An Architectural Blueprint for a Real-World Internet

Alex Gluhak[1], Manfred Hauswirth[2], Srdjan Krco[3], Nenad Stojanovic[4], Martin Bauer[5], Rasmus Nielsen[6], Stephan Haller[5], Neeli Prasad[6], Vinny Reynolds[2], and Oscar Corcho[8]

[1] University of Surrey, UK
[2] National University of Galway, Ireland
[3] Ericsson, Serbia
[4] FZI, Germany
[5] NEC, Germany
[6] Aalborg University, Denmark
[7] SAP, Switzerland
[8] Universidad Politécnica de Madrid, Spain

Abstract. Numerous projects in the area of Real-World Internet (RWI), Internet of Things (IoT), and Internet Connected Objects have proposed architectures for the systems they develop. All of these systems are faced with very similar problems in their architecture and design and interoperability among these systems is limited. To address these issues and to speed up development and deployment while at the same time reduce development and maintenance costs, reference architectures are an appropriate tool. As reference architectures require agreement among all stakeholders, they are usually developed in an incremental process. This paper presents the current status of our work on a reference architecture for the RWI as an architectural blueprint.

Keywords: Real-World Internet, Internet of Things, Internet Connected Objects, Architecture

1 Introduction

Devices and technologies ubiquitously deployed at the edges of the networks will provide an infrastructure that enables augmentation of the physical world and interaction with it, without the need for direct human intervention, thus creating the essential foundations for the Real-World Internet (RWI).

Leveraging the collective effort of several projects over the last number of years [SENSEI, ASPIRE, IOT-A, PECES, CONET, SPITFIRE, SemsorGrid4Env], this chapter presents the current status of the work aimed at definition of an RWI reference architecture. The core contribution of this paper is the distillation of an initial model for RWI based on an analysis of these state of art architectures and an understanding of the challenges. This is achieved by:

- An identification of a core set of functions and underlying information models, operations and interactions that these architecture have in common.
- A discussion on how these architectures realize the above identified functions and models and what features they provide.

J. Domingue et al. (Eds.): Future Internet Assembly, LNCS 6656, pp. 67–80, 2011.

2 The Real World Internet

Since the introduction of the terminology over a decade ago, the "Internet of Things (IoT)" has undergone an evolution of the underlying concepts as more and more relevant technologies are maturing. The initial vision was of a world in which all physical objects are tagged by Radio Frequency Identification (RFID) transponders in order to be uniquely identified by information systems. However, the concept has grown into multiple dimensions, encompassing sensor networks able to provide real world intelligence or the goal-oriented autonomous collaboration of distributed objects via local wireless networks or global interconnections such as the Internet.

Kevin Ashton, former Director of the Auto-ID Center, once famously formulated: "Adding radiofrequency identification and other sensors to everyday objects will create an Internet of Things, and lay the foundations of a new age of machine perception".

We believe that machine perception of the real world is still at the heart of the Internet of Things, no matter what new technologies have meanwhile become available to enable it. As such, one of the key roles of the Internet of Things is to bridge the physical world and its representation in the digital world of information systems, enabling what we refer to in part of the Future Internet Assembly (FIA) community as the so called Real World Internet (RWI).

The RWI is the part of a Future Internet that builds upon the resources provided by the devices [HAL] of the Internet of Things, offering real world information and interaction capabilities to machines, software artifacts and humans connected to it.

The RWI assumes that the information flow to and from IoT devices is taking place via local wired and wireless communication links between devices in their proximity and/or through global interconnections in the form of the current Internet and mobile networks or future fixed and mobile network infrastructures.

One important property of the RWI which distinguishes it from the current Internet is its heterogeneity, both regarding the types of devices as well as communication protocols used. IPv6 and in particular 6LoWPAN play an important role, but other proprietary wireless protocols will see continued use as well. To deal with this heterogeneity, services – in the form of standard Web Services and DPWS[1], but more likely using RESTful approaches and application protocols like CoAP – provide a useful abstraction. As services play a pivotal role in the Future Internet Architecture, the use of services for integrating the RWI also fits well into the overall architectural picture. One has to keep in mind though that RWI services have some different properties from common, enterprise-level services: They are of lower granularity, e.g., just providing simple sensor readings and, more importantly, they are inherently unreliable; such RWI services may suddenly fail and the data they deliver has to be associated with some quality of information parameters before further processing.

