

UNIVERSITÀ DELLA CALABRIA

DIPARTIMENTO DI INGEGNERIA INFORMATICA, MODELLISTICA,
ELETTRONICA E SISTEMISTICA



Dottorato di Ricerca in
Information and Communication Technologies
XXX Ciclo

Tesi di Dottorato

**A Methodology for the Development of
Autonomic and Cognitive
Internet of Things Ecosystems**

Claudio Savaglio



UNIVERSITÀ DELLA CALABRIA

Dottorato di Ricerca in
Information and Communication Technologies
XXX Ciclo

Tesi di Dottorato

**A Methodology for the Development of
Autonomic and Cognitive
Internet of Things Ecosystems**

Claudio Savaglio

Coordinatore
Prof. Felice Crupi

Supervisore
Prof. Giancarlo Fortino

DIPARTIMENTO DI INGEGNERIA INFORMATICA, MODELLISTICA,
ELETTRONICA E SISTEMISTICA
Novembre 2017

Settore Scientifico Disciplinare: ING-INF/05

UNIVERSITÀ DELLA CALABRIA



UNIVERSITÀ DELLA CALABRIA


Dipartimento di Ingegneria Informatica, Modellistica,
Elettronica e Sistemistica


**Dottorato di Ricerca in
Information and Communications Technology**


CICLO
XXX

**A Methodology for the Development of
Autonomic and Cognitive
Internet of Things Ecosystems**

Settore Scientifico Disciplinare ING-INF/05

Coordinatore: Ch.mo Prof. Felice Crupi
Firma 

Supervisore/Tutor: Ch.mo Prof. Giancarlo Fortino
Firma 

Dottorando: Dott. Claudio Savaglio
Firma 

At the end of this walk of life that is the Ph.D., and before moving on to the next adventures, I want to thank my advisors, Proff. Wilma Russo and Giancarlo Fortino, for their daily support and Proff. Mengchu Zhou and Carlos E. Palau for having me as Visiting Scholar respectively at New Jersey Institute of Technology and at Universitat Politècnica de València. Thank you also to Antonio and Raffaele, my workmates and friends, as well as to all the colleagues which crossed my path. Last but not least, to whom has always been with me, and to whom are missing but would have wanted to be here: my gratitude goes to all of you.

Abstract

Advancements on microelectromechanical systems, embedded technologies, and wireless communications have recently enabled the evolution of conventional everyday things in enhanced entities, commonly defined *Smart Objects (SOs)*. Their continuous and widespread diffusion, along with an increasing and pervasive connectivity, is enabling unforeseen interactions with conventional computing systems, places, animals and humans, thus fading the boundary between physical and digital worlds.

The *Internet of Things (IoT)* term just refers to such futuristic scenario, namely a loosely coupled, decentralized and dynamic ecosystem in which billions (even trillions) of self-steering SOs are globally interconnected becoming active participants in business, logistics, information and social processes. Indeed, SOs are able to provide highly pervasive cyberphysical services to both humans and machines thanks to their communication, sensing, actuation, and embedded processing capabilities.

Nowadays, the systemic revolution that can be led through the complete realization of the IoT vision is just at its dawn. As matter of facts, whereas new IoT devices and systems have been already developed, they often result in poorly interoperating “Intra-nets of things”, mainly due to the heterogeneity featuring IoT building blocks and the lack of standards. Thus, the development of massive scaled (the total number of “things” is forecasted to reach 20.4 billion in 2020) and actually interoperable IoT systems is a challenging task, featured by several requirements and novel, even unsurveyed, issues. In this context, a multidisciplinary and systematic development approach is necessary, so to involve different fields of expertise for coping with the cyberphysical nature of IoT ecosystem. Henceforth, full-fledged IoT methodologies are gaining traction, aiming at systematically supporting all development phases, addressing mentioned issues, and reducing time-to-market, efforts and probability of failure.

In such a scenario, this Thesis proposes an application domain-neutral, full-fledged agent-based development methodology able to support the main engineering phases of IoT ecosystems. The definition of such systematic ap-

proach resulted in *ACOSO-Meth (Agent-based COoperating Smart Objects Methodology)*, which is the major contribution of this thesis along with other interesting research efforts supporting (i.e., a multi-technology and multi-protocol smartphone-based IoT gateway) and extending (i.e., a full-fledged approach to the IoT services modeling according to their opportunistic properties) the main proposal. Finally, to provide validation and performance evaluation of the proposed ACOSO-Meth approach, four use cases (related to different application contexts such as a smart university campus, a smart digital library, a smart city and a smart workshop) have been developed. These research prototypes showed the effectiveness and efficiency of the proposed approach and improved their respective state-of-the-art.

Riassunto

Recenti sviluppi nei campi della tecnologia integrata, microelettromeccanica, e comunicazioni wireless hanno consentito l'evoluzione di semplici oggetti di uso quotidiano in prodotti tecnologicamente avanzati, comunemente definiti *Smart Objects (SOs)*. La loro ampia e progressiva diffusione, supportata da una connettività crescente e pervasiva, li rende capaci di interagire in maniera adattiva con sistemi di calcolo tradizionali, luoghi e persone, contribuendo così a sfumare il confine tra il mondo reale e quello virtuale.

L' *Internet of Things (IoT)* fa riferimento proprio ad un tale scenario, cioè un ecosistema dinamico, decentralizzato e destrutturato, nel quale miliardi (eventualmente triliardi) di SOs, autonomi ed in continua evoluzione, sono connessi su scala globale e prendono parte attivamente ai processi sociali, commerciali, logistici e informatici. Infatti, sfruttando le proprie capacità di comunicazione, rilevazione, computazione ed attuazione, gli SOs sono in grado di fornire servizi cyberfisici altamente avanzati e pervasivi ad utenti umani e ad altre macchine.

La rivoluzione sistemica derivante da una piena realizzazione dell'IoT è, tuttavia, ancora agli albori. Infatti, sebbene nuovi dispositivi e sistemi IoT siano già stati sviluppati, il più delle volte questi costituiscono delle "Intra-net of Things", cioè sistemi isolati che non interagiscono reciprocamente a causa dell'eterogeneità delle loro componenti e dell'assenza di standard di riferimento. Lo sviluppo su larga scala di sistemi IoT effettivamente interoperabili (previsioni stimano in 20.4 miliardi il numero totale di "cose" nel 2020), infatti, rappresenta un compito complesso, con numerosi requisiti da rispettare e nuove criticità, per certi aspetti ancora non del tutto enucleate. In tale contesto, per fronteggiare la natura cyberfisica degli ecosistemi IoT oggetti di sviluppo si rende necessario un approccio sistematico e multidisciplinare, in grado di coinvolgere diverse professionalità e competenze. Pertanto, metodologie di sviluppo stanno guadagnando popolarità, per supportare pienamente la realizzazione di ecosistemi IoT, dalla fase di analisi a quella implementativa, riducendo al tempo stesso gli sforzi ed i tempi necessari.

Proprio in tale direzione, questa Tesi propone una metodologia di sviluppo, rinominata *ACOSO-Meth (Agent-based COoperating Smart Objects Methodology)*, che è completa, neutrale rispetto al dominio applicativo, e basata sul paradigma ad agenti, con l'obiettivo di supportare le fasi principali di ingegnerizzazione di ecosistemi IoT. ACOSO-Meth rappresenta il cardine di questa Tesi, assieme ad altri contributi che la supportano (ad es., un framework comparativo per analizzare middleware, framework e piattaforme IoT, un gateway IoT implementato su uno smartphone capace di interfacciare più tecnologie e protocolli di comunicazione) ed estendono (un approccio per la modellazione a tutto tondo dei servizi IoT in accordo alle loro caratteristiche opportunistiche). Infine, per validare ACOSO-Meth e valutarne le prestazioni, sono stati sviluppati e presentati quattro casi d'uso (relativi a differenti scenari applicativi quali smart university, smart digital library, smart city e smart workshop) che hanno dimostrato l'efficacia e l'efficienza dell'approccio, contribuendo altresì a migliorare lo stato dell'arte.

Contents

List of Figures	vii
List of Tables	x
1 Motivations, Objective, Contributions and Organization of the Thesis	1
1.1 Motivations	1
1.2 Objective and Contributions of the Thesis	2
1.3 Structure of the Thesis	3
2 Background and Framework-supported State-of-the-Art Analysis	5
2.1 Introduction	5
2.2 Background: IoT visions and enabling paradigms	6
2.2.1 Agent-based Computing	7
2.2.2 Autonomic Computing	9
2.2.3 Cognitive Computing	9
2.3 Services in the IoT: state-of-the-art	10
2.3.1 Analysis and limitations of IoT services specifications ..	13
2.4 IoT ecosystem development requirements	17
2.5 Framework-supported state-of-the-art survey	21
2.5.1 Analysis phase	21
2.5.2 Design phase	22
2.5.2.1 Simulation-based Design	24
2.5.3 Implementation phase	25
2.5.4 Development Methodologies	26
2.6 Comparative Analysis	27
2.7 Summary	29

3	A methodology for the development of autonomic and cognitive Internet of Things ecosystems	31
3.1	Introduction	31
3.2	Analysis phase	32
3.3	Design phase	34
3.3.1	A Hybrid Agent-Oriented Simulation-based Design Approach	36
3.4	Implementation phase	38
3.5	Discussion	42
3.6	A methodology extension: towards Opportunistic IoT services	45
3.6.1	Opportunistic IoT service modeling	46
3.7	Summary	50
4	Smart and Interoperable IoT Ecosystems	53
4.1	Smart Unical	53
4.1.1	Analysis phase	55
4.1.2	Design phase	56
4.1.3	Implementation phase	56
4.2	Smart Digital Libraries	64
4.2.1	Analysis phase	65
4.3	Opportunistic IoT services	72
4.3.1	Smartphone-based Mobile IoT Gateway	72
4.3.2	Smart City scenario: Crowd Safety service	79
4.3.3	Smart Workshop scenario: SmartConnectivity and SmartHealth services	80
4.4	Summary	83
5	Conclusions, Publications and Future Work	85
5.1	Publications related with this Thesis	88
5.1.1	Journal Articles	88
5.1.2	Book Chapters	89
5.1.3	Conference Papers	89
5.2	Future Work	92
	References	95

List of Figures

2.1	Smart Object (SO) components	7
2.2	Agent-based Computing	8
2.3	Autonomic Computing	10
2.4	Cognitive Loop	11
2.5	IoT service modeling in (a) [1]; (b) [2]; (c) [3]; (d) [4]; (e) [5]; (f) [6].	14
2.6	Scale in IoT systems	18
3.1	Relationships among ACOSO-Meth metamodels at different phases	33
3.2	Analysis Phase: High-Level SO Metamodel	35
3.3	Design Phase: ACOSO-based SO Metamodel	36
3.4	INET node metamodel	37
3.5	Layered view of (a) the ACOSO SO Architecture and (b) the OMNeT++ Node Architecture	40
3.6	JACOSO three-layered architecture	40
3.7	Implementation phase: JACOSO SO Metamodel	42
3.8	Proposed IoT Service model	46
3.9	IoT Entities and their roles in IoT Service provision	47
3.10	Smart Object modeling and main features related to IoT Service provision (in red the extensions with respect to [3])	48
3.11	Abstract IoT Service modeled through an FSA	50
4.1	The Smart UniCal infrastructure: the SmartBridge part (dotted yellow bordered area) which crosses the SmartDIMES (yellow bordered area with building identification codes)	54
4.2	High-Level SmartBridge Metamodel at analysis phase.	55
4.3	ACOSO-based SmartBridge Metamodel at design phase.	56
4.4	JACOSO-based SmartBridge Metamodel at implementation phase.	57

