency questions are used to determine the scope and adequacy of the ontology. Uschold and King (1995) propose a method for building ontologies based on their experience with the Enterprise Ontology for system interoperation, while *Methontology* (Fernandez et al. 1997) builds on the main activities of software development and knowledge engineering methods, proposing an ontology

development life cycle based on evolving prototypes. *CommonKADS* (Schreiber et al. 1999) is a methodology for knowledge engineering in general, which is used to design and analyse knowledge-intensive, structured systems. A knowledge engineer such as a risk analyst in an enterprise, or a knowledge engineer in an ontological domain, can use it to detect the knowledge expansion, e.g. the opportunities based on the available knowledge resource, and the knowledge acquisition bottleneck.

Alternatively, the *Developing Ontology Grounded Methods and Applications* framework (DOGMA) is a formal ontology engineering framework inspired by various scientific disciplines, such as database semantics and natural language processing (De Leenheer et al. 2007). Although DOGMA partly draws on the best practice of the other methodologies, it differs from these approaches by providing a strict separation between the lexical representation of concepts and their relationships and the semantic constraints. This separation results in higher reuse possibilities and design scalability, and eases ontology engineering, as the complexity is divided and agreement can be more easily reached. Furthermore, the definition of terms in a natural language and the grouping of terms have been incorporated. By grounding knowledge in natural language, domain experts and knowledge engineers can use ordinary language constructs to communicate and capture knowledge. Therefore, domain experts do not have to tackle or learn to think in new paradigms, e.g. without the need to express their knowledge in RDF or OWL. Indeed, the complexity of just capturing knowledge is difficult enough already. Based on this approach, end users are able to represent the domain of discourse in terms they understand. Once the elicitation process is finished, and the ontology is formalised, the DOGMA tools can output the information to the requested paradigms. For example, simple linguistic structures like *lexons* can be transformed into *RDF triples,* which results in data (i.e. *facts*) being available as part of the *Linked Open Data* (LOD) project[133].

[133] http://linkeddata.org/

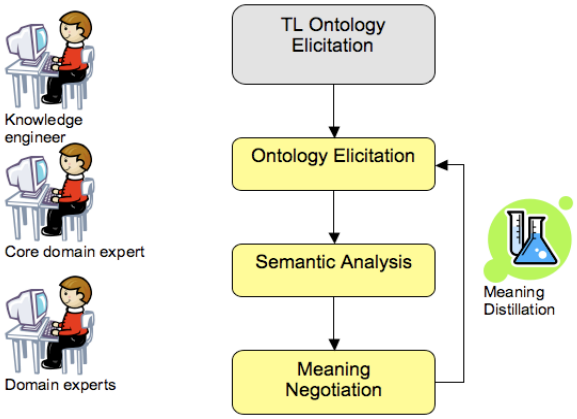

Fig. 11.4 DOGMA-MESS Iterative Process.

The *Meaning Evolution Support System* (DOGMA-MESS) is STARLab's methodology and tool to support community-driven ontology engineering (de Moor et al. 2006). The benefit of DOGMA-MESS is that it allows the domain experts themselves to capture meaning, while relevant commonalities and differences are identified, so that each iteration in the process results in a useable and accepted ontology. Hence, it provides an efficient, community-grounded methodology to address the issues of relevance. Figure 11.4 illustrates how a domain ontology is created through the interaction among three different types of stakeholders, namely the *domain expert*, the *core domain expert* and the *knowledge engineer*. The *domain expert* is a professional within the domain of discourse, while the *core domain expert* has a deep understanding of the domain across different organisations. The *knowledge engineer*, who has excellent expertise in representing and analysing formal semantics, is responsible to assist the domain experts and core domain experts in the processes of ontology creation, validation and evolution.

11.3.3 Application of Ontology Engineering in the Internet of Things

In this section we describe three ontology-based services that would enable three different areas of interoperability as needed for DiY application creation in the Internet of Things.

11.3.3.1 Knowledge Integration and Sharing

As the Web has changed from a mere repository of documents to a highly distributed platform where new types of resources can be discovered and even easily shared, one can extrapolate the Web as an Internet of Things making everyday objects addressable via IPv6 (Sundmaeker et al. 2010), as well as an *Internet of Services* making services easy to implement, consume, and trade.

However, the diversity of this increasing volume of data, services, and devices implies that it is impossible for them to work together, as many are designed independently, with particular, different application domains in mind. Making knowledge transparent to users and services thus requires the development of a *formal* and *precise* vocabulary that (i) defines concepts shared by a community and (ii) can be processed by machines.