[1] Device Profile for Web Services

3 Reference Architecture

In this section we present an initial model on which several of the current RWI architecture approaches are based. While not as comprehensive as a reference architecture, it already identifies the major underlying system assumptions and architectural artifacts of the current RWI approaches. The model has been developed through a careful analysis of the existing RWI architectures according to the following dimensions:

1. Underlying system assumptions,
2. functional coverage of the services provided by the architectures,
3. underlying information models in the architectures, and
4. operations and interactions supported in these architectures.

3.1 Underlying RWI Architecture Assumptions

Common to all RWI architectures is the underlying view of the world, which is divided into a real and a digital world as depicted in Fig. 1. The real world consists of the physical environment that is instrumented with machine readable identification tags, sensors, actuators and processing elements organized in domain specific islands in order to monitor and interact with the physical entities that we are interested in. The digital world consists of:

a) Resources which are representations of the instruments – Resource level,
b) Entities of Interest (EoI) which are representations of people, places and things – Entity level, and
c) Resource Users which represent the physical people or application software that intends to interact with Resources and EoI.

Providing the services and corresponding underlying information models to bridge the physical and the digital world by allowing users/applications to interact with the Resources and EoI is the main contribution of the RWI reference architecture towards a RWI. Typically, RWI architectures provide two abstraction levels for such interactions: resource level and entity level.

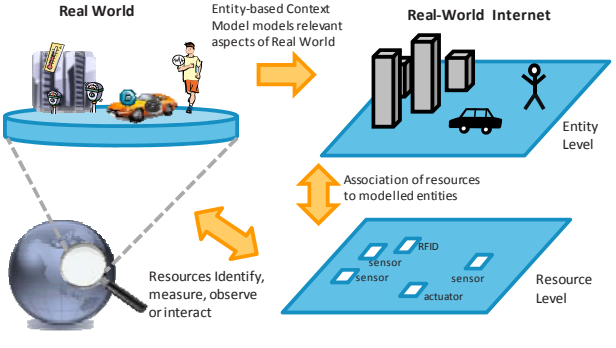

Fig. 1. World-view of RWI systems

On the resource level, resource users directly interact with resources. Such interactions are suitable for certain types of RWI applications where the provided information or interaction does not need any context (e.g., an understanding of how information is related to a real-world entity).

On the entity level, some RWI architectures offer the option to applications to use an inherent context model which is centered around the EoI. For these EoIs, relevant aspects like the activity of a person or the current location of a car are modeled as context attributes. Applications can base their requests on EoI and context attributes. The underlying requirement is that the resources providing information are associated with the respective entities and attributes, so that the services offered by the RWI architectures can find the required resources for the entity-level requests. Therefore, architectural components exist that enable contextualized information retrieval and interaction, as well as dynamic service composition.

Besides the above assumptions, various architectures take also socio-economic aspects into consideration, as they consider various actors in one or more business roles within the context of the RWI eco-system created around their architecture, forming the so-called RWI communities. The main roles in these communities are:

1. Resource Providers who own the resources,
2. Framework Providers who own the architectural framework components, and
3. Resource Users who are the main users of the resources or architectural services.

3.1 Functional Coverage of RWI Architectures

This section explores the different functional features provided by the service functions of the existing architectures to support the interactions between resources and resource users and the corresponding business roles inside the RWI ecosystem.

Resource discovery is one of the basic services RWI architectures provide for resource-level access. It allows resource users to lookup and find resources that are made available by resource providers in an RWI community. Resource users specify characteristics of a resource, e.g., the identifier or type they are interested in, and receive (references to) one or more resources that match the requested criteria.

Context information query is a more advanced functionality provided by some RWI architectures for entity level access. It allows resource users to directly access context information in the RWI concerning EoIs or find resources from which such information can be obtained. Unlike resource discovery, context information queries involve semantic resolution of declarative queries and require resources and entities to be adequately modeled and described.

Actuation and control loop support is another advanced functionality providing access to RWI resources at entity level. It allows resource users to declaratively specify simple or complex actuation requests or expected outcomes of actuations on an EoI. The respective functions ensure that resource users are provided with an adequate set of resources able to achieve the specified objectives or that appropriate actions are executed according to the specified outcomes.