4.5	Interaction diagram of the Smart Bridges Smart Vibration service.	58
4.6	Snapshot of the IoT devices of (a) SmartBridge, (b) SmartDIMES and (c) SmartSenSysCalLab.....	58
4.7	SmartSenSysCal, SmartDIMES, and SmartBridge performance evaluation considering different communication paradigms (C/S or P2P) and MGR models (D or N)	61
4.8	SmartDesk Model	66
4.9	JSON representation of a smart desk according to the SO metadata model.....	67
4.10	Digital Library Main Concepts	67
4.11	Communication scenario for testing the smartphone-centric application.	73
4.12	Software architecture of the smartphone-centric gateway.	75
4.13	Specific implementation of the smartphone centric gateway application.	76
4.14	Screenshots of the mobile gateway application: a) Main GUI, b) Multiple interfaces choice and activation, c) Data received on a specific interface.	77
4.15	CPU load by activating all the interfaces.....	78
4.16	Memory usage by using all the interfaces.....	78
4.17	Energy consumption.	79
4.18	Metamodeling of the Crowd Safety opportunistic IoT service ...	81
4.19	Simplified FSM describing Crowd Safety IoT Service	81
4.20	Metamodeling of the SmartHealth and SmartConnectivity opportunistic IoT services	82

List of Tables

2.1	Comparison of service specifications featuring [1], [2], [3] and [7]	16
2.2	System-Level Requirements (SLRs)	19
2.3	Thing-Level Requirements (TLRs)	20
2.4	Related work's support to the development phases (Y = totally supported, P = partially supported, Blank = not supported)	28
2.5	Related work's fulfillment of development requirements (Y = totally fulfilled, P = partially fulfilled, Blank = not fulfilled) . . .	29
3.1	Comparison of main entities of SOs metamodels of ACOSO-Meth, IEEE P2413, AIOTI and IoT-A.	33
3.2	Mapping guidelines	39
3.3	Evolution of the SO main concepts from the analysis to the implementation phase	44
4.1	Smart Objects constituting Smart Unical along with their provided services	54
4.2	Main hw/sw characteristics of the IoT devices used to implement Smart Unical SOs and their service	60
4.3	Operation modalities of the Smart Unical Smart Objects	63
4.4	Smart Unical performance evaluation	64
4.5	Mapping between the Resource DLRM Concepts, the Smart Object and the SmartDesk ones	68
4.6	Mapping between the DLRM Content Domain Concepts, the Smart Object and the SmartDesk ones	69
4.7	Mapping between the DLRM User Domain Concepts, the Smart Object and the SmartDesk ones	69
4.8	Mapping between the DLRM Functionality Domain Concepts, the Smart Object and the SmartDesk ones	70
4.9	Mapping between the DLRM Policy Domain Concepts, the Smart Object and the SmartDesk ones	70

List of Tables

4.10 Mapping between the DLRM Quality Domain Concepts, the Smart Object and the SmartDesk ones	71
4.11 Mapping between the DLRM Architecture Domain Concepts, the Smart Object and the SmartDesk ones	71
4.12 Smartphones used for the testbed.	76
4.13 IoT devices connected to the Smartphone-centric application throughout several interfaces.	77

List of Abbreviations

<i>ABC</i>	Agent-Based Computing
<i>ACL</i>	Agent Communication Language
<i>ACOSO</i>	Agent-based COoperating Smart Object
<i>ACOSO-Meth</i>	Agent-based COoperating Smart Objects Methodology
<i>AIOTI</i>	Alliance for the Internet of Things
<i>AMS</i>	Agent Management System
<i>API</i>	Application Programming Interface
<i>BDI</i>	Belief, Desire, and Intention
<i>BMF</i>	Building Management Framework
<i>BPMN</i>	Business Process Model Notation
<i>BSN</i>	Body Sensor Network
<i>C\S</i>	Client Server
<i>CASE</i>	Cloud-assisted Agent-based Smart home Environment
<i>CMS</i>	CommunicationManagementSubsystem
<i>CPU</i>	Central Processing Unit
<i>CVO</i>	Composite Virtual Object
<i>DES</i>	Discrete Event System
<i>DF</i>	Directory Facilitator
<i>DL</i>	Digital Library
<i>DLRM</i>	Digital Library Reference Mode
<i>DMS</i>	DeviceManagementSubsystem
<i>EoI</i>	Entities of Interest
<i>e-SOA</i>	event-driven Service Oriented Architecture
<i>FIPA</i>	Foundation for Intelligent Physical Agents
<i>FSA</i>	Finite State Automaton
<i>HVAC</i>	Heating, Ventilation and Air Conditioning
<i>HW</i>	Hardware
<i>IDE</i>	Integrated Development Environment
<i>IEEE</i>	Institute of Electrical and Electronics Engineers
<i>IoT</i>	Internet of Things

<i>IoT-A</i>	Internet of Things - Architecture
<i>IP</i>	Internet Protocol
<i>J2ME</i>	Java Micro Edition
<i>JACOSO</i>	JADE-based ACOSO
<i>JADE</i>	Java Agent DEvelopment Framework
<i>JSON</i>	JavaScript Object Notation
<i>KMS</i>	Knowledge Base ManagementSubsystem
<i>MAC</i>	Medium Access Control
<i>MAPS</i>	Mobile Agent Platform for Sun SPOTs
<i>MAS</i>	Multi Agent System
<i>MDR</i>	Message Delivery Ratio
<i>MGR</i>	Message Generation Rate
<i>OMG</i>	Object Management Group
<i>OS</i>	Operating System
<i>OWL-S</i>	Web Ontology Language for Service
<i>P2P</i>	Peer-to-Peer
<i>PC</i>	Personal Computer
<i>PoI</i>	Properties of Interest
<i>QoS</i>	Quality of Service
<i>RAM</i>	Random Access Memory
<i>RE</i>	Residual Energy
<i>REST</i>	REpresentational State Transfer
<i>RFID</i>	Radio Frequency Identification
<i>ROM</i>	Read Only Memory
<i>RTT</i>	Round Trip Time
<i>RWO</i>	Real World Object
<i>SBD</i>	Simulation-Based Design
<i>SDL</i>	Smart Digital Library
<i>SLR</i>	System Level Requirement
<i>SO</i>	Smart Object
<i>SOA</i>	Service-Oriented Architecture
<i>SPEM</i>	Software Process Engineering Modeling
<i>SPINE</i>	Signal Processing in Node Environment
<i>SSN</i>	Semantic Sensor Networks
<i>SW</i>	Software
<i>TCP</i>	Transmission Control Protocol
<i>TLR</i>	System Level Requirement
<i>TMS</i>	TaskManagementSubsystem
<i>UDP</i>	User Datagram Protocol
<i>UML</i>	Unified Modeling Language
<i>USDL</i>	Unified Service Description Language
<i>VE</i>	Virtual Entity
<i>VO</i>	Virtual Object
<i>WSAN</i>	Wireless Sensor and Actuator Network
<i>WSN</i>	Wireless Sensor Network

Motivations, Objective, Contributions and Organization of the Thesis

1.1 Motivations

Everyday objects are being continuously augmented with novel communication, sensing, actuation and computation capabilities, so extending their conventional uses. Therefore, they have been generically defined “smart” and progressively exploited in a plethora of application domains (health, transportation, manufacturing, etc.). The massive proliferation and the global networking of such heterogeneous Smart Objects (SOs) are pushing an epochal paradigm shift from the current human-centered “Internet” to the so called “Internet of Things” (IoT), a global interconnected scenario in which SOs, conventional computing systems, and humans communicate and cooperate in a synergic fashion to implement cyberphysical and highly pervasive services. The management of such a scenario will require a minimal human intervention because bio-inspired computing paradigms, such as autonomic and cognitive computing, will be jointly applied, both at Thing- and System-level, to pursue higher degrees of autonomy, adaptivity, and smartness. As result, the IoT promises to change the way we live and work in a few years, with not entirely predictable consequences. Because of its disruptive impact, the IoT has become a prominent topic within the academia, industry and society, being widely recognized as the most convincing candidate for leading the next Industrial revolution. Indeed, the IoT market value is expected to exceed one trillion euros by the 2020 just in the European Union, when it is foreseen that almost 26 billion of SOs (eight SOs per person) will daily impact our life.

An ecosystem is generically defined as a set of, eventually heterogeneous, communities, which comprise both biotic and abiotic components (in their turn, more or less heterogeneous) interacting with each others and with the surrounding environment. Likewise, IoT ecosystems (e.g., Smart Cities) consist of numerous and notably different systems (e.g., Smart Roads, Smart Buildings, Smart Grids), which in their turn integrate heterogeneous but interacting components (e.g., human users, cars, smartphones, gateways, smart meters) for realizing innovative and contextualized services.

For making such IoT ecosystems actually dynamic and proactive, specific requirements (both at System- and Thing-level) need to be met and proper methodologies followed: indeed the development, management and integration in real-world applications of IoT ecosystems are complex and challenging tasks. Broadly speaking, using an engineering methodology is widely recognized as a fundamental practice in any system development process, since the manual and non-systematic application of complex techniques, methods and frameworks would very likely reduce effectiveness, increase development time and tend to be error-prone. Particularly in the case of IoT ecosystems, notably dynamic and heterogeneous with each others in terms of functionalities, scales, and underlying technology, the need for a full-fledged development methodology is much as ever crucial. However, despite a variety of research efforts in the IoT context individually focusing on device, network and application design, a full-fledged and general methodology to support the entire IoT ecosystem development process, from analysis to implementation, is missing. Overlooking such deficiency could be a critical pitfall, compromising the full exploitation of the actual IoT potential.

1.2 Objective and Contributions of the Thesis

The objective of the Thesis is the definition of a methodology, named *ACOSO-Meth (Agent-based COoperating Smart Objects Methodology)*, for fully supporting the development of autonomic and cognitive IoT ecosystems. This Thesis contributes to the state-of-the-art in IoT system engineering with the following three main contributions:

- The first contribution is the design of a comparison framework comprising IoT fundamental development requirements. The systematic identification of the fundamental IoT development requirements and properties raised from a thorough state-of-the-art analysis, and, to the best of our knowledge, such analytic review work was lacking in the literature. The comparison framework has inspired the ACOSO-Meth development but it can be reused to analyze future work in the field.
- The second contribution is represented by the proposed ACOSO-Meth methodology, that aims at supporting the whole development process of IoT ecosystems, from the analysis to the design and finally implementation phase. ACOSO-Meth follows an application-neutral approach that is based on the jointly exploitation of well-known computing paradigms (in particular, agent-based, autonomic, and cognitive computing) and supported by a set of metamodels (located at different abstraction levels but strongly interrelated), development frameworks (i.e., the ACOSO middleware) and simulation platforms (i.e., the OMNeT++ network simulator). Two case studies have been prototyped and reported to show the effectiveness and efficiency of the proposed ACOSO-Meth in different application scenarios.

Specifically, the case studies are: (a) Smart Unical, a complex IoT ecosystem providing cyberphysical services related to structural, indoor space and wellness monitoring within a university campus; (b) Smart Digital Library, in which it is shown how SOs can be included into a Digital Library for being effectively discovered, queried and managed.

- The third contribution refers to a research line extending the proposed ACOSO-Meth methodology and specifically focused on IoT services, which promise to play a central role in the IoT ecosystems. Differently from conventional computing services (e.g., web-services, and ubiquitous services) that are usually loosely impacted by context-awareness, co-location or transience, IoT services require to actually consider the overall spatio-temporal context of the heterogeneous entities involved in the service provisioning. Therefore, a novel and full-fledged approach to IoT service modeling, aiming to fully support the subsequent phases of verification and simulation, is presented and its application shown in two concrete case studies related to (c) crowd safety on a large mass event, in the context of a Smart City; (d) connectivity recovery and monitoring of workers' health status, in the context of a Smart Workshop.

Most of this Thesis work has been carried out also under the framework of the *INTER-IoT H2020* EU research project (<http://www.inter-iot-project.eu/>), that aims at the development of an open cross-layer framework to provide voluntary interoperability among heterogeneous IoT platforms spanning single and/or multiple application domains.

1.3 Structure of the Thesis

This Thesis is organized as follows.