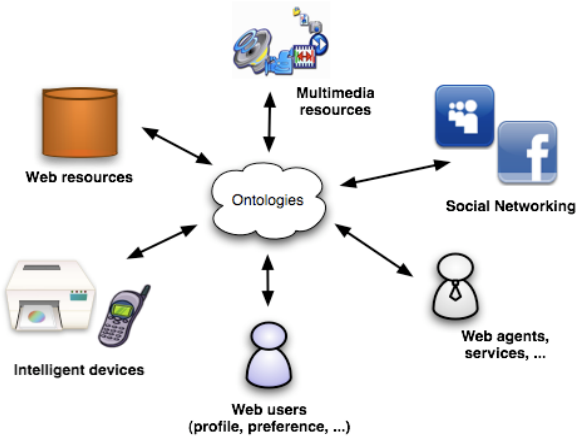

Fig. 11.5 Ontology-based Knowledge Integration and Sharing

Figure 11.5 shows how a semantic layer can be used to facilitate knowledge integration and sharing on the Web. For example, the annotation of devices with concepts from the FIPA *Device Ontology Specification*[134] would enable users to retrieve devices, like smartphones, based on their capabilities. Similarly, Eid et al. (2007) have developed an ontology to discover sensor data like GPS or temperature, consisting of three components:

- the *Suggested Upper Merged Ontology* (SUMO) (Niles and Pease 2001) is an upper level ontology developed by the IEEE to promote interoperability, information search and retrieval, automatic inference, and extendibility;
- the *Sensor Hierarchy Ontology* (SHO) includes models for data acquisition units and data processing and transmitting units; whereas

[134] http://www.fipa.org/specs/fipa00091/PC00091A.html

- the *Sensor Data Ontology* (SDO) describes the context of a sensor with regard to spatial and/or temporal observations.

Alternatively, the *Web Service Modelling Ontology* (WSMO)[135] provides a conceptualisation for the core elements of Web Services. For example, the web service component provides a vocabulary to describe the capabilities, interfaces, and internal working of a Web Service. In turn, this allows the discovery of services, their invocation (context based parameterisation and data transformation), and their mediation (e.g., the composition and orchestration of services).

11.3.3.2 Ontology-based Search

The DiYSE project aims to develop a framework to enable communities to create and exchange applications (i.e. software components) for ubiquitous computing and ambient intelligence, leveraging the Internet of Things. In practice, these applications are likely to come from a number of repositories and in a variety of formats. For example, software components could be indexed or tagged, based on the type of devices used to run the software. However, different repositories are likely to use different terminologies for the indexing.

This is where *ontology-based search* comes in as a solution, as in that approach the indexes are expressed in terms of an ontology which translates and hides the different repositories that have committed to it. The advantage of this method is that users can retrieve information based on *unambiguous terms*, thus enabling interoperability when *interpreting both queries and replies*. For example, a community may define their own ontology to define the capabilities of their software components. This ontology would then be used to annotate software components, thus enabling interoperability. The advantage is that a user may search for existing solutions *according to the community's vocabulary*. If the DiY user finds existing solutions, then he can either reuse the solution directly or extend the solution to solve other problems. Otherwise, the DiY user may submit his own annotated solution, which can then be retrieved by other members of the community.

In DiYSE, the ontology-based search needs to serve two types of users. *Technical users* are likely to use technical terms to define the capabilities of a hardware or software component, whereas *non-technical users* will use other non-technical terms that are meaningful to them. As a result, semantic 'translation' methodologies are researched that resolve the differences between technical and non-technical terminologies in order to get to a true *DiY ecosystem* on top of the Internet of Things.

[135] http://www.wsmo.org/

11.3.3.3 Context-aware Computing

A further challenge, which is typical for mobile distributed computing in general but becomes even more explicit in a sensor-rich environment, is to exploit the *dynamical changes* in the environment in applications, by means of those applications having the capability to *adapt to the context* in which they are running. Therefore, *context-aware computing* (Schilit et al. 1994) focuses on gathering information about *users*, like status, location, preferences and profile, next to *environment factors* such as lighting conditions, noise level, network connectivity, nearby things and even social aspects. This information is then used to adjust the behaviour of an application to suit user needs and preferences.

As is done in the DiYSE framework, context management can be supported semantically by two core elements, namely the *contextual ontology* and the *context model*. The *contextual ontology* provides a conceptualisation of the characteristics of *real world objects*, while the *context model* provides access to the *contextual knowledge*. For example, the *iHAP* ontology (Machuca et al. 2005) provides a vocabulary to represent (i) spatial description, (ii) actor description, (iii) context features description, (iv) service description, and (v) device description in smart environments like vehicles, homes or public buildings. So, based on for example this ontology, agents and users are able to interoperate to provide context-aware services in the dynamically changing smart environments.

In short summary of this section, we have discussed three areas in which ontology engineering methodologies are important for the Internet of Things, and for DiY creation on top of it in particular, as researched within the scope of the DiYSE project, namely *knowledge integration and sharing*, *ontology-based search*, and *context-aware computing*. Essentially, ontologies are a means to the agreements made among a community and are intrinsically community-based, and so form an enabling step needed for effective sharing and creation activities among DiY communities. Even when a software agent 'commits' to such ontology by using the same vocabulary in a consistent manner, it shares the same knowledge as the agents designed by others in the same community. So, this kind of shareability, as originating from the community, also enables the agents to seamlessly interoperate with each other. In other words, the fundamental principle of ontology engineering is 'autonomy' (Meersman 2010), granting many engineering advantages to the application builders and professionals, up to occasional DiY users.

Furthermore, many other generally applicable ontology engineering techniques can interestingly be leveraged in the Internet of Things, like for instance around *modelling* (Spyns et al. 2002; Baglioni et al. 2008), *querying* (Loiseau et al. 2006), *reasoning* (Baglioni et al. 2008), *annotating* (Kim and Park 2005) and *matching* (Tang et al. 2010).