Dynamic resource creation is an advanced functionally of some architectures and mainly relates to virtual resources such as processing resources. It enables the dy-

namic instantiation of resources (e.g., processing services) on resource hosts in order to satisfy context information requests and actuation requests.

Session management functionality is provided to support longer lasting interactions between resources and resource users, in particular if these interactions span multiple resources. Longer lasting interactions may require adaptation of the interactions to system dynamics, such as change of availability of resources, e.g., the replacement of one or more resource endpoints during the lifetime of the interaction, shielding this complexity from the resource user.

Access control functionality is essential to ensure that only authorized resource users are able to access the resources. It typically involves authentication of resource users at request time and subsequent authorization of resource usage. Another aspect of resource access is access arbitration, if concurrent access occurs by multiple authorized users. This requires mechanisms to resolve contention if multiple conflicting requests are made including pre-emption and prioritization.

Auditing and billing functionality are necessary to provide accounting and accountability in an RWI architecture. Based on the accounting model, resource users can be charged for the access to resources or provided information and actuation services. Accountability and traceability can be achieved by recording transactions and interactions taking place at the respective system entities.

3.2 Smart Object Model

At its core, the proposed architectural model defines a set of entities and their relationships, the Smart Object Model. The entities form the basic abstractions on which the various system functions previously described operate. The object model reflects a clear separation of concerns at the various system levels and their real-world interrelationships according to the assumptions described in Section 2.1.

A central entity in the Smart Object Model is the concept of a resource. Conceptually, resources provide unifying abstractions for real-world information and interaction capabilities comparable to web resources in the current web architecture. In the same way as a web user interacts with a web resource, e.g., retrieve a web page, the user can interact with the real-world resources, e.g., retrieve sensor data from a sensor. However, while the concept of the web resource refers to a virtual resource identified by a Universal Resource Identifier (URI), a resource in the RWI context is an abstraction for a specific set of physical and virtual resources.

The resources in the Smart Object Model abstract capabilities offered by real-world entities such as sensing, actuation, processing of context and sensor data or actuation loops, and management information concerning sensor/actuator nodes, gateway devices or entire collections of those. Thus a resource has a manifestation in the physical world, which could be a sensor, an actuator, a processing component or a combination of these. In the latter case we refer to it as a composite resource. A resource is unique within a system (domain) and is described by an associated resource description, whose format is uniform for all resources across systems and domains. This uniform resource description format enables and simplifies the reuse of existing resources in different contexts and systems and is a major contribution of the proposed architectural model.

The Smart Object Model distinguishes between the (physical) instances of system resources and the software components implementing the interaction endpoints from the user perspective (Resource End Point – REP). Furthermore, the model distinguishes between the devices hosting the resources (Resource Host) and the network devices hosting the respective interaction end points (REP Host). This separation enables the various system functions described in the previous section to deal with real-world dynamics in an efficient and adequate manner and facilitates different deployment models of a system. Fig. 2 shows the Smart Object Model in terms of entities and their inter-relationships.

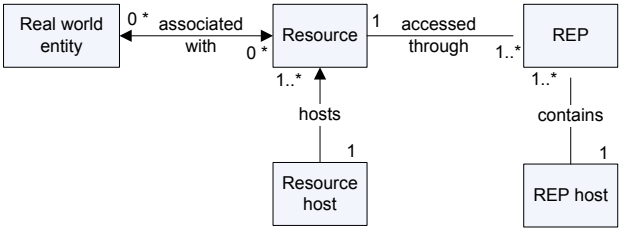

Fig. 2. Key entities and their relationships in the RWI system model

A REP is a software component that represents an interaction end-point for a physical resource. It implements one or more Resource Access Interfaces (RAIs) to the resource. The same resource may be accessible via multiple REPs, through the same RAI or different ones. In comparison to the current web architecture, REPs can be considered equivalent to web resources, which are uniquely identified by a URI.

The device hosting a resource is referred to as the Resource Host. Sensor nodes are typical examples for resource hosts, but there can be arbitrary devices acting in this role, for example, mobile phones or access points that embed resources. A REP Host is a device that executes the software process representing the REP.