Chapter 2 contains a review of the currently available visions on the IoT, emphasizes the adopted SO-based IoT perspective, and includes a brief overview of its main enabling paradigms. The discussion is focused, however, on the state-of-the-art analysis of IoT services, architectures, platforms, middlewares and methodologies. Instead of an exhaustive survey, unfeasible due to their heterogeneity and definitively not functional for this thesis' purposes, the related works have been presented with respect to (i) their distinctive features in supporting the different development phases; (ii) the fulfilled development requirements, both at Thing- and System-level. A comparison framework has been designed accordingly and presented to analyze the surveyed contributions, but it is suitable for being applied to compare future work in the field.

Chapter 3 presents *ACOSO-Meth*, an application domain-neutral, full-fledged, agent-based approach able to support the main engineering phases of IoT systems and applications and, simultaneously, to fulfill the fundamental System- and things-level requirements. In particular, analysis, design (as well as simulation-based design), and implementation phases are discussed,

along with related modeling techniques, simulation tools and programming specifications. Specifically, a set of operational metamodels, each of which is functional to a different development phase, is presented and the relations among the entity concepts in the different phases are explained. Finally, based on an extension of the ACOSO-Meth, a novel approach aiming at the definition and full-fledged modeling of “Opportunistic IoT Services” is proposed.

Chapter 4 describes some IoT ecosystems with related interesting use cases i.e., a smart university, a smart digital library, a smart city and a smart workshop. These IoT ecosystems represent challenging scenarios whose implementation, specially due to their heterogeneity, required a systematic approach along with proper technological solutions. Therefore, use cases development processes have been (partially or completely) supported by the ACOSO-Meth, that demonstrates flexibility and generality.

Finally, Chapter 5 includes a summary of the main results of this Thesis, concluding remarks and comments on possible future research directions that can derive from the work here presented. For the sake of completeness, a list of the publications related to the Thesis is also reported.

Background and Framework-supported State-of-the-Art Analysis

Arisen at the conjunction of different enabling paradigms and technologies, the Internet of Things (IoT) possesses an enormous, disruptive potential for changing the way we interact with the world, but it also brings challenging development issues.

This chapter introduces the different visions and enabling paradigms behind the IoT, elicits a set of requirements for the IoT ecosystem development, and presents an analysis of the current state-of-the-art of IoT services, architectures, platforms, middlewares and methodologies, surveyed through a comparison framework.

2.1 Introduction

Since early 2000s, technological advances in wireless communication, embedded processing, sensing and actuation, are fueling rapid spread of novel cyberphysical artifacts [8]. Ranging from simple movement detectors and temperature sensors, to more sophisticated smartphones and smart cars, they can sense the physical world, process data, and impact the surrounding environment in different ways, for example by triggering actions through actuators or engaging customized users interactions. In the IoT context, such devices have been massively networked and provided with (different degrees of) intelligence, being defined as “Smart Objects” (SOs) [9] and exploited in a multitude of scenarios, e.g., industrial automation, logistics, utilities management, public security, entertainment, ambient assisted living and wellness, to name just a few. The promise of an “anywhere, anytime, anything and anyone” connectivity for blurring the line between the cyber and real worlds outlines the IoT as a revolutionary concept, rich in potential as well as in multi-facet requirements and development issues [10]. As a matter of fact, an IoT ecosystem development process is intrinsically complex at the Thing-level (“in the small”) as well as at the System-level (“in the large”), because features such as smartness, dynamicity, interoperability and autonomy are trans-

versely required [11]. To comprehensively support needs arising in the complex development of such heterogeneous IoT ecosystems, different mainstream paradigms and approaches (especially in closely related fields of wireless sensor networks, distributed systems, ubiquitous and pervasive computing [12]), have been jointly exploited [13]. Among these, Agent-based Computing (ABC) [14] has been widely recognized as comprehensive and effective support to develop decentralized, dynamic, cooperating and open IoT systems, particularly in conjunction with other complementary paradigms, e.g., cloud [15], autonomous [16] and cognitive [17] computing.

2.2 Background: IoT visions and enabling paradigms

Within the literature there is a number of (even deeply) different IoT definitions due to the overlapping of at least three technical visions, which outline a common IoT scenario but from different perspectives [18]. The “Thing-oriented” vision emphasizes the importance of the IoT devices as joining links between the physical and the virtual worlds. The “Network-oriented” vision is essentially focused on communication aspects which ensure an “anywhere, anytime and anything connectivity”. Finally, the “Semantics-oriented” vision concerns the scalable management and effective exploitation of the massive amount of heterogeneous data generated by IoT devices. We adhere to the “Thing-oriented” perspective and specifically to the SO-based IoT vision, in which SOs are the fundamental IoT building blocks [9]. SOs are autonomous everyday-things augmented with *sensing/actuation*, *data management* and *network capabilities*, as depicted in Figure 2.1. Each of the aforementioned features is essential for making SO self-aware and context-aware in providing *cyberphysical* and *ubiquitous services* to both human and digital users [19]. Differently from passive Radio Frequency Identification (RFID) items and Wireless Sensor Networks’ (WSNs) motes, SOs are not only able to provide identification, sensing, and object-to-object communication, but they can also deeply understand their context, performing ad-hoc networking and complex goal-oriented decision-making. However, with respect to the traditional computer systems, SOs design is notably more challenging. In fact, SOs could also be constrained by limited hardware resources (RAM, storage and CPU), physical dimensions and price politics (imposed by the market). Moreover, SOs are technologically and functionally heterogeneous, so their clustering forms IoT systems of different scales (from a single Smart Home to a whole Smart City) that appear as loose collections of miscellaneous devices and subsystems [10, 20]. Such heterogeneous features and requirements pose several design challenges, and therefore an SO-based IoT ecosystem development and management claim for proper paradigms and techniques. In fact, before their actual deployment, SO-based IoT ecosystems require to be carefully analyzed, designed, programmed and simulated, even more than conventional computer

systems. To such purpose, Agent-based, autonomic and cognitive computing are suitable enabling paradigms.

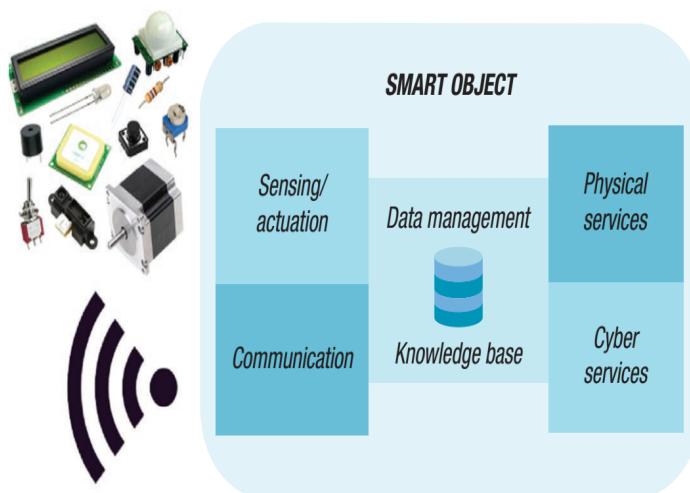


Fig. 2.1. Smart Object (SO) components

2.2.1 Agent-based Computing

Agent-based Computing (ABC) is centered around the concept of an agent [14], a sophisticated software abstraction defining an autonomous, social, reactive and proactive entity. Agents are situated in some environment (namely, world of perceived resources) and act to achieve their design objectives, exhibiting flexible problem solving behaviors (Figure 2.2). Agents, interacting and cooperating to solve problems / realize services that are beyond the capabilities of a single agent, constitute a MAS (Multi Agent System) [21]. MASs are distributed and self-steering societies, featured by a strong situatedness and well-defined organizational relationships, covering variety of application domains (e.g., sociology, economy, logistics). The above characterization, although not exhaustive, indicates that ABC provides a set of key abstractions and metaphors for straightforwardly *modeling* complex systems, their components, interactions and organizational relationships. Beside modeling, ABC is also a well-established *programming* paradigm for concretely implementing agents advanced features and for effectively addressing key requirements typical of modern (distributed) applications. Indeed, agent's, society's and environment's modeling abstractions have been exploited to devise a high-level, distributed programming paradigm, centered around two cornerstones [22]: (i) encapsulation of control (each agent has its own thread of control and reasoning capabilities, thus designing context-aware entities with autonomous

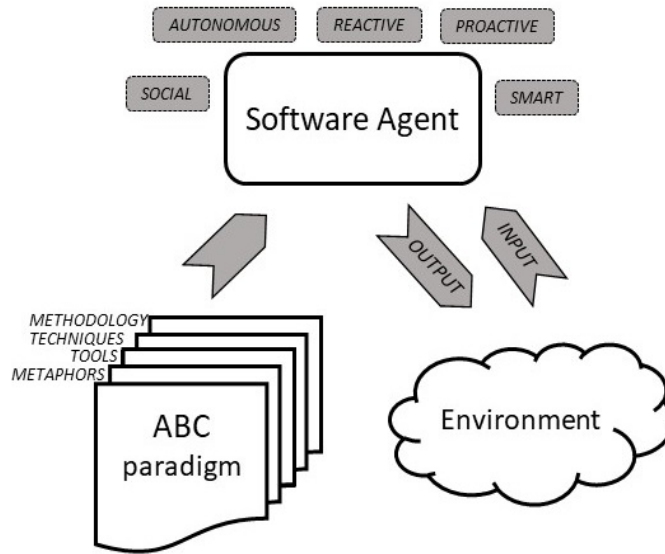


Fig. 2.2. Agent-based Computing

behaviors), and (ii) interaction (including coordination and cooperation mechanisms, based on high-level asynchronous message passing). The adoption of shared communication standards and management specifications (e.g., the IEEE FIPA-based system platforms and communication languages [23, 24]) allows agents to act also as interoperability facilitators, by incorporating a variety of resources and existing legacy systems within the agent society. Such advantages enable agent-based programming paradigm to enhance systems performance (i.e., computational efficiency, reliability, responsiveness), interoperability and scalability, specially with respect to the centralized approaches. Finally, computing systems, modeled and programmed following the agent-oriented approach, can be straightforwardly simulated for effectively studying macro phenomena and patterns, as well as individual behaviors and environment evolution [25]. Indeed, agent-based *simulation* allows evaluating agent-based systems exposing discrete, not linear, adaptive behaviors even in highly interacting, distributed, scaling-up, virtual scenarios. To properly exploit the surveyed agent-oriented metaphors, techniques and tools (thus providing a systematical approach to the agent-based modeling, programming, and simulation), several agent-oriented development *methodologies* have been designed and successfully applied [26]. However, as highlighted in [27, 28], ABC is neither a universal nor necessarily effective development solution, since agent-level and society-level pitfalls can occur from different perspectives (management, conceptual design, etc.), thus outweighing any agent-related benefits. Therefore, the adoption of ABC paradigm needs to be carefully assessed. However, although software agents and IoT arose in very different computer ages

with very different initial purposes (collaborative computation and RFID-based object traceability, respectively), the ABC paradigm has proved to be well-suited to support the development of autonomic and cognitive IoT systems [29].

2.2.2 Autonomic Computing

Natural self-governing systems are defined “*autonomic*” when the establishment of policies and rules is sufficient to guide the self-management process. For example, the human autonomic nervous system is able to free our consciousness from managing major and minor, inter-connected and complex, vital functions. In computer science, the attribute autonomic refers to computing systems that can manage themselves according to high-level objectives initially defined by the administrator [16]. From an architectural point of view, instead, autonomic systems may be considered as interactive collections of autonomic elements, each of which performs its objectives and interactions following their own policies and the system ones. Since autonomic elements have complex life-cycles, it is required that they expose autonomy, proactivity, and goal-directed interactivity with the environment: these are precisely the agent-oriented [14] architectural concepts. The depth analysis of history and of features of the Autonomic Computing falls outside the scope of this Thesis; anyway, the four main aspects that characterize autonomic systems or elements, reported in Figure 2.3, are:

- self-configuration, which enables system and its components to automatically and seamlessly configure themselves following high-level policies, despite vendors’ specifics, technological heterogeneity and low-level details;
- self-optimization, which guides system to continually seek opportunities for improving performance, without human intervention of tuning;
- self-protection, which automates system defense and prevents from system failures due to malicious attacks;
- self-healing, which consists in the automatic detection, diagnosis and repairing of system defections, both hardware and software.