11.4 The DiYSE Service Framework

On top of the network of connected sensor and actuator hardware, uniformly abstracted via sensor abstraction middleware and semantic annotation, the DiYSE service framework provides a *number of service-level functions*, in turn needed to support the application creation layer above it, in which professional developers up to non-technical end users can shape the smart space by collaboratively creating and deploying Internet of Things applications. Next to the service functionality for *composition, deployment and execution*, this in particular entails also functionality to *adapt and personalise* applications, as well as the *creation* thereof, to context of use, respectively creation.

Figure 11.6 positions the DiYSE service framework in the high-level overview of the overall DiYSE architecture. Mediating between all the identified actors and application areas versus the underlying Internet of Things technologies, three main functional areas for the framework can be distinguished. We discuss each of them in the next sections.

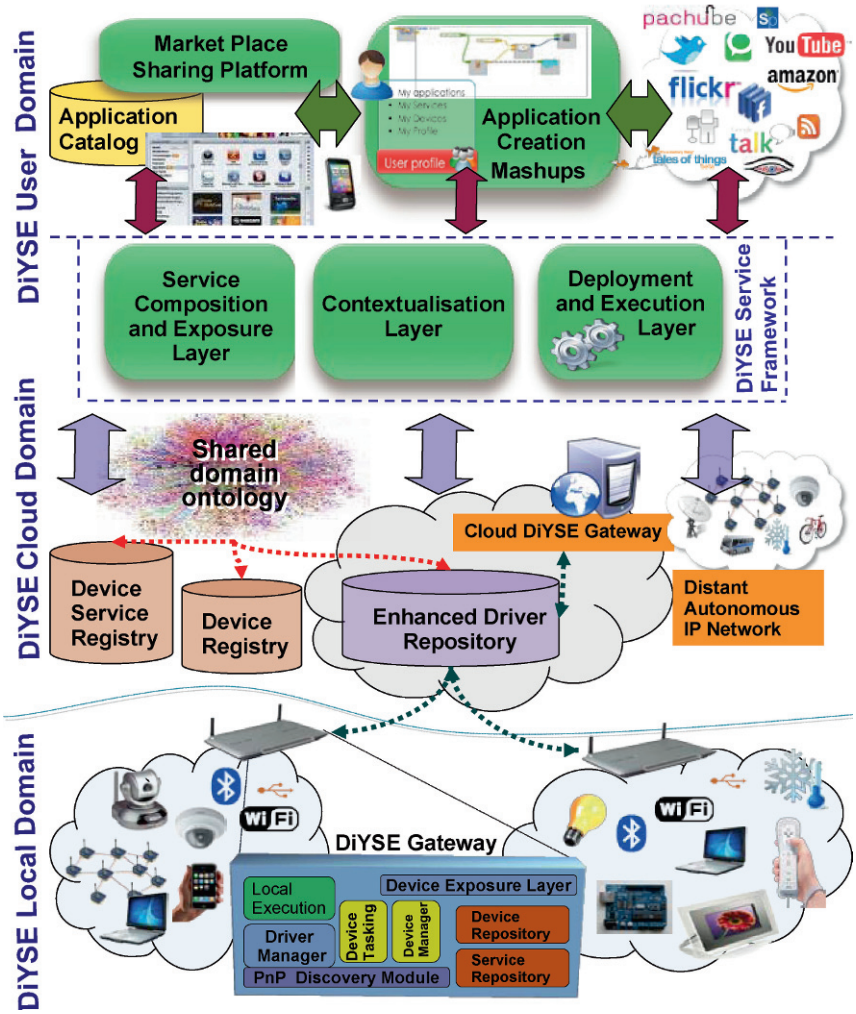

Fig. 11.6 Position of the Service Framework in the DiYSE Overall Architecture View

11.4.1 Contextualisation Layer

Under the *Contextualisation Layer* for DiYSE we group the components for contextualisation and personalisation, serving the application and application creation environment on top of it. In particular, we distinguish three, highly interrelated functions in it:

- **User profiling and personalisation:** A *user profile* is a structured data record containing user-related information like identifiers, characteristics, abilities, needs and interests, preferences, behavioural history and extrapolations thereof for predicting and anticipating future behaviour. It can therefore be exploited to provide personalised, user-context-aware service recommendation, leveraging related user profiles from the crowd, and context-awareness during eventual service use.
- **Modelling of the physical context information:** The environmental context is also a very relevant feature in service oriented environments, particularly in 'smart' environments, where services are expected to behave intelligently, learning from and anticipate on what happens in the surroundings. In general, the establishment of an effective *context model* is essential for designing context-aware services. Strang and Linnhoff-Popien (2004) provide a survey of the most important context modelling approaches for pervasive computing, such as *key-value* models, *mark-up scheme* models, *graphical* models, *object oriented* models, *logic-based* models and *ontology-based* models. As discussed in previous sections, DiYSE has selected an *ontology-based* model for representing the context. Ontologies have the important benefit of providing a uniform way for specifying the model's core *concepts* as well as an arbitrary amount of *sub-concepts* and *facts*, altogether enabling contextual *knowledge sharing* and reuse in a ubiquitous computing system.
- **Reasoning:** Another key issue in the study of DiY applications is the *reasoning* about environmental context and user information, allowing for deduction of *new* knowledge in addition of the directly detected information. As the ultimate goal is to make the services and the surrounding *smart*, i.e., more closely in accordance with the specific user expectations, a fundamental challenge exists in deriving *correct and stable* conclusions from the typically *imperfect* context data acquisition in the highly dynamic and heterogeneous ambient environment.