As mentioned before, the resources and REPs are conceptually separated from their hosts to facilitate different deployment options. In some cases a REP host and a resource host can be co-located on the same physical device, e.g., in the case of a mobile phone. Similarly, there may be cases where the REP is not hosted on the resource host itself, for example, a computer in the network or an embedded server may act as the REP host for a resource, which is physically hosted on a sensor node connected to it. This distinction is important when mobility, disconnections and other system dynamics come into play, as it provides a conceptual model to effectively keep the system state consistent for the correct operation of an overall system. Moreover, this separation of concerns provides a means of protecting low-capability resources, e.g., low-power sensor nodes, from attacks by hosting their REPs on more powerful hardware.

Unlike other models, the Smart Object Model considers also real-world entities in its model and manages the dynamic associations between the real world entities and the sensors/actuators that can provide information about them/act upon them. Examples of the real-world entities – also known as Entities of Interest or EoIs – are persons, places, or objects of the real world that are considered relevant to provide a service to

users or applications. A resource in the Smart Object Model thus provides (context) information or interaction capabilities concerning associated real-world entities.

3.3 Interaction Styles

The classes of system functions described in Section 2.1 may be realized through different interaction styles which can be classified along the following dimensions:

- Synchronous or asynchronous: Does the operation block the thread of control and wait for a result (blocking) or is it executed in parallel (non-blocking)?
- Session context: If an interaction depends on previous interactions, then the system must store and maintain the state of a "conversation".
- One-shot or continuous: The interaction may either return a single result immediately or run continuously and return results as they come along.
- Number of participants: Interactions among resources and resource users can be 1:1, 1:n or m:n.

Well-known styles can easily be mapped to these dimensions, for example, synchronous-continuous would be "polling", whereas asynchronous-continuous would be "event-driven".

Each style can be implemented in various ways, depending on the specific system and its requirements. A continuous interaction can e.g. be implemented through a pub/sub service; a complex event processing system via polling in regular intervals or by a simple asynchronous callback mechanism, etc. The different choices determine not only the resource consumption and communication stress on the underlying infrastructure but also the flexibility, extensibility, dependability, determinism, etc. of the implemented system. However, the interfaces to these choices at the implementation architecture level should be uniform as this allows the exchange of one communication infrastructure by another without requiring major recoding efforts of an application and also enables an n-system development and deployment.

Also, the interaction patterns manifest themselves in communication flows of different characteristics. In order to effectively support these flows, different types of communication services may be required from the underlying communication service layer. Table 1 shows a simple way to assess and compare interaction styles of different architectures by arranging the possible combinations in a two-dimensional grid.

Table 1. Classification of interactions

Architecture name		Synchronicity		Session Context	
		sync	async	yes	no
Duration	One-shot				
	Continuous				
Participants	1:1				
	1:n				
	m:n				

4 Analysis of Existing Architectures

In this section we briefly review five of the most relevant RWI architecture approaches with respect to the functional coverage provided in the context of the above defined reference architecture. These approaches have been recently developed in the ASPIRE, FZI Living Lab AAL (Ambient Assisted Living), PECES, SemsorGrid4Env and SENSEI European research projects. Following this, a number of other relevant architectures are identified and a table at the section's end summarizes the functional coverage of the five main architectures.

4.1 ASPIRE

The ASPIRE architecture [ASPIRE] is based on EPGglobal [EPC] with a number of objective-specific additions. In a Radio Frequency Identification (RFID) based scenario, the tags act as hosts for the resources in form of Electronic Product Codes (EPCs), IDs or other information as well as for value-added information in form of e.g. sensor data. The resource hosts are abstracted through the RFID readers due to the passive communication of the tags. The Object Naming Service (ONS) corresponds to the Entity Directory that returns the URLs of relevant resources for the EPC in question – this is the White Pages service. The EPC Information Service (EPCIS) implements the Resource Directory by storing more rich information of the resource. The information stored covers WHAT, WHERE, WHEN and WHY for an EPC and can be used as a Yellow Pages service. The Application Layer Event (ALE) functionality implements the functionality of a Semantic Query Resolver (SQR). The ALE operates through an Event Cycle specification (ECspec) where resources are defined. ASPIRE introduces a Business Event Generator (BEG) which implements additional logic for interactions using semantics of the specific RFID application. Query planning is done through the definition of an ECspec and can be mapped into the SQR. In addition, three request modes are standardized corresponding to interactions. "Subscribe" issues a standing request for asynchronous reporting of an ECspec and is defined as a continuous request with no one-shot scenario. "Poll" issues a standing request for synchronous reporting of an ECspec and maps to the synchronous interaction, but again only for continuous requests. "Immediate" maps to the synchronous one-shot interaction and requires no ECspec as it focuses on one-shot customized reporting.