The term “self-*” hence refers to a cognitive system or element which exposes all such features. In a complex scenario such as the IoT, the design of systems that prescind from a constant human monitoring as well as the adoption of techniques automatizing the node’s management are more than ever necessary. For these reasons, the Autonomic Computing principles have inspired the design of numerous IoT architectures and frameworks, as discussed in Section 2.5.

2.2.3 Cognitive Computing

Cognitive systems [17] have been originally considered a “self-*” kind of systems since they autonomously make use of the information gained from the

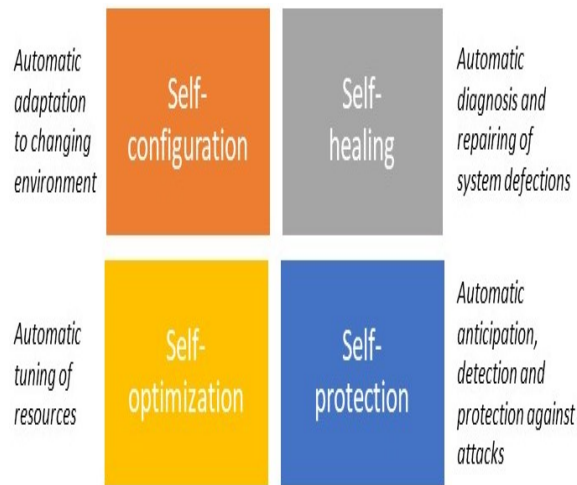


Fig. 2.3. Autonomic Computing

experience of each node to improve the overall system performance. In details, each node is involved in a loop, defined cognition loop and depicted in Figure 2.4, which oversees

- the perception of the current system conditions;
- the planning of actions according to both input and policies;
- the decision between the available solutions;
- the actuation of the plan; and
- the learning from the consequence of the performed actions.

Context-awareness and self-awareness are essential requirements to realize the cognition loop, since it requires that every node has knowledge about itself, its functionalities, and its interfaces to the outside. Just like the autonomic systems, cognitive systems have been conceived to cope with the increasing network complexity but relying as little as possible on human intervention [30]. Hence, in analogy with the autonomic system's architecture, cognitive systems aggregate cognitive agents, which are entities with reasoning capabilities able to cooperate, to act selfishly, or to do both. Since the need of cognition is spread among the system components and layers, and it is not only limited to the management one, the cognitive systems have given rise to an independent line of research, which often exploits the ABC [14] as enabling paradigm.

2.3 Services in the IoT: state-of-the-art

Services notably contributed to the spread of Internet, which evolved from a restricted/small-sized academic and military network into a worldwide plat-

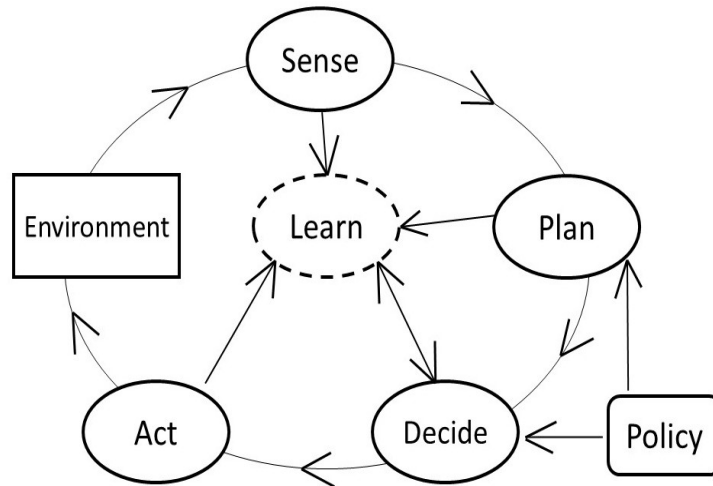


Fig. 2.4. Cognitive Loop

form hosting applications of all kinds [31]. Likewise, services promise to represent the real drivers for the IoT and within the IoT ecosystems [32, 33]. Indeed, everyday objects, conventional computing systems, places, pets and people, supported by ubiquitous and seamless connectivity, take part in novel, advanced, cyberphysical services (indicated as IoT services in the follow), which are expected to revolutionize every application context. Although IoT is gaining momentum, and regardless the substantial background on computing services, the development of an IoT service is a challenging and not fully mastered task. Suffice it to say that, as underlined in [34], the state-of-the-art lacks of a well-established definition for an IoT service. In particular, service modeling is often a neglected or underestimated activity, thus complicating the overall development process and limiting IoT services potentials. Traditional computing services are based on a vertical data flow between physical and application layers, and each service is often independent [31]. Conversely, IoT services exploit both data and cyberphysical functionalities provided by a horizontal landscape of heterogeneous entities, sharing the same resources and environment. Due to their complexity, IoT services require a specific development methodology, so to be thoughtfully designed, formally verified, and simulated. Such a full-fledged approach, so long as supported by a preliminary and systematic modeling phase, paves the way toward reliable, fast and effective IoT service development [35]. Even though there has been much talk about IoT services, the majority of the related results directly or indirectly derive from only a few IoT service models.

One of the most important contributions derives from the IoT-A project [1] (see Figure 2.5(a)), in which a detailed IoT service model has been provided and then exploited as an architectural building block (“IoT Service Layer”)

in different IoT platforms [36] like Butler and ICore. The IoT-A service model extends the previous one developed within the SENSEI project [37], and is totally aligned with the ones of AIOTI [38] and FIWARE [39] initiatives, as well as with the IEEE P2413 “Standard for an Architectural Framework for the IoT” [40]. According to the SSN (Semantic Sensor Networks) ontology [41], it describes an IoT service as a well-defined and standardized interface enabling interactions with the real world, specifically through its Virtual Entities (VEs, namely physical entities abstractions). Indeed, IoT services allow accessing a VEs status, properties and functionalities (sensing, actuation, computation, storage or networking) by means of its Resources, thereby hiding VE heterogeneity/complexity to IoT developers and users. Associations between IoT services and VEs are established according to both dynamic (e.g., IoT service current status, VE location, and VE resource availability) and static information (for example, IoT Service specifications and quality of service, VE id and dimension). In particular, relevant information for each IoT service is coded in a Service Description Model according to the business-oriented USDL (Unified Service Description Language) [42].

This paves the way toward the application of the IoT-A service model within the world of Business Processes (BPs): indeed, by extending the BPMN 2.0 [43], it is possible to treat IoT services as IoT-aware BPs [7]. In particular, as shown in Figure 2.5(e), Participants (Human or any Physical/Digital entity) mainly featured by a role, location, devices and resources, are involved in Activities: these are executable compound of work described by some textual labels (activity type, contextual conditions, annotation, quality metrics, etc.) and implemented through both sensing or actuation tasks, namely atomic operations concretely realizing the related activity. Such conceptual view on IoT-services according to the BPM modeling supports the integration of IoT entities into the Enterprise SOA world and into service science [5, 34].

Similarly to the IoT-A project, authors of [44, 2] propose an SSN-based model in which IoT services are provided according to established associations between Physical Entities (PE, namely every person, place, or object whose spatio-temporal attributes and preferences constitute its Context). Differently from business-oriented service model of [1], however, the IoT service model of [44, 2] specifically focus on semantic IoT service description, thus extending the OWL-S (Web Ontology Language for Service) [45]. Indeed, each IoT Service (see Fig2.5(b)) is featured by a ServiceProfile describing what a service does (functional and not-functional properties), a ServiceModel eliciting how a service works (processes and related Preconditions, Effects, Inputs, and Outputs), and a ServiceGrounding specifying how a service is concretely implemented (message formats, serialization, transport and addressing, etc.). In particular, with respect to the original OWL-S service model, ServiceProvisionConstraint, ContextPrecondition and ContextEffect classes have been introduced within the Service Profile to explicitly consider context-awareness and cyberphysicality at the modeling phase. Indeed, the ContextPrecondition class specifies the conditions related to the PE Context (namely its spatio-

temporal features) that should hold before the service can be provided (Precondition specifies just general functional preconditions). Similarly, the ContextEffect class describes changes to the external world or environment (Effect just describes the change to the service provider entity). Finally, ServiceProvisionConstraint class represents PE physical constraints that are relevant to the service provision.

IoT-A like, but not SSN-based, service models are reported in [4, 46] and in [3]. In particular, service model in [4] mainly consists of IoT services and Entities of Interest (EoI), representing physical objects featured by their Properties of Interest (PoI, namely desired properties associated with an EoI) and Devices (their physical interface with the other EoI). IoT services, instead, are featured by a set of Requirements which consider a specific application context, an EoI, its PoI, and PoIs observation rate and provided reliability (as shown in Figure 2.5(d)). IoT service Requirements are specified in a declarative way and can be autonomously processed and matched with the expected levels of dependability. Conversely, as reported in Section 3.2, [3] proposes an application-neutral, UML-like IoT service high-level metamodel. Each Service is described through some textual information (name, a description, service type, input and return parameter types, and zero or more QoS Indicators) and at least one Operation (in its turn, provided by a description, a set of input parameters and return parameters type) which implements the service by defining an individual action that may be invoked. In this service model (as shown in Figure 2.5(c)), Services are provided by SOs and consumed by any kind of IoT entity (human users, conventional digital systems or other SOs).

A completely different approach to service modeling is carried out in [47, 6]. In particular, these models are specifically conceived for operational purposes more than for descriptive goals. Indeed, both works exploit (extensions of) Petri Nets [48] to model real world entities as Nets, their operations as transitions and their IoT services as a sequence of states, as shown in Figure 2.5(f). Such operational modeling allows controlling the correctness of IoT services among dynamic context changes, thus exhaustively and automatically checking their compliance to a given set of specifications.

2.3.1 Analysis and limitations of IoT services specifications

In the previous Section, a general overview of most relevant IoT service models has been provided, along with some insights about entities involved in services provision/consuming. Even at a first glance, it could be noted that IoT service models of Figure 2.5 are conceptually aligned since their main features present marked analogies. Indeed, they include a (mostly) coarse and uniform description of service providers/consumers (i.e., SOs, Participant, Physical/Virtual/Real World Entity, Entity of Interest) and they consider an IoT service not as a monolithic body but as the result of subunits (i.e., Operations, Processes, Tasks,) concretely implementing the service constituent low-level

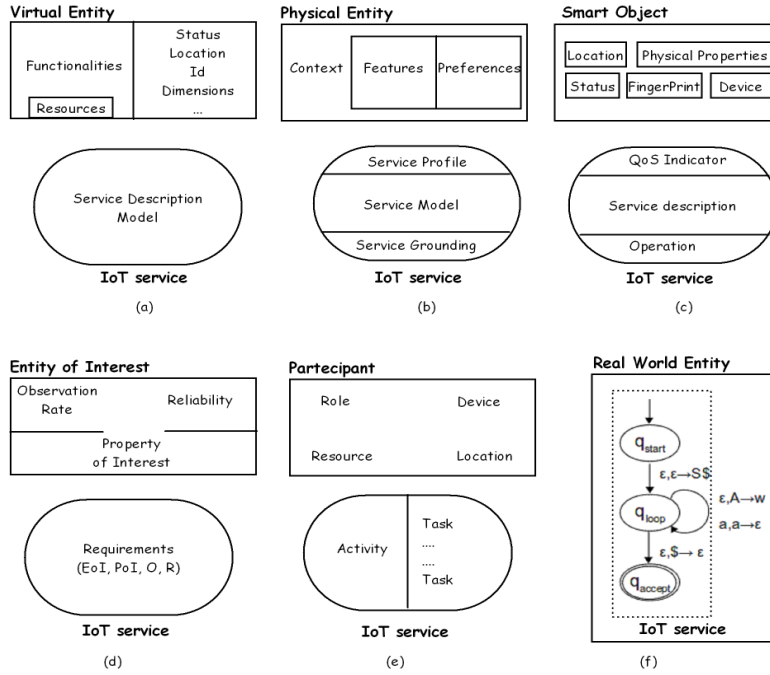


Fig. 2.5. IoT service modeling in (a) [1]; (b) [2]; (c) [3]; (d) [4]; (e) [5]; (f) [6].

actions. Interestingly, service specifications of [1], [2], [3] and [7] show a quasi one-to-one mapping and hence they have been deepened and compared in Table 2.1. Beside their analogies, however, the presented IoT service models share several deficiencies in modeling important IoT services-related concepts, mainly related to IoT entity, context and policy modeling, that are hereinafter discussed along with some examples contextualized in the IIoT scenario [49].