11.4.2 Service Composition and Exposition Layer

The *Service Composition and Exposition Layer* in DiYSE groups the functions that enable the upper user-facing tools to list and access the different available services and service parts as provided by any actor of the DiY community, i.e., third parties and professionals as well as any users.

It comprises the following functions:

- **Service exposure:** This function provides a unified access to the services and components made available by different levels of users, professionals and third parties (Blum et al. 2008), which is essential for the envisioned DiYSE creation process. It thus enables the different types of users to discover, compose and

publish at a properly abstracted service-level. Besides that, functionality such as instantiation and the related exception handling, authentication and authorisation, layered functional exposure, configuration and service user interface representation in the DiYSE creation process is envisioned.

- **Semantic engine:** The semantic engine function provides the service exposure function with the abstractions to semantically mediate interaction of devices, services and actors, according to the methods discussed in the sections on *semantic interoperability*, section 11.3, leveraging a set of *shared ontology repositories* for that purpose.
- **Orchestrator-compositor:** As a key part in the DiYSE creation process, the dynamic composition and orchestration of hybrid and composite services is needed, leveraging the semantic engine as well as the Contextualisation and Personalisation Layer, and closely interacting with the service exposure function, registering also newly composed applications (ESI 2008).

11.4.3 Execution Layer

The main objective of the DiYSE *Execution Layer* is to execute the composed, distributed applications in a dynamic, context-aware manner.

One of the main challenges in this layer is to establish a mechanism for managing dependencies on all the context data at runtime, ranging from user profiles and user context up to the various aspects of the environment's context, sensor data streams as well as events of new devices appearing or disappearing in the environment. Moreover, there is a *tight relation to the devices in the environment* because of the tangible interaction envisioned in DiYSE, as discussed in later sections, requiring device-level mediation mechanisms at lower layers, as previously discussed.

Solutions that have been proposed for execution at the end of a creation process include the use of *software variability* for defining those parts in a workflow that may vary at runtime (Bastida et al. 2008). Before the workflow is executed, its variable parts are instantiated according to the relevant contextual parameters.

A further aspect to be considered for the Execution Layer is the potential semantic binding of *running* component instances, and the *dynamic* adaptation of these bindings when the environment changes.

11.4.4 DiYSE Application Creation and Deployment

As the main objective of the DiYSE project is to eventually enable non-technical users to create their own Internet of Things applications based on available de-

vices and service parts, leveraging the environmental and user context, we have stipulated the overall phases in the DiYSE creation process as follows:

- **Installation of sensors, devices or actuators:** The main challenge in this phase is the dynamic and correct registration of all required device information in a device registry, a driver registry and the service registry, while providing the user with a highly intuitive procedure, ideally not requiring any 'unnatural', i.e. seemingly unneeded interventions. Support at the hardware and network level for this in DiYSE was discussed in section 11.2.
- **User design of the application:** This phase is where the user creates, configures or composes a (partially) new application, taking into account the capabilities (expected to be) available in the environment, according to his/her own profile and context. The challenge here clearly is to provide the user with the right kind of tools at the *right level of abstraction*, according to his/her expertise level. Also, validation or simulation of device interactions and device data as part of the design is an important aspect. This is partly related to the discussions in section 11.3 on *semantics*, but is also closely combined with the *interaction* as discussed in section 11.5.
- **Production of the application runtime code:** After designing the new application, its runtime code can be factored, among other processes using variability techniques and semantic mapping, as related to what was previously under section 11.4.
- **Deployment of the application:** The run time application code is eventually deployed in a consecutive phase, possibly in a distributed or mobile manner, according to dynamic device and network resource conditions and other context factors, as applicable at that moment in time, and adapting to environmental context changes.
- **Execution of the application:** After deployment, the regular execution life cycle phases comes in action, effectively starting or stopping dependable subservices for the application, for the assumed context and user reach.

Finally, users start interacting with the newly created application.

11.5 Interactions, Using and Creating in Smart Spaces

The smart space consists of interactive components, sensors and actuators and allows for very versatile interaction with the services given shape by the tangible, distributed interface. One could say that the smart space ideally forms an ecology, in which people seamlessly interact with the environment to achieve specific goals, in particular also the creation goal which the previously discussed service framework is aiming to support at the software level.