4.2 FZI Living Lab AAL

The FZI Living Lab AAL architecture [LLAAL] represents a combination of the service-oriented provision of AAL services and event-driven communication between them, in order to enable a proactive reaction on some emergent situations in the living environment of elderly people. The system is based on the OSGi service middleware and consists of two main sub systems: the service platform openAAL and the ETALIS Complex event processing system (icep.fzi.de). It provides generic platform

services like context management for collecting and abstracting data about the environment, workflow based specifications of system behaviour and semantically-enabled service discovery. Framework and platform services are loosely coupled by operating and communicating on shared vocabulary (most important ontologies: AAL domain, Sensor-ontology). The architecture can be mapped on the RWI Reference Architecture as follows. RWI sensors and RWI actuators are analogous to the AAL sensors and actuators. AAL AP (assisted person) corresponds to the RWI Resource User and RWI Entities of Interest (Entities of Interest) are analogous to the contextual information provided by AAL contextual manager. ETALIS (CEP engine) and Pub-sub service correspond to the RWI CEP Resource and RWI pub-sub service, respectively. Both one-shot and continuous interactions are supported between components, whereas the primary way of interaction is the asynchronous-continuous, i.e. event-driven one.

4.3 PECES

The PECES architecture [PECES] provides a comprehensive software layer to enable the seamless cooperation of embedded devices across various smart spaces on a global scale in a context-dependent, secure and trustworthy manner. PECES facilitates the easy formation of communities and collaboration across smart spaces, thus supporting nomadic users and remote collaboration among objects in different smart spaces in a seamless and automatic way. The PECES middleware architecture enables dynamic group-based communication between PECES applications (Resources) by utilizing contextual information based on a flexible context ontology. Although Resources are not directly analogous to PECES middleware instances, gateways to these devices are more resource-rich and can host middleware instances, and can be queried provided that an application-level querying interface is implemented. Entities of Interest are analogous to the contextual information underlying PECES. These entities are encapsulated as any other contextual information, a model abstraction which can include spatial elements (GIS information), personal elements (personal profiles) and devices and their profiles. The PECES Registry component implements a Yellow Pages directory service, i.e., services are described through attributes, modeled as contextual information, and a range of services (resources). Any service (resource) matching that description may be returned by the registry. Although no “session context” is required, a pre-requirement exists that interacting PECES applications, whether they are entities or resources, must be running the PECES middleware before any interaction may occur. Both one-shot and continuous interactions are supported between components and PECES provides the grouping and addressing functionality and associated security mechanisms that are required to enable dynamic loosely-coupled systems. The number of participants can be m:n, as PECES primarily targets group-based communication scenarios.

4.4 SemsorGrid4Env

The SemSorGrid4Env architecture [SSG4Env] provides support for the discovery and use of sensor-based, streaming and static data sources in manners that were not necessarily foreseen when the sensor networks were deployed or the data sources made available. The architecture may be applied to almost any type of real world entity, although it has been used mainly with real world entities related to natural phenomena (e.g, temperature, humidity, wave length). The types of resources considered are: sensor networks, off-the-shelf mote-based networks or ad-hoc sensors; streaming data sources, normally containing historical information from sensors; and even relational databases, which may contain any type of information from the digital world (hence resource hosts are multiple). These resources are made available through a number of data-focused services (acting as resource endpoints), which are based on the WS-DAI specification for data access and integration and which are supported by the SemSorGrid4Env reference implementation. These services include those focused on data registration and discovery (where a spatio-temporal extension of SPARQL – stSPARQL -, is used to discover data sources from the SemSorGrid4Env registry), data access and query (where ontology-based and non-ontology-based query languages are provided to access data: SPARQL-Stream and SNEEql – a declarative continuous query language over acquisition sensor networks, continuous streaming data, and traditional stored data), and data integration (where the ontology-based SPARQL-Stream language is used to integrate data from heterogeneous and multimodal data sources). Other capabilities offered by the architecture are related to supporting synchronous and asynchronous access modes, with subscription/pull and push-based capabilities, and actuating over sensor networks, by in-network query processing mechanisms that take declarative queries and transform them into code that changes the behavior of sensor networks. Context information queries are supported by using ontologies about roles, agents, services and resources.