With respect to IoT entity modeling, [5], [6], [4], and [2] have a very coarse-grained entity notion (same representation without any distinctive features among humans, things, and places), considering it as a generic, passive service participant. However, IoT entities are notably heterogeneous and play an active role (both as service providers and/or service users), hence requiring detailed descriptions (partially provided in [1] and mostly in [3]). For example, in a typical production line as the one of [50], it is important to model specifically human operators, machines, robots, conveyors and product parts with their own features to capture not only their static descriptions (e.g., operator id, conveyor length, robots drivers version) but, most importantly, their dynamic status (e.g., busy or idle machines, partially or totally assembled products), physical properties (e.g., conveyors load and speed, machines internal temperature), available devices (e.g., temperature, humidity, pressure, presence sensors, video cameras, hydraulic, pneumatic, electric, and thermal

actuators), etc. In such a way, IoT entities can be better characterized and their interactions and services effectively designed, enabling at the same time a thorough context modeling.

With respect to context modeling, in [5], [2], [4], and [3] the concepts of context and location essentially overlap, while it lacks in the other two service models. However, in order to effectively impact the service provision, any implicit or explicit information related to the entity current/historical situation and the surrounding cyberphysical environment should be taken into account also at a modeling stage. For example, if we consider a chemical factory as in [51], multiple factors linked to different entities (rooms humidity, barrels wear and tear, machines vibrations) contribute to the “stock service” context and affect the production systems. Hence, context should be designed as first class abstraction (including, but not limited to the location concept), so providing also suitable inputs to dynamically choose the best service execution according to the available policies.

Finally, all the surveyed service models are essentially neutral with respect to the concept of policy, that provides macro level specifications and, in general, represents an effective way to adapt dynamic systems. Taking inspiration from the hybrid product-services surveyed in [52] and considering a radiology hospital department, equipment machines embedding both usage and legal health policies could (i) monitor healthcare professionals current exposure to radiations, timely alerting them in case of dangerous situations, and (ii) enable pay-per-use business models and customized risk bonuses that consider the actual healthcare professionals’ use of the equipment machines. Hence, the service model should allow the inclusion of different policies, thereby making the service provision more flexible and adaptive.

In conclusion, although these models notably improved their original versions aiming at an effective IoT service modeling, the aforementioned limitations prevent them for being concretely applied. Indeed, they have been designed considering static environments and generic IoT entities with established interactions, so that service provisioning is typically modified just according to user current position or a sensed phenomenon. However, IoT services would dynamically appear and disappear because their provision may be meaningful only in a certain space/time and it may be heavily subject to heterogeneous constraints and conditions (related, for example, to current user status, context, different policies, etc.). Nevertheless, the aforementioned IoT service models represent a good starting point to open a discussion about a novel yet actually exploitable “Opportunistic” IoT service model, presented in Chapter 3.6.

Table 2.1. Comparison of service specifications featuring [1], [2], [3] and [7]

High-Level UML-like SO service model [3]	Ontology service model extending OWL-S [2]	BPMN 2.0 extension for IoT service [1], [7]
<i>Service</i> : interface to expose SO functionalities	<i>Service</i> : invocable functionality with properties described in the ServiceProfile	<i>IoT Activity</i> : executable compound work
<i>Service Id</i>	<i>Service Name</i>	<i>Activity Name</i>
<i>Service Description</i>	<i>Service Description</i>	<i>Activity Description File</i>
<i>ServiceType</i> : specifies sensing, actuation or computation goal.	<i>ServiceCategory</i> : refers to an entry in some ontology or taxonomy of services.	<i>Activity Type</i> : Sensing and Actuation Activities extend the original Activity set.
<i>Service QosIndicator</i> : specifies a service quality parameter	<i>ServiceParameter</i> : describes the quality guarantees provided by the service	<i>Annotation and IoT Quality metric</i> : measure aspects relevant to the Activity quality
<i>Service InputParameterType</i> : specifies service required data type	<i>ServiceInput</i> : information required for the service execution	<i>Activity Input Data Set</i> : data required to Activity execution
<i>Service OutputParamType</i> : specifies service output data type.	<i>ServiceOutput</i> : information generated as output of the service execution.	<i>Activity Output Data Set</i> : data produced as a result of Activity execution.
NA	<i>ServicePrecondition & ServiceContextPrecondition</i> : functional/context condition required for a valid service execution	<i>Event-, location- and time-based Conditions</i> : evaluated in Gateways to drive Activity execution
NA	<i>ServiceEffect & ServiceContextEffect</i> : change to the service provider entity/environment resulting from the service execution	NA
NA	<i>ServiceProvisionConstraint</i> : physical static PE's constraint relevant to the service execution	NA
<i>Operation</i> : individual operation that may be invoked on a Service.	<i>Process</i> : interaction with the Service defined into the Service Model.	<i>Actuation/Sensing Task</i> : atomic unity of work compounded into an IoT Activity.
<i>Service User</i> : any human, SO or digital system exploiting the SO.	<i>Physical Entity</i> : any entity involved in the Service provision/fruiton.	<i>Participant</i> : Human or any Physical/Digital Entity involved in the Activity.

2.4 IoT ecosystem development requirements

IoT ecosystems are composed of many distributed and interacting components that are usually heterogeneous in terms of hardware devices, communication protocols, software interfaces, data, and semantics. To effectively support their development, general and specific requirements need to be defined [11, 53]. While the general requirements allow effective and flexible middleware [54, 55] for facilitating IoT system/thing programming, the specific requirements are purposely defined for a target IoT system/thing by considering its specific application domain. In the following, we focus on the former that are common to all IoT ecosystems. In particular, we group such requirements in two categories:

- System-level (Table 2.2), which includes requirements related to the whole distributed system and its development, and
- Thing-level (Table 2.3), which encompasses requirements particularly referring to the “things”, such as RFID items, SOs, mobile devices, and robots, in an IoT system.

Requirements listed in Tables 2.2 and 2.3 have been outlined by thoroughly analyzing the state-of-the-art of IoT middleware, architectures and platforms, focusing on their main features and extracting common keywords. Such requirements are not totally new, since they have been already studied in several fields of computer science and engineering. However, at both levels, they recur at the same time and with a substantial prominence within the IoT context and they allow accommodating all the most important features of IoT ecosystems. Indeed, conventional computing devices and everyday things tend to converge in the IoT, requiring virtual networked alias (SLR₁), software interfaces (SLR₃) and communication/data abstractions (SLR₂-SLR₅) to synergistically cooperate, despite their heterogeneity (TLR₁). To cope with such cyberphysical (SLR₄) and dynamic scenario rich in continuously evolving (TLR₄) and augmented (TLR₂) things, proper methodologies are needed (SLR₆) to fully support the IoT system development. Furthermore, decentralized management (TLR₃) mechanisms are essential for making things autonomous and effectively integrated in their application contexts. Finally, at both System- and Thing-level, the cyberphysical nature of SOs introduces important novel elements in the characterization of SO-based IoT systems, particularly with regard to the concept of “scale” (SLR₇ and TLR₅). In traditional distributed systems, the concept of scale is closely related to the number of involved computing nodes, their geographical distribution and logical organization among different administrative domains. Such domains usually have different configurations, policies and privileges, thus emphasizing the need of interoperability and coordination mechanisms [56]. In traditional agent-based systems, the scale concept usually overlaps with agent population [57] and agents distribution among host devices, regardless of their actual geographic location (note that one of the peculiarities of agents is their mobility). Finally,

in WSNs the concept of scale refers both to the number of involved devices and to their spatial collocation, as the radio communications are strongly susceptible to interference and mutual collisions [58, 59]. It is just in the WSN context, indeed, that the concept of density, intended as the number of sensors per unit area, appears [58]. In conclusion, depending to the application contexts, the “scale” term is differently defined as well as its characterizations (large, medium, small scale) can notably vary (a large scale WSNs very likely will differ from a large scale computational grid in terms of geographical extension, population and density). Within the SO-based IoT context, therefore:

- It is handy to refer to well established concepts of “small-medium-large scale” taken from traditional distributed systems, as long as such definitions are not exclusively attributable to geographical factors. Moreover, it is convenient to take into account the network infrastructure, in particular the number of subnets involved, in order to better evaluate system performance; and
- Since SOs are highly pervasive and mostly based on wireless communications, the density issue pointed out for WSNs strongly recurs. Although not only simple sensors but even other kinds of functionally heterogeneous devices are involved within SO-based IoT systems, the density remains a useful metric to characterize scenarios when the number of SOs changes.

On the basis of such considerations, and specifically for an unambiguous characterization of the case studies of Chapter 4, hereinafter we classify IoT systems and SOs in small-medium-large scale on the basis of their physical dimension and density, as shown in Figure 2.6. Similar criteria for scenario characterization are defined in [11, 60].

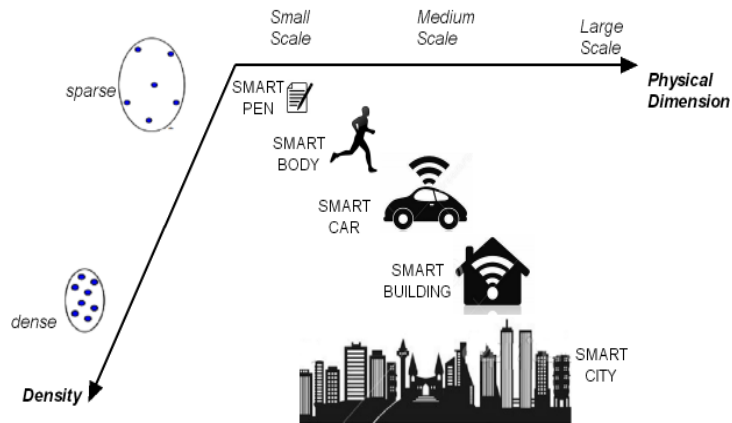


Fig. 2.6. Scale in IoT systems

Table 2.2. System-Level Requirements (SLRs)

Requirement	Description
<i>SLR₁: Hardware Devices (Virtualization)</i>	IoT systems typically comprise heterogeneous devices; in order to facilitate their use, abstractions are needed to virtualize and let them be used, as they are homogeneous by following a kind of a “plug&play” paradigm [11].
<i>SLR₂: Communication (Abstractions)</i>	Software components and devices need to communicate with each other. Communication abstractions are needed to make them interact and cooperate, independently from the available low-level network protocols [8].
<i>SLR₃: Software Interfaces</i>	As software interfaces are usually heterogeneous, they need to be generic and standardized through higher level mechanisms such that their use is straightforward. Thus, software components based on such high-level interfaces can be seamlessly accessed [61].
<i>SLR₄: Physicality (Self and Context Awareness)</i>	Hardware and software components in IoT systems and entire IoT systems themselves are intrinsically situated. This implies that they have static or dynamic locations and refer to one or multiple contexts during their life-cycle. Abstractions are therefore needed to capture the concepts of location and context, as they are useful in the design and implementation of IoT systems [62].
<i>SLR₅: Data (Abstraction)</i>	Different hardware and software components, e.g. sensors, machines, smart objects, and mobile apps, usually produce data according to different modalities, formats and types. Thus, abstractions are needed to formalize data streams generated by such components. Continuous data streams, discrete data and sporadic events should be defined under a common framework. Moreover, the representation of data types needs to be standardized as it would allow interoperability in data exchange among heterogeneous components [63].
<i>SLR₆: Development Process (Methodology)</i>	To analyze, design and implement IoT systems, suitable software engineering methods and tools need to be defined. They should be able to effectively model IoT systems by using high-level modeling abstractions and fully support their design, implementation, deployment and management [13].
<i>SLR₇: Systems Scale Characterization</i>	IoT systems can notably differ in terms of geographical extensions, network infrastructures and number of involved IoT devices. Hence, it is useful to define some criteria to facilitate the unambiguous characterization of their scale and possibly enable their comparison [56].