11.5.1 Service Interaction and Environment Configuration

The interaction will be related to *using* the services provided by or through the environment, or to *configuring* the environment itself. The latter task receives special attention in the DiYSE project, as we want to enable the DiY end users to perform this task in most cases. This requires the interaction to be very intuitive and 'programming'-like solutions are out of the question. Configuring the environment, or defining the 'intelligence' of the environment, consists for instance of:

- associating input *events*, like a button press, a sensor reading change, or GUI widgets, to *actions* in the environment, like motors, valves or other actuators, or application settings, via a set of *behaviour rules*,
- defining dependencies on *context* information, like presence detectors, time of day, or temperature,
- personalising services, like *look and feel* adaptation to the user's identity, taking into account preferences for content, adapting to use patterns, switching to the preferred input or output modality, or
- creating *mash-ups* of existing controls, defining a "macro" of control operations specifying a personal *remote* interaction to services.

In this project we envision the use of *physical browsing* techniques to help selecting the physical target objects for use in the configuration by means of touching or pointing actions. The project is also researching the use of templates, wizards or *'define by doing'* approaches to simplify the configuration. The *'define by doing'* approach requires the environment to be completely observable. The user will define complex functionality by creating the specified circumstances and performing a series of actions that *show* the system what is expected as application behaviour in those circumstances.

11.5.2 Ecological Design Approach

Refining beyond the essentially different phases as discussed in the previous sections, the design of a smart space ecology does not happen in one step, but consists of the design of the components (i.e., devices, sensors, actuators, single device applications) for use in the environment, *enabling design*, and the design of the functional environment, *local design*. While the *enabling design* will typically be performed by professionals as part of their product development, *local design* can be performed also by the end user, tailoring his environment and combining the functionality of the enabled products, as exemplified by the configuration task in the previous paragraph. We coin this to be the *Ecological Approach to Smart Environments* (EASE). Note that the approach emphasises the importance of *in-*

volving the users at all phases of the design (Keinonen 2007; Norros and Salo 2009).

11.5.3 Architectural Support and Modelling for Interaction

To get the various interactive components in the environment to work together to provide such a context-aware personalised interactive experience is not a trivial task. Also from the user perspective, the interaction capabilities of the components must be described properly and be advertised, dependencies on context information must be specified, and, as indicated before in the discussions on context, the user's preferences and abilities must be taken into account.

The architecture as described in the previous sections provides a good base for this interactive environment. Interactive component capabilities can be described using the same ontology-based semantic methods when applying a suitable interaction ontology. Interactive events can be channelled through the available interoperability solutions. Context information is provided at a suitable level by the brokers in the system.

The interactive applications themselves need to be modelled in such a way, that their interactions are easy to map to the components available in the environment. This requires new solutions. Most user interfaces are designed for a specific platform, even a specific device, and cannot be transferred to other platforms, let alone a set of interactive components. Remote interfaces, using *HTML Forms* for example, have partly solved this problem, as the platform rendering the user interface may be different from the one running the application. This approach has also allowed for scaling the user interface to devices with various viewports. The method is sometimes referred to as a *multi-channel* approach. It works well with '*window, icon, menu, pointing device*' (WIMP) devices and has successfully been mapped to mobile devices as well. Mapping for multi-modal interaction or distributing the interaction over a multi-device solution requires more versatile modelling of the user interface. A starting point for this kind of modelling can be found in the *Abstract UI* solutions found in UsiXML (Limbourg et al. 2005), Teresa (Paternò 1999) and the like. But unlike those modelling solutions, the DiYSE system must moreover be able to resolve the mapping issues *at run time*, in the *changing* environment. This is one of the central themes of research in the project.

11.5.4 Example Personalised Interaction Method: Smart Companion Devices

11.5.4.1 Multimodal Mood Detection in Smart Companions

In contrast to the multi-channel, spatially distributed user interaction discussed as a critically needed paradigm leveraging the possibly 'thin' nature of sensors and actuators as an intuitive, natural user (creation) interface in the Internet of Things, another asymptote of rich, intuitive user interaction is the one of a single- or multi-object, 'thick' *smart object* paradigm, offering and embodying a human-like counterpart for the user-creator. In the DiYSE project, this is seen as an advancement beyond classical multimedia interfaces, which ads up to the 'things' available in the smart space for use and DiY creation.

In particular, for *smart companions*, being robotic pets whose appearance and behaviour are tailored to human interaction, a comfortable user experience requires the establishment of a meaningful robot behaviour illusion. This can be achieved employing a variety of techniques, aimed at the recognition of auditory and visual cues, such as speech/speaker and face/gesture recognition, of which the most advanced variant is the *multimodal* approach, combining voice, image and gesture recognition to derive context. So, context data available through the smart companion device can be exposed in the DiYSE environment for use by other services and applications and vice versa, forming a rich connection to the interacting user.

In the current state of the art, smart companions lack the ability to detect what is arguably the most important factor present in normal human interaction: the *mood of the speaker*. While speech and non-verbal analysis methods can be extended to detect mood or emotions, also such *affect recognition* can be further enhanced by associating image analysis to it for face and gesture recognition. These affect detection techniques have been the object of extensive research, but the associated computational cost has generally kept their application restricted to relatively powerful computing platforms, restricting the interaction illusion to being it 'via' the computer.

The aim of the smart companion work in DiYSE is therefore entailing two steps: (i) to research and implement affect detection algorithms on a PC platform, and (ii) to migrate them to an embedded platform present in a state-of-the-art smart companion robot. A standard back-end software interface is also foreseen for tying into the DiYSE context-awareness functions, integrating user mood as well as adapted companion behaviour as enrichments of the DiYSE smart user interaction.