4.5 SENSEI

The SENSEI architecture [SENSEI] aims at integrating geo-graphically dispersed and internet interconnected heterogeneous WSAN (Wireless Sensor and Actuator Networks) systems into a homogeneous fabric for real world information and interaction. It includes various useful services for both providers and users of real world resources to form a global market space for real world information and interaction. SENSEI takes a resource oriented approach which is strongly inspired by service oriented principles and semantic web technologies. In the SENSEI architecture each real world resource is described by a uniform resource description, providing basic and semantically expressed advanced operations of a resource, describing its capabilities and REP information. These uniform descriptions provide the basis for a variety of different supporting services that operate upon. On top of this unifying framework SENSEI builds a context framework, with a 3 layer information model. One of the key support services is a rendezvous mechanism that allows resource users to discover and query resources that fulfill their interaction requirements. At lower level this is realized by a federated resource directory across different administrative domains. On top of it, the

architecture provides a semantic query support, allowing resource users to declaratively express context information or actuation tasks. Using a semantic query resolver and the support of an entity directory (in which bindings of real world resources and entities are maintained) suitable sensor, actuator and processing services can be identified and dynamically combined in order to provide request context information or realize more complex actuation loops. In order to increase flexibility at run-time, dynamic resource creation functionality allows for the instantiation of processing resources that may be required but not yet deployed in the system. The SENSEI architecture supports both one-time and longer lasting interactions between resource users and resource providers, that can be streaming or event based and provides mechanism through the execution manager to maintain a desired quality of information and actuation despite system dynamics. A comprehensive security framework provides functions for the realization of a variety of different trust relationships. This is centered on a security token service for resource users and AAA (Authentication, Authorization and Accounting) service to enforce access at the access controlled entities covering resources and framework functions. Furthermore AAA services perform accounting and auditing for authorized use of real world resources.

4.6 Other Architectures

A number of projects focus on aspects beyond the architectural blueprint presented in this chapter, the most prominent being SPITFIRE [SPITFIRE] and IoT-A [IoT-A]. As these projects have just started and have not produced architectures yet, they can only be included in the future work on an RWI reference architecture.

SPITFIRE aims at extending the Web into the embedded world to form a Web of Things (WoT), where Web representations of real-world entities offer services to access and modify their physical state and to mash up these real-world services with traditional services and data available in the Web. SPITFIRE extends the architectural model of this chapter by its focus on services, supporting heterogeneous and resource-constrained devices, its extensive use of existing Web standards such as RESTful interfaces and Linked Open Data, along with semantic descriptions throughout the whole architecture.

The IoT-A project extends the concepts developed in SENSEI further to provide a unified architecture for an Internet of Things. It aims at the creation of a common architectural framework making a diversity of real world information sources such as wireless sensor networks and heterogeneous identification technologies accessible on a Future Internet. While addressing various challenges [ZGL+], it will provide key building blocks on which a future IoT architecture will be based, such as a global resolution infrastructure that allows IoT resources to be dynamically resolved to entities of the real world to which they can relate.

4.7 Summary of Project Realizations

Table 2a. Functional coverage of current RWI architecture approaches

	SENSEI	ASPIRE	PECES	SemSorGrid4Env	LLAAL
Resource discovery	Using a standalone or federated (peered) resource directory as a rendezvous point that stores resource descriptions	Using ONS and EPCIS for id-based or information centric resource discovery	Distributed registry, information centric (yellow pages)	Using an RDF-based registry of data sources, with spatio-temporal query (stSPARQL) capabilities	Using an RDF-based registry
Context information query	SPARQL based query interface, semantic query resolution involving ED (Entity Directory) and RD (Resource Directory)	Query of the EPCIS which stores WHAT, WHERE, WHEN and WHY for all resources	Via registry or queries to resources via roles	Role, agent, service and resource ontologies as information models and corresponding stSPARQL queries	Contextual Manager provides an ontology-based information storage that captures sensor information and user input
Actuation and control loops	Actuation task model, support for execution of control loops through execution manager	Actuation based on applications and business events in the related components.	Implicit in application code	In-network query processing capabilities (SNEE) with mote-based sensor networks	Procedural Manager manages and executes easy to define and installation independent workflows which react to situations of interest
Dynamic resource creation	Dynamic instantiation of processing resources using predefined templates	Incorporated in ALE and BEG based on level of application interaction	Dynamic roles and dynamic smart spaces	Data services are generated dynamically according to WS-DAI (Web Services Data Access and Integration) indirect access mode	Composer analyses services which are available in a certain installation and selects and combines those services to achieve the (abstract) service goals