Table 2.3. Thing-Level Requirements (TLRs)

Requirement	Description
<i>TLR₁: Heterogeneity and Interoperability</i>	Applications that use “things” should be programmed independently from vendors-specific things. For instance, if an application is based on a “smart chair”, it should be able to use smart chairs built by different vendors. Moreover, applications should be able to exploit things to be built in the future. This implies to adopt a standardized approach or, if not applicable (standardization is a very long process), to exploit software layering-based dynamic adaptation techniques between application and the thing level [18].
<i>TLR₂: Augmentation Variation</i>	“Things” usually provide a set of devices and services that can vary in quantity and types both among different things and among similar things. In particular, different things can provide same services whereas two similar things can provide different services. Thus, things cannot be crisply classified only by their type and may expose non-standard interfaces. Augmentation variation of things is an important requirement as it defines how things can modify their augmentation by providing diversified services that can change during their lifecycle. This implies to design not only methods to dynamically add/modify/remove things services and devices but also how they are actually furnished [64].
<i>TLR₃: Decentralized Management</i>	An effective management is crucial in IoT applications where tons of distributed “things” could potentially interact with each other and/or be used to fulfill a final goal. Applications and things should be therefore able to dynamically adapt as things could continuously change for different purposes (augmentation variation, mobility, failures, etc.). Thus, the matching among things services and application requirements should be often done at run-time. Discovery services are therefore strategic in such a dynamic context to find and retrieve things according to their static and dynamic properties [65].
<i>TLR₄: Dynamic Evolution</i>	Applications and “things” should be simply and rapidly prototyped and upgraded through proper programming abstractions. The evolution can be driven by programming, learning, or both. In particular, evolution by learning is usually based on smart self-evolving components able to self-drive their evolution on the basis of some learning models [9].
<i>TLR₅: Things Scale Characterization</i>	“Things” can notably differ in terms of physical dimensions and number of aggregated devices. Hence, it is useful to define some criteria to facilitate the unambiguous characterization of their scale and possibly enable their comparison [57].

2.5 Framework-supported state-of-the-art survey

In this Section, the state-of-the-art of IoT platforms, architectures, middlewares and methodology has been surveyed. Due to their heterogeneity, instead of an exhaustive survey of their features, it is reported how these contributions perform the development phases of analysis, design and implementation, and if they follow a methodological approach. Simultaneously, for each surveyed work, has been reported which of the SLRs and TLRs presented in Section 2.4 it fulfills. The outcome of such state-of-the-art analysis is the comparison framework (split in two parts for the sake of readability) of Tables 2.4 and 2.5. It provides an overview of the current situation of IoT platforms, architectures, middlewares and methodology, highlighting common practices and trends, as well as lacks and limitations. Such insights represented the starting points to set up our development methodology for autonomic and cognitive IoT ecosystems. Furthermore, the comparison framework can be straightforwardly reused to compare future work in the field. To the best of our knowledge, this is a novel research contribution in this context.

2.5.1 Analysis phase

Goal: identifying the main entities of the IoT and abstracting their basic features and high-level interactions. This is a preliminary phase often exploiting metamodels to provide expressive, but not too much complex, high-level representations which can be handled by IoT developers with different expertise.

The IoT-A project [66] envisages a technological-agnostic and application-neutral reference model architecture to be further specified in different domains (information domain, functional domain, etc.) according to developer needs. It is centered around the concept of an “IoT Entity”, namely an augmented physical entity hosting some “IoT devices” (e.g., sensors, actuators, tags, TLR₂), that is virtualized through a “Virtual Entity” (SLR₁) for exposing its “Resources” to the “Users” through some “Services”. On the basis of these concepts, IoT-A project provides different models that can be used to derive a concrete IoT architecture following specific computing paradigms. By means of different views, perspectives and metamodels, IoT-A aims to offer a unified approach to the analysis of IoT systems, in order to promote cross-domain interaction (SLR₂), to support interoperability (TLR₁) and to reduce fragmentation within an IoT context. Most of the indications provided by IoT-A have inspired the AIOTI (Alliance for the Internet of Things) [38], and the IEEE P2413 [40] reference models, specifically their domain models. Indeed, they describe IoT entities and their relationships through common definitions and recurrent building blocks’ high-level metamodels, with the final goal of promoting a unified approach to the development of IoT systems.

In [67], it is proposed a metadata model to represent functional and non-functional characteristics of SOs in a structured way (SLR₅). The metadata

model is divided into four main categories: Type, Device, Service, and Location. The Type is the SO type (e.g., smart pen, smart table, smart space); a secondary type can also be given that contains information about the SO design classification as proposed in [9], (TLR₂). The Device defines the hardware/software characteristics of the SO device (e.g., sensors, actuators, computing units) that allow to augment and make “smart” the object. Services contain the list of services provided by the SO; in particular, an SO service can have one or more operations implementing it. Lastly, the Location represents the position of the SO (SLR₄) and it can be set in absolute terms, specifying the coordinates (latitude and longitude), and/or in relative terms through the use of location tags. Because of its generality (SLR₁), this metadata model can characterize an SO in any application domain of interest and has been extended and re-organized in [68, 69], and it represents the starting point of the ACOSO-Meth High Level Metamodel presented in Section 3.2.

Finally, [70] presents an high-level models for interacting SOs, namely digital representation of physical artifacts (SLR₁), augmented with sensors, actuators, processing and networking units. Artifacts have Properties which represent physical characteristics, capabilities, and services and are modeled as a first-order logic function, while a snapshot of all such properties at a given time constitutes the artifact’s State. Plugs are the input/output interfaces of an artifact, exposing its featuring properties (output plug) or indicating its requirements (input plug) for enabling certain functionalities (SLR₃). When two plugs are compatible, they are associated through Synapses, namely logical communication channels (SLR₂) between the nodes of the distributed system, while two or more artifacts (simple or composite) can be combined in an Artifact Composition (TLR₁, TLR₄) in order to provide a complex service.

2.5.2 Design phase

Goal: modeling the functional components of the system, their specific relationships and interactions. Differently from the analysis phase, the design phase focuses on identifying suitable paradigms (supported by well-defined semantics) and enabling mechanisms, but without being coupled to a specific technology or any implementation detail.

In [71], the actor-oriented paradigm inspired design models which deeply describe cyberphysical systems in terms of their concurrent and parallel interactions, with a particular attention on the intrinsic complexity, heterogeneity and sensitivity to timing (TLR₁). Such aspects can be effectively modeled thanks to the implicit actors’ semantics (SLR₅) that allows a fine-grained design of IoT entities and of both software processes (deeply rooted in sequential steps, SLR₁) and physical processes (by contrast, rarely procedural, SLR₄). Likewise the actor-oriented paradigm, also the ABC represent a suitable paradigm for supporting the design of main SOs features abstracting them from low-level details or specific implementation constraints, as reported in Section 2.2.1. Indeed, agent-based design models allows capturing key charac-

teristics of SOs and IoT systems, at different degrees of granularity and in a technology-agnostic way, because strong conceptual relation exists between agents and SOs, as well as between MAS and IoT systems. In particular, SO autonomy, proactiveness and situatedness are implicitly embedded in the agent model, while other important SO features can be explicitly described through agent-related concepts. This is the case of [72, 73, 74], and [75], which express SO functionalities in terms of goals, SO working plan in terms of behaviors, and SO augmentation-related components (like knowledge bases, sensors and actuators) in terms of dynamically bindable agent resources (TLR₂). However, these works adopt different mechanisms for specifically characterizing SOs/agents. In particular, in [76, 77] each agent/SO has a role (taken from a scenario-dependent repository, e.g., smart car, smart driver-support or smart road for the transportation context) that determines, by default, its own behaviors, goals and communications paradigms; similarly, in [78], SO/agent plans and goals are encoded in templates reflecting their functionalities. Other contributions do not reference a-priori defined roles or templates (TLR₄). For example, in [79], each agent/SO has a self-model (an automaton) driving its actions according to stimuli (modeled as messages) from other agents or the environment. Similarly, in [80, 81], SOs actions/reactions are encoded in behaviors, driven by incoming (internal/external) events and design goals (encapsulated in state-based tasks). Finally, in [72], SO/agent self-state is dynamically determined by combining its real-time sensor data, position and status of computational units. Autonomous and cognitive properties can be successfully instilled from the design phase by properly exploiting the ABC paradigm. For example, self-steering agentified SOs play a crucial role in I-Core [78], a cognitive management framework that gives a three-layered architectural model. At the lower level, there are real world objects (RWOs) and their digital cognitive counterparts, called virtual objects (VOs, SLR₁). VOs virtualize RWOs functional and non-functional features. They are self-managed entities and can be dynamically created, destroyed, modified or aggregated at the intermediate level (TLR₃, TLR₄). By matching application requirements and single VO capabilities (exploiting pattern recognition and machine learning techniques), the need of composite virtual objects (CVOs) arises. CVOs are sophisticated entities that represent cognitive mashup of semantically interoperable VOs, aiming at the development of situation-aware, user-tailored and proactive services at the top-level. Similarly, [82] presents a comprehensive definition of CIoT (Cognitive Internet of Things) and hence a layered framework conceived to implement five cognitive tasks, namely environment perception, data analytics, semantic analysis/knowledge discovery, intelligent decision-making, and on-demand service provisioning. Again, physical/virtual things are abstracted into agents that, with minimum human intervention, interact with each other enabling a smart resource allocation, automatic network management and intelligent service provisioning (SLR₁, TLR₁, TLR₃, TLR₄). Finally, Cascadas [83] (Componentware for Autonomous Situation-aware Communications, and Dynamically Adaptable Services) presents a high-level ar-

chitecture centered on the notion of autonomic elements (AEs) and designed on the principles of situation awareness, semantic self-organization, interoperability, scalability and modularity (SLR₁, SLR₃, SLR₃). Each AE is developed following an agent-oriented approach and, by means of a shared communication interface, of a conventional passing messages paradigm, and of a set of a software plugins for the data-format conversion (SLR₂, SLR₅), it provides adaptive, composite and situated communication intensive services.

2.5.2.1 Simulation-based Design

Simulation-based design (SBD) is a design approach focusing on simulation as the key-enabler for evaluating, validating and optimizing design choices or design alternatives. It exploits models, simulation tools, and techniques, starting at earliest conceptual design phases, to support developers in making informed design choices [84]. In particular, simulation allows validating design choices and discloses unexpected behaviors before actual system deployment, that is often time-consuming, costly and error-prone. Particularly in the IoT context, where interactions are subject to variety of contingent factors (e.g., SOs density, physical network design, traffic congestion, wireless signal attenuation and coverage), being able to simulate the system have a paramount importance [35]. In fact, it allows understanding overall dynamics, estimating performance, and validating models, protocols and algorithms featuring under-development SOs and IoT systems. However, no IoT-specific simulators are currently available.