11.5.4.2 Embedded Systems for Autonomous Smart Companions

So, as indicated, the enhancements proposed for the smart companion require a sufficiently compact, computationally powerful, and relatively low cost hardware platform. In fact, while serving a different purpose, such hardware requirements are of a similar nature as those needed for DiYSE Gateways needed to connect the 'thin' sensor and actuator nodes. Indeed, fortunately, nowadays available typical DiY electronics boards could be selected[136] as appropriate for this purpose too, namely *Beagle Board*[137] and *Gumstix Overo*[138].

11.5.4.3 Affect Recognition in DiYSE

By not taking into account the affective state of the user, the traditional *Human-Computer Interaction* systems are often perceived as cold and unnatural when compared to human-to-human communications. In the past decade, advances have been achieved toward the collection of large databases of affective displays, as well as toward the analysis of human behaviours by means of *audio-based*, *video-based*, and *audiovisual* methods[139] (Zeng et al. 2009).

A prerequisite in designing automatic affect recognition systems is the availability of *databases containing labelled data* of human affective expressions. Since manual labelling of emotional expressions is time consuming, subjective, error prone, and expensive, many databases consist of 'artificially' acted emotions, but also recordings of real, spontaneous affective behaviour were collected from human interviews, phone conversations, meetings, computer-based dialogue sys-

---

[136] The most popular embedded systems are built around the *ARM* architecture. Lately, Intel has introduced the *Atom* processor to cater for the same range of applications. These considerations narrowed our choices to (i) *Texas Instruments OMAP* based single-board computers and (ii) *Intel Atom* based system, presenting a power consumption versus code portability trade-off. From that, *OMAP* (v3), supporting *WindowsCE*, *Symbian*, *Android* and *Linux*, was eventually selected, leading to Beagle-Board and Gumstix Overo as the preferred platforms for the smart companion.

[137] http://beagleboard.org/

[138] http://www.gumstix.net/Overo/

[139] Most *audio-based* systems are trained and tested on acted speech in order to recognize prototypical emotions. Beside the selection of the classifier, another issue concerns the optimal feature set among linguistic and paralinguistic descriptors, as well as the reliable extraction of these cues (e.g. pitch-related prosodic features). *Vision-based* affect recognition studies mainly focus on facial expression analysis by means of pattern recognition approaches. The best choice for designing automatic recognizers seems to be the combined use of both geometric and appearance features. However, an important challenge remains the robustness to arbitrary head movement, occlusions, and scene complexity. Finally, while the vast majority of the *audiovisual-based* systems implement a decision-level fusion strategy and some other studies focus on the feature-level fusion approach to recognize coarse affective states (e.g., positive, negative, or neutral), the model-level fusion methods have the advantage of making use of the correlation between audio and video data streams without the requirement of perfect synchronization of these streams.

tems, etc.. While the automatic tool *Feeltrace* (Cowie et al. 2000) was developed for labelling such emotional expressions, the development of semi-supervised labelling methods remains an open issue.

As a first implementation of affect recognition in DiYSE, we chose to use the *EmoVoice* suite (Vogt et al. 2008) in combination with a voice recognition algorithms designed by UMons/Multitel. *EmoVoice* is in fact intended to be used by *non-experts*, opening further possibilities for DiY community scenarios where DiY creators can directly improve the affect recognition for an envisioned application purpose.

11.5.5 Multimodal Middleware Protocol

Multimodal approaches combining voice, image, and gesture recognition must necessarily acquire data from a *variety* of devices. The dedicated *Multimodal Middleware Protocol* (MMP) provides the low level architecture to glue different device modality components in a single user interface network. MMP's goal is to compose this network, abstracting details like underlying network protocols and the meaning of custom messages, so that all higher layer semantics and logic can relate to the composite multimodal interface. In the DIYSE concept the level above the MMP is a powerful context reasoning system, providing context-aware computing features, gathering information about users and their environment to adjust the behaviour of applications. Through the natural interfaces provided by multimodal devices such as the smart companion, context is seamlessly extended to social expressivity.

MMP interconnects devices and can store their capabilities in a central point, called a *Multimodal Hub* (MMH). Once a device modality component connects to the MMH, the MMH stores the user interfacing capabilities in terms of production and consumption of human communication events as sent by the component, and then manages the connections between components based on default or user-configured rules.