Table 2b. Functional coverage of current RWI architecture approaches

	SENSEI	ASPIRE	PECES	SemSorGrid4Env	LLAAL
Session management	Execution manager responsible for main-tenance of long last-ing requests	ECspecs, filtering and collection as well as read cycles.	Implicit via middleware	Limited management, through WS-DAI indirect access mode	N/A
Access control	Security token service for resource users and AAA service to en-force access at the access controlled entity, resource access proxy service for cross-domain access	Role-based access control for individual middleware compo-nents	Expressive (based on ontologies), enforce-able	N/A	N/A
Auditing and billing	auditing for AAA service to perform accounting and au-thorized use	N/A	N/A	N/A	N/A
Underlying Resource model	SENSEI resource description and ad-vanced resource descriptions with semantics	EPC and value-added sensing	PECES role ontology	According to W3C Semantic Sensor Network Ontology	LL AAL Sensor-level ontology. It supports integration of sensors and AAL services
Underlying context model	3 layer information model (raw data, observation & meas-urement, ontology based context model)	EPCIS standard	PECES context ontology	Observation&Measuremen t, role, agent, service and resource ontologies	Context Ontology: low- and top-level. It supports context reasoning from a low-level sensor-based model to a high-level service-oriented model

5 Concluding Remarks

The chapter presents a blueprint for design of systems capable of capturing information from and about the physical world and making it available for usage in the digital world. Based on the inputs and analysis of several research projects in this domain, it provides an outline of the main architectural components, interactions between the components and a way to describe the information and capabilities of the components in a standardized manner. Although not the final RWI reference architecture, the blueprint already captures the main features of such systems well as can be seen from the analysis of architectures designed in five different projects.

The work on the IoT reference architecture will continue to be driven by the RWI group of the FIA in collaboration with the FP7 IOT-i coordinated action project (http://www.iot-i.eu) and the IERC, the European Research Cluster on the Internet of Things (http://www.internet-of-things-research.eu/). The results will be contributed to the FIA Architecture track. It is expected that the final architecture will be ready by the end of 2011.

References

[ASPIRE] Advanced Sensors and lightweight Programmable middleware for Innovative RFID Enterprise applications, FP7, http://www.fp7-aspire.eu/

[CONET] Cooperating Objects NoE, FP7, http://www.cooperating-objects.eu/

[EPC] EPCGlobal: The EPCglobal Architecture Framework 1.3 (March 2009), available from: http://www.epcglobalinc.org/

[HAL] S. Haller: The Things in the Internet of Things. Poster at the Internet of Things Conference, Tokyo (IoT, 2010) (2010), available at http://www.iot-a.eu/public/news/resources/TheThingsintheInternetofThings_SH.pdf [Accessed Jan. 24, 2011]

[IoT-A] EU FP7 Internet of Things Architecture project, http://www.iot-a.eu/public

[LLAAL] FZI Living Lab AAL, http://aal.fzi.de/

[PECES] PErvasive Computing in Embedded Systems, FP7, http://www.ict-peces.eu/

[SemsorGrid4Env] Semantic Sensor Grids for Rapid Application Development for Environmental Management, FP7, http://www.semsorgrid4env.eu/

[SENSEI] Integrating the Physical with the Digital World of the Network of the Future, FP7, http://www.ict-sensei.org

[SPITFIRE] Semantic-Service Provisioning for the Internet of Things using Future Internet Research by Experimentation, FP7, http://www.spitfire-project.eu/

[ZGL+] Zorzi, M., Gluhak, A., Lange, S., Bassi, A.: From Today's INTRAnet of Things to a Future INTERnet of Things: A Wireless- and Mobility-Related View. IEEE Wireless Communications 17(6) (2010)