Agent-based simulators, being centered on high level agent abstraction, do not directly address issues that characterize SO-to-SO interactions in a physical environment (such as limited computational and energetic resources, network congestion due to SO density, interference obstructing wireless communications, and so on). Therefore, agent-based simulations have been exploited to inspect high-level issues, such as increased collective dynamics and behavioral patterns in IoT systems, assuming that agents are deployed in aseptic environments without any connectivity issues [85] (SLR₁, TLR₁). In contrast, conventional network simulators allow a contextualized and detailed management of low-level communication features [75] (SLR₄). Specifically, such simulators allow validating network design choices and deeply analyzing network performance, but they are usually exploited in application agnostic scenarios [86]. To exploit the benefits of both approaches, network and agent-based simulators have been jointly exploited, but the actual validity [87] of such combination is hard to verify. Nevertheless, considering that IoT ecosystems deals with computation, networking, and physical dynamics, the hybrid solution is anyway considered the most suitable simulation approach [35].

2.5.3 Implementation phase

Goal: actually realizing the outcome of the design phase by means of specific programming paradigms, adopting well-established specifications and development tools.

Because of deep heterogeneity of resources and communication protocols in the IoT context, many authors propose SOA and REST for making SOs functionalities accessible under the form of Web Services over standard Internet protocols, which are platforms and programming languages independent [88, 89, 90] (SLR₁, SLR₂, SLR₅). Conversely, there is a wide research line proposing an agent-oriented approach for programming uniform interfaces and thus transparently interacting with resources and SOs. Authors of [76, 77, 80, 78, 91, 70, 90] exploit software adapters (developed for specific technologies and somehow internally coordinated, e.g., through a device manager, as in [80]) for accessing agent/SO augmentation devices (SLR₃). This approach improves modularity and extensibility (TLR₂), since it leverages pluggable software components that can be defined when needed, and customized within the target resource. Instead, [73, 74, 79] follow a different approach: each resource is directly coupled with one agent that interfaces the resource itself with the related SO, or with rest of the system. This solution completely hides the underlying technological heterogeneity, but it is not suitable for such constrained devices that cannot support an agent-based architecture (SLR₃, TLR₁). Apart from resource handling, agent-oriented programming contributes to overcoming lack of communication/coordination standards (SLR₁, SLR₂, SLR₅) within the IoT arena:

- by implementing the IEEE FIPA “de facto” standard specifications [23], which standardize both message format (specifically, the Agent Communication Language [92], ACL, is used for encoding message envelope) and message content (whose concepts, typically expressed through metadata-oriented languages, refer to ontology for facilitating data and the context management), as well as provide effective message transport service (leveraging on both semi/centralized and distributed services of agent discovery); and
- by supporting the SOs virtualization [78, 15, 79], thus paving the way towards integration of self-steering agentified SOs within the Cloud. In such a way, by outsourcing computation/storage resources, the SO hardware/software limitations are mitigated and complex analytics services can be provided even at the edge of the network [93].

With regard to the last point, relevant contributions are provided by [94, 90]. In [94] authors present the Cloud-assisted Agent-based Smart home Environment (CASE) platform, which allows the distributed sensing and actuation in Smart Home environments (SLR₄, TLR₃). The CASE platform presents a three-layered architecture that exploits the distributed multi-agent paradigm and the cloud technology for offering analytics services. In the CASE

architecture, agents are disseminated both on local devices and at the edge (performing in-node feature extraction, activity discovery, activity recognition) and on the cloud (defining articulated strategies of actuation or executing complex algorithms). Such architectural design allows a real-time event processing since the computation is close to the information sources, an efficient communication by propagating across the system only aggregated information, and an increase of reliability and scalability through the use of (even complex) distributed algorithms. A virtualization framework using the sensor-as-a-service notion to expose IoT clouds connected objects functional aspects in the form of web services is presented in [90]. In detail, the framework exploits an event-driven service oriented architecture (e-SOA) paradigm along with a set of policy-based service access mechanisms based on ontology and semantic rules (SLR₃, TLR₃). The goal of the virtualization layer is to expose the functional aspects of underlying IoT cloud and information in the form of services (TLR₁, SLR₁). The architecture is composed of three layers: (i) the Real-world access layer (gets real-world information and carry it to the upper layer for further processing, adopting an adapter oriented approach to address the technical diversity regarding sensor types and communication mechanisms), (ii) the Semantic Overlay layer (provides the semantic model of underlying IoT cloud, IoT/sensor/event ontology and policies), and (iii) the Service Virtualization layer (expose the functional aspects of underlying IoT cloud and information in the form of services).

2.5.4 Development Methodologies

Despite a variety of research efforts that tackle different specific issues within an IoT system development process, a full-fledged IoT methodology is missing. There are many studies which, instead of providing a proper methodology, collect domain-specific best practices, guidelines, checklists and templates. For example, Slama et al. [95] and Collins [96] build up a repository of technology-dependent solutions coming from the experience in the industrial/business world and specifically directed to the IoT makers and enterprises. In fact, they propose reference architectures and guidelines to make specific-purpose devices interoperable (TLR₁) through abstract data models (SLR₅) and high-level software interfaces (SLR₃). Differently, some researchers present general-purpose approaches. IoT-A [66] is a systematic collection of architectures, reference models, common definitions and guidelines that can be used to derive a concrete IoT architecture. By means of different views, perspectives and metamodels, IoT-A aims to offer a unified approach to the development of IoT systems, in order to promote cross-domain interaction (SLR₂), to support interoperability (TLR₁) and to reduce fragmentation within an IoT context. Zambonelli [97] proposes a software engineering methodology centered on the main general-purpose concepts related to the analysis, design and implementation phases of IoT systems and applications. Such concepts are used to identify the key software engineering abstractions (SLR₁, and SLR₃-SLR₅) as well as

a set of guidelines and activities that may drive the IoT systems development. The envisioned methodology, however, lacks the definition of models and tools to represent different conceptual and software artifacts. In tandem with these contributions, some agent-oriented methodologies have been specifically extended for the IoT context [98, 97], and [99]. Indeed, beside disciplining the exploitation of agent-based suite of models, programming techniques and simulation tools, these work have been extended for dealing with requirements that are typically overlooked by agent-based methodologies, such as (i) the cyberphysical nature of the involved entities and environments, foreseeing by design, solutions for interoperability, security and scalability [98, 99] (SLR₄); (ii) the identification of IoT users and stakeholders, depicting significant use cases through textual descriptions [97] and technical notations, like UML (Unified Modeling Language) [100] or BPMN (Business Process Model Notation) [43], for meeting different expertise and perspectives (SLR₃, TLR₁, SLR₅); and (iii) the analysis of the infrastructural features and limitations according to the specific IoT system requirements [98], since these factors cannot be considered independently. Without extensively dealing with all these factors, even effective and well-known conventional agent-based software development methodologies like Tropos [101] are definitively inadequate, and thus unable to actually unfold the full IoT potential.

2.6 Comparative Analysis

Contributions surveyed in the previous sections are compared in Table 2.4 on the basis of the development phases they (totally, partially or do not) support. In particular, Table 2.4 shows if each single work provides fine or coarse grained models for the analysis phase, if it supports the design phase (and, eventually, a simulated-based design phase), if it has been concretely implemented, and if these phases have been driven by a development methodology. Instead, Table 2.5 shows if the surveyed works (totally, partially or not at all) fulfill the SLRs and TLRs presented in Section 2.4. By examining the reported tables it should be noted that:

- with respect to Table 2.4, analysis and simulation-based design phases are mostly overlooked, despite of their importance for the further implementation. Even though IoT is a well-established research area, interestingly, these two aspects haven't been deeply investigated. Moreover, none of the surveyed methodology seamlessly support the different phases of the development process. Indeed, they individually present guidelines or best practices, models and design techniques, but without including them into a comprehensive and generic approach.
- with respect to Table 2.5, the surveyed works neither completely support the TLRs and SLRs, nor cover the entire development process. Indeed, they tend to differently address specific issues in particular application

contexts. In particular, both the thing’s and system’s scale characterization are systematically unfulfilled requirements.

- because of the entire set of requirements and issues related to the development of IoT systems, ABC gained consensus a suitable paradigm for modeling, programming and simulating IoT ecosystem. Indeed, the agent-oriented view of the world is perhaps the most natural way of approaching several types of (natural and artificial) systems, featured by a relevant complexity, dynamicity, situatedness, and autonomy, and, in particular, strong conceptual relation exists between agents and SOs, as well as between MAS and IoT systems.

Such considerations have been carefully taken into account in our development methodology for autonomic and cognitive IoT ecosystems. Indeed, tacking advantage from the lessons learned from the state-of-the-art analysis, a full-fledged, agent-oriented, and metamodel-based engineering approach has been developed, reserving equal importance to the analysis, design (and simulated-based design), and implementation phases.

Table 2.4. Related work’s support to the development phases (Y = totally supported, P = partially supported, Blank = not supported)

	Analysis		Design (* if SBD supported)	Implementation	Methodology
	Fine	Coarse			
[66]	Y				Y
[38]	Y				
[40]		Y			
[67]		Y		Y	
[78]	Y		Y	Y	
[70]		Y	Y	Y	
[71]			Y	Y	
[76, 77]			Y	Y	
[80]			Y	Y	
[81]			Y	Y	P
[102]			Y	Y	
[75]			Y*		
[85, 86, 87]		P	Y*		
[83]			Y	P	
[91]		P	Y		
[73, 74]			Y	Y	
[79]		P	Y	Y	
[94, 90]			Y	Y	
[88, 89]			Y	Y	
[98, 97, 99]			Y		Y
[95]		P			Y
[96]		P			Y

Table 2.5. Related work’s fulfillment of development requirements (Y = totally fulfilled, P = partially fulfilled, Blank = not fulfilled)

	System-Level Requirements (SLRs)							Thing-Level Requirements (TLRs)				
	SLR ₁	SLR ₂	SLR ₃	SLR ₄	SLR ₅	SLR ₆	SLR ₇	TLR ₁	TLR ₂	TLR ₃	TLR ₄	TLR ₅
[66]	Y	Y		P	P	Y		Y	Y		P	
[38]	Y	Y		P	P			Y	Y			
[40]	Y	Y			Y			Y		Y		
[67]	Y			P	Y			P	Y	Y		
[78]	Y	Y	Y	P		P		Y	Y	P		
[70]	Y	Y	Y					Y			Y	
[71]	Y			Y	Y			P	P			
[76, 77]	Y	Y			Y	P		Y		Y	P	
[80]	Y	Y	Y	P				Y	Y		Y	
[81]	Y	Y	Y	P	Y	P		Y	Y		P	
[102]	Y	Y		P	Y	P		Y		P	P	
[75]	Y		Y	Y	Y			Y		P		
[85, 86, 87]	Y	P	Y					Y				
[83]	Y	Y		Y		P		Y		Y	Y	
[91]	Y	Y	Y	P		P		Y	P		P	
[73, 74]	Y		Y	P	Y			Y	P			
[79]	Y		Y	Y		P		Y		Y	Y	
[94, 90]	Y	Y	Y	Y				Y	P			
[88], [89]	Y	Y	Y		P			Y		P	Y	
[98, 97, 99]	Y		Y	P	Y	Y		Y				
[95]			Y		Y	Y		Y	Y			
[96]			Y		Y	Y		Y	Y			

2.7 Summary

The development of IoT ecosystems and related services is a complex process featured by several and heterogeneous requirements, thus demanding for the joint exploitation of different computing paradigms. In the past years, several IoT architectures, service models, and methodologies have been proposed in the literature and have been surveyed in this chapter. These solutions face notable challenges such as physical device virtualization, decentralized and autonomous management, guideline identification, but they tend to tackle different specific requirements, typically one at a time, without providing a full-fledged approach to support the entire IoT system development process, from analysis to implementation. Indeed, the main outcome from the study of the state-of-the-art is the lack of a complete IoT development methodology specifically tailored on the distinctive requirements of IoT ecosystems and related services.

A methodology for the development of autonomic and cognitive Internet of Things ecosystems

The development of IoT ecosystems and services, their management as well as their integration in real applications, are complex and challenging, thereby requiring suitable models, methods, techniques and technologies.

This chapter describes *ACOSO-Meth*, a novel application domain-neutral, agent-oriented, and metamodel-based methodology that completely supports the development of autonomic and cognitive SO-based IoT ecosystems as well as of novel “Opportunistic” IoT services.

3.1 Introduction

The analysis of the state-of-the-art of the IoT domain has highlighted that the IoT ecosystem development is a very complex and articulated process, since the fulfillment of specific requirements (both at System- and Thing-level) is necessary to enable dynamic cooperation among cyberphysical SOs over heterogeneous IoT systems. Hence, using an engineering methodology is widely recognized as a fundamental practice because the manual and non-systematic application of complex techniques, methods and frameworks would very likely reduce effectiveness, increase development time and tend to be error-prone. In such direction, middlewares represent suitable solutions to speed up system development and prototyping, as well as management and evolution. In particular, the *ACOSO* middleware [80] provides agent-oriented modeling and programming techniques for effectively realizing advanced SO-based IoT systems in any IoT application context, fulfilling both Thing- and System-level requirements identified in Section 2.4. Just the exploitation of the ABC paradigm is crucial to straightforwardly instill smartness and autonomy within a single SO and realize cognitive and autonomic IoT systems [103]. On the basis of these considerations and in order to support the SO analysis, design and implementation phases, the *ACOSO-Meth* (Agent-based COoperating Smart Objects Methodology) has been developed [3, 104]. It integrates within a comprehensive methodology the agent-oriented modeling and programming

techniques provided by the ACOSO middleware with a set of metamodels placed at different abstraction levels and completely decoupled from any specific application context.

As showed in Figure 3.1 (diagrams are compliant with Object Management Group (OMG) Software Process Engineering Modeling (SPEM) 2.0 [105]), ACOSO-Meth supports the analysis phase through a high-level model describing main basic SO features. Such model is specialized and better detailed, thus evolving in the design and implementation phases. In particular:

- at the Analysis phase, a High-Level SO Metamodel is exploited;
- at the Design phase, an ACOSO-based SO Metamodel specializes the analysis-level metamodel in order to model the functional components of the system, their relationships and interactions. Such metamodel can be also mapped on the OMNeT++ platform [106] and simulated for having immediate feedback on design decisions; and
- at the Implementation phase, a JACOSO (JADE-based ACOSO) Metamodel specializes the ACOSO-based SO Metamodel with respect to a particular implementation based on the JADE platform [107].

Every phase introduces new features and a higher degree of detail in the metamodels, maintaining at the same time strong relations with the higher-level metamodels. This allows the straightforward transition from the analysis to implementation phases, seamlessly supporting the translation of high-level system models into design-level agent-oriented platform-independent models that, in turn, may be refined into agent-oriented implementation platform-dependent system models.

Finally, our definition of “Opportunistic” IoT service has been provided and an extension of the ACOSO-Meth has been proposed, presenting a novel and full-fledged approach to IoT service modeling. In particular, the High-Level SO Metamodel has been extended to support IoT service development by means of (i) general purpose descriptive metamodels, suitable for the service analysis; and (ii) detailed operational models, instantiated over specific domains or case studies, suitable for service implementation and verification. Due to such features, the proposed descriptive and operational models meet the desiderata emerging from the “Opportunistic” IoT service definition as well as the skills of the different professionals involved in its development.

3.2 Analysis phase

The metamodel portrayed in Figure 3.2 is a very high level metamodel, since its components may characterize an ecosystem of SOs [67] in any application domain, e.g., smart cities, smart factories, and smart homes. In fact, it models the main aspects of a generic SO/SO ecosystem in a very straightforward way, sharing similar characteristics with IEEE P2413 [40], AIOTI [38] and IoT-A

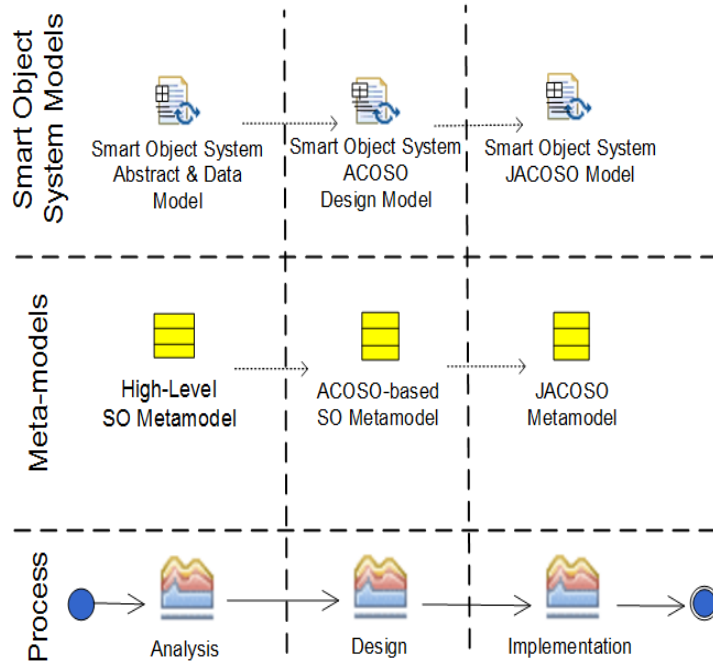


Fig. 3.1. Relationships among ACOSO-Meth metamodellers at different phases

[108] reference models surveyed in Section 2.5. As matter of fact, main coarse-grained SO concepts (namely SO physical/virtual representation, SO user, SO service, and SO device) recur in all the aforementioned models, as well as in the High-Level SO Metamodel, as shown in Table 3.1. To fully support the SO analysis phase, ACOSO-Meth High-Level SO Metamodel exposes further features, reported by means of a UML class diagram in Figure 3.2. These features describe both static (e.g., SO id, creator) and dynamic (mainly related to the services provided, e.g., quality-of-service indicators) SO characteristics.

Table 3.1. Comparison of main entities of SOs metamodellers of ACOSO-Meth, IEEE P2413, AIOTI and IoT-A.

ACOSO-Meth High-Level SO MM [3]	AIOTI SO MM [38]	IoT-A SO MM [66]	IEEE P2413 SO MM [40]
SO	Virtual Entity	Virtual Entity	Virtual Entity
SO Physical Properties	Thing	Physical Entity	Physical Entity
SO Device	IoT Device	Device/Resource	IoT Device
SO Service	IoT Service	Service	N/A
SO User	User	User	User

They are categorized in five main groups:

- *SO BasicInfo* comprises basic SO information. In detail, the Status contains a list of variables, given as pairs \$name, value\$, that capture the SO state; Location represents its geophysical position (expressed in absolute terms by specifying latitude and longitude and/or in relative terms through the use of location tags); PhysicalProperty describes a physical property of the original object without any hardware augmentation and embedded smartness (it contributes to determining its scale); FingerPrint comprises immutable SO information like the SO identifier (or Id, which allows its unique identification within an IoT system), SOCreator that creates the SO for personal use, business or research purposes, SOType represents an SO type, e.g., a smart pen, smart building, and smart city, and QoSParameters defines one or more QoS parameters associated to the SO, e.g., precision, reliability, and availability.
- *SO Service* models a digital service provided by an SO. Each service is characterized by a name, description, type (e.g., sensing and actuation), input parameter type and return type. Each Service is implemented by one or more Operations and by zero or more QoSIndicators whose associated values are provided. In detail, an Operation, which defines an individual operation that may be invoked on a service, has a description, a set of input parameter types necessary for its invocation, and a return type related to its output value.
- *SO User* identifies an entity using the services provided by an SO. In particular, SO Users can be Humans (representing the classical man-machine use relationship), Smart Objects (representing a less conventional use relationship, in which SOs take advantage of services exposed by other SOs and vice versa) or Digital Systems (representing a generic digital entity, like a web server, software agent, robot or a more complex system).
- *Augmentation* defines the hardware and software characteristics of a device that allows augmenting the physical object and making it smart. A device can be specialized in one of the following three categories: (i) Computer, which represents the features of a processing unit of the SO, e.g., PC, smartphone, and embedded computer; (ii) Sensor, which models the characteristics of a sensor node of the SO; and (iii) Actuator, which models the characteristics of an actuator node of the SO.
- *SO Aggregation* supports aggregation among SOs. In particular, a complex SO (e.g., a Smart City) may physically or logically aggregate other SOs to provide more advanced and integrated services.

3.3 Design phase

A High-Level SO Metamodel at the analysis level is refined to obtain an ACOSO-based SO Metamodel (Figure 3.3), which allows, at the design level,

tify information/error messages produced by the SO sensors, actuators, etc.; and (iv) ServiceEvent, raised from internal, external or device event sources, which specifically drives UserDefinedTasks to define application-oriented functionalities. The ACOSO-based SO metamodel components (Figure 3.3) are categorized into four main groups:

- *SO Basic Info* is spread between the SO itself and KBManagementSubsystem (KMS). The latter handles information pertaining its global current state, inference rules and other useful data that can be shared among tasks.
- *SO Service* provided by SOs is encapsulated in specific application-level UserDefinedTasks. They are highly customizable, easily programmable, and interact with other SO components through ServiceEvents.
- *Augmentation* is handled by the DeviceManagementSubsystem (DMS), which manages sensors, actuators and devices embedded into the SO. Interactions between SOs and such augmentation devices, regardless of their specific technology or protocol, are conducted through DeviceEvents.
- *SO Communication* is handled by the CommunicationManagementSubsystem (CMS), which provides a common interface enabling communication toward the SO itself (through InternalEvents) or toward external entities (by means of ExternalEvents).

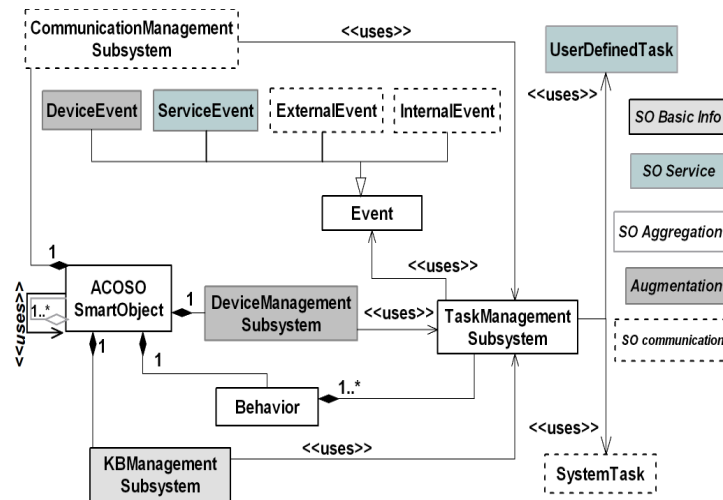


Fig. 3.3. Design Phase: ACOSO-based SO Metamodel

3.3.1 A Hybrid Agent-Oriented Simulation-based Design Approach

Computation, networking, and physical factors equally contribute to IoT ecosystems' dynamics: as a consequence, simulation approaches that address

only the concerns of software are inadequate. Thus, for simulating an ACOSO-based IoT system, it has been chosen an hybrid agent-oriented approach [109], [110] based on the ACOSO middleware and on the INET extension for the OMNeT++ network simulator. OMNeT++ [106] is an open source discrete event simulation platform that mainly aims to simulate communication among entities of distributed computer systems. OMNeT++ is fully programmable and modular, and it follows a reusable, component based approach to build up complex and customizable network scenarios. An OMNeT++ node is composed of multiple compound and simple modules interacting through the message-passing paradigm, while settings, properties, and data can be retrieved from different configuration files. Several extensions can be integrated to simulate specific typologies of networks. In particular, INET is an extension suite that introduces specific protocols and models for WSNs [111]. An INET compound module representing a generic wireless node comprises modules that are grouped in packages, organized according to a simplified ISO OSI model, and interacting through cMessages (a class of message objects representing events, messages, or jobs in a simulation), as depicted in Figure 3.4. In addition, different device boards can be plugged at the physical level to manage INET node components ranging from wired/wireless physical interfaces and radio antenna to sensors and actuators. SO features can be equivalently