11.5.6 The Ultimate Example: Simple Smart Space Interaction with Multi-device Interfaces

Beyond the smart companion view, more heterogeneous scenarios are thus ultimately envisioned in DiYSE. Here is an illustrative example:

Peter arrives at home listening to his favourite MP3 music after an average day of work. The lights in the hallway turn on automatically as he enters and when he enters the kitchen to start making dinner the music is automatically transferred

to the kitchen audio system so that he can remove his ear plugs and have his hands free. While preparing the dinner, his wife Katie arrives. She tells him with enthusiasm about the inspiring events she experiences at a work trip. She touches the screen in the kitchen with her mobile phone, which contains the pictures she has taken during her work trip. The screen comes alive and displays an overview of the pictures taken during the day. The touch screen of her phone simultaneously changes for use as a touch pad to control the cursor on the screen. She navigates to the first picture of interest and says 'Start slide show'. The screen starts to display the slideshow. When a video patch appears in the middle of the slide show, Peter's music fades out and they hear the audio track of the video. When a particularly beautiful picture comes up, Peter "steals" the picture by touching the screen with his mobile. The light slightly disturbs their viewing and Peter points at the light in the kitchen with his mobile and a personalised service view pops up. He selects a dimmed atmosphere by tapping his mobile a few times…

The implementation of scenarios like this requires the tight cooperation of all the available devices in the environment. The simultaneous use of interactive features of existing devices to operate new services constitutes to the multi-device interaction experience. Next step refinements of the DiYSE architecture will consider these aspects to yet a more complete extent.

11.6 Conclusion - Future Work of the Consortium

In this chapter we have sketched the wide variety of aspects tackled in the ongoing endeavour of enabling mass creativity in the Internet of Things, as envisioned by the DiYSE project.

As the main conclusion of the work done in the project until now, it is clear that a number of infrastructural measures and creation-supporting functions need to be in place to realise DiY application creation in the Internet of Things.

With enhanced, semantically annotated device drivers potentially auto-provisioned in a DiYSE Gateway function, a middleware for proper distributed execution across sensor network nodes, and a service framework that exposes context-awareness enabling functions and composable service building blocks towards the creation environment, a first basis for enabling such DiY application creation in the Internet of Things has been defined.

In order to realise the ultimate goal of DiY *smart spaces*, intuitively shapeable by non-technical actors, further creation-related enablers are still needed, both at the level of back-end services and tools, as well as support for *sharing* DiY experiences across large communities.

At the time of writing of this chapter, the consortium is progressing the detailed work on the DiYSE architecture according to the elements discussed, is implementing first prototypes, and is conducting interaction (co-)design user research,

further fine-tuning towards the full enablement of communities sharing DiY smart space applications and smart objects.

Acknowledgements

This work is supported by the ITEA2 Eureka cluster, and the respective national funding authorities of the project partners, under the European ITEA2 project 08005, DiY Smart Experiences (DiYSE), conducted by 40 partners from 7 European countries. Beyond this book chapter, more information about the project can be found at the project's public website http://www.dyse.org.

References

Baglioni M, Macedo J, Renso C, Wachowicz M (2008) An Ontology-Based Approach for the Semantic Modelling and Reasoning on Trajectories. In: Song I-Y et al. (eds) Advances in Conceptual Modeling – Challenges and Opportunities, Springer, Berlin Heidelberg

Bastida L, Nieto FJ, Tola R (2008) Context-Aware Service Composition: A Methodology and a Case Study. SDSOA 2008 Worshop, ICSE Conference Proceedings, Leipzig, Alemania

Blum N, Dutkowski S, Magedanz T (2008) InSeRt, SEW, 32nd Annual IEEE Software Engineering Workshop

Cowie R, Douglas-Cowie E, Savvidou S, McMahon E, Sawey M, Schöder M (2000) 'Feeltrace': An Instrument for Recording Perceived Emotion in Real Time. Proc. ISCA Workshop Speech and Emotion

De Leenheer P, de Moor A, Meersman R (2007) Context Dependency Management in Ontology Engineering: a Formal Approach. J Data Semant 8:26-56

de Moor A, De Leenheer P, Meersman R (2006) DOGMA-MESS: A meaning evolution support system for inter-organizational ontology engineering. Proceedings of the 14th International Conference on Conceptual Structures (ICCS 2006)

Dormer P (1997) The Culture of Craft. Manchester University Press, Manchester, UK

Eid M, Liscano R, El Saddik A (2007) A Universal Ontology for Sensor Networks Data. Proc. IEEE International Conference on Computational Intelligence for Measurement Systems and Applications (CIMSA 2007)

ESI (2008) A3.D20. The Approach to Support Dynamic Composition. SeCSE Deliverable

Fernandez M, Gomez-Perez A, Juristo N (1997) Methontology: From Ontological Art towards Ontological Engineering. Proc. AAAI97 Spring Symposium Series on Ontological Engineering

GeSI, The Climate Group (2008) SMART 2020: Enabling the low carbon economy in the information age. Creative Commons. http://www.smart2020.org/_assets/files/02_Smart2020Report.pdf. Accessed 25 September 2010

Gruninger M, Fox M (1995) Methodology for the Design and Evaluation of Ontologies. Proc. of the Workshop on Basic Ontological Issues in Knowledge Sharing

Guinard D, Trifa V, Pham T, Liechti O (2009) Towards Physical Mashups in the Web of Things. Proceedings of INSS 2009 (IEEE Sixth International Conference on Networked Sensing Systems), Pittsburgh, USA. http://www.vs.inf.ethz.ch/publ/papers/guinardSensorMashups09.pdf. Accessed 25 September 2010

IFTF (2008) The Future of Making. http://www.iftf.org/system/files/deliverables/SR-1154%20TH%202008%20Maker%20Map.pdf. Accessed 25 September 2010

ITU-T (2008) Intelligent transport systems and CLAM. ITU-T Technology Watch Report #1

Karp P, Chaudri V, Thomere J (1999) XOL: An XML-Based Ontology Exchange Language. SRI International. http://www.ai.sri.com/pkarp/xol/xol.html. Accessed 25 September 2010

Keinonen T (2007) Immediate, product and remote design. International Association of Societies of Design and Research, Honkong

Kim J-J, Park JC (2005) Annotation of Gene Products in the Literature with Gene Ontology Terms Using Syntactic Dependencies. IJCNLP 2004. Lect Notes Comput Sci 3248:787-796

Limbourg Q, Vanderdonckt J, Michotte B, Bouillon L, López-Jaquero V (2005) USIXML: A Language Supporting Multipath Development of User Interfaces. In: Bastide R, Palanque P, Roth J (eds) Engineering Human Computer Interaction and Interactive Systems. Springer, Berlin, Heidelberg

Loiseau Y, Boughanem M, Prade H (2006) Evaluation of Term-based Queries using Possibilistic Ontologies. In: Herrera-Viedma E, Pasi G, Crestani F (eds) Soft Computing in Web Information Retrieval: Models and Applications. Springer

Luke S, Spector L, Rager D, Hendler J (1997) Ontology-based Web Agents. Proc. International Conference on Autonomous Agents (Agents97)

Machuca M, Lopez M, Marsa Maestre I, Velasco J (2005) A Contextual Ontology to Provide Location-aware Services and Interfaces in Smart Environments. Proc. IADIS International Conference on WWW/Internet

Meersman R (2010) Hybrid Ontologies in a Tri-Sortal Internet of Humans, Systems and Enterprises. Keynote talk, InterOntology'10 Conference, KEIO Tokyo

Miller E, Manola F (2004) RDF primer: W3C recommendation. World Wide Web Consortium. http://www.w3.org/TR/2004/REC-rdf-primer-20040210/. Accessed 25 September 2010

Mogensen K (2004) Creative Man. The Copenhagen Institute for Futures Studies, Denmark

Niles I, Pease A (2001) Towards a Standard Upper Ontology. Proc. 2nd International Conference on Formal Ontology in Information Systems (FOIS-2001)

Norros L, Salo L (2009) Design of joint systems - a theoretical challenge for cognitive systems engineering. Cogn Technol Work 11:43-56

Paternò F (1999) Model-based design and evaluation of interactive applications. Springer, London

Schilit B, Adams N, Want R (1994) Context-aware computing applications. Proc. IEEE Workshop on Mobile Computing Systems and Applications (WMCSA'94)

Schreiber G, Akkermans H, Anjewierden A, de Hoog R, Shabolt N, Van de Velde W, Wielenga B (1999) Knowledge Engineering and Management: The CommonKADS Methodology. MIT Press

Sennet R (2008) The Craftsman. Yale University Press, New Haven, CT

Shove E, Watson M, Hand M, Ingram J (2007) The Design of Everyday Life. Berg, London, UK

Smith B (2003) Ontology: An Introduction. In: Floridi L (ed) Blackwell Guide to the Philosophy of Computing and Information. Blackwell

Spyns P, Meersman R, Jarrar M (2002) Data modelling versus Ontology engineering. SIGMOD Rec 31:12-17

Sterling B (2005) Shaping things. MIT Press, Cambridge, MA

Strang T, Linnhoff-Popien C (2004) A context modeling survey. First International Workshop on Advanced Context Modelling, Reasoning and Management, Nottingham, England

Studer R, Benjamin R, Fensel D (1998) Knowledge Engineering: Principle and Methods. Data & Knowl Eng 25:161-197

Sundmaeker H, Guillemin P, Friess P, Woelfflé S (eds) (2010) Vision and Challenges for Realising the Internet of Things. CERP-IoT. http://www.internet-of-things-research.eu/pdf/IoT_Clusterbook_March_2010.pdf. Accessed 25 September 2010

Tang Y, Zhao G, De Baer P, Meersman R (2010) Towards Freely and Correctly Adjusted Dijkstra's Algorithm with Semantic Decision Tables for Ontology Based Data Matching. In: Mahadevan V, Zhou J (eds) Proc. of the 2nd International Conference on Computer and Automation Engineering "ICCAE 2010". IEEE, Suntec city, Singapore

Uschold M, King M (1995) Towards a methodology for building ontologies. Proc. of the Workshop on Basic Ontological Issues in Knowledge Sharing

van Harmelenand F, McGuinness D (2004) OWL web ontology language overview: W3C recommendation. World Wide Web Consortium. http://www.w3.org/TR/2004/REC-owl-features-20040210/. Accessed 25 September 2010

Vogt T, André E, Bee N (2008) EmoVoice – A Framework for Online Recognition of Emotions from Voice. Proc. Workshop on Perception and Interactive Technologies for Speech-Based Systems

Von Hippel E (2005) Democratizing Innovation. MIT Press, Cambridge, MA

Zeng Z, Pantic M, Roisman GI, Huang TS (2009) A Survey of Affect Recognition Methods: Audio, Visual, and Spontaneous Expressions. IEEE Trans Pattern Anal Mach Intell 31:39-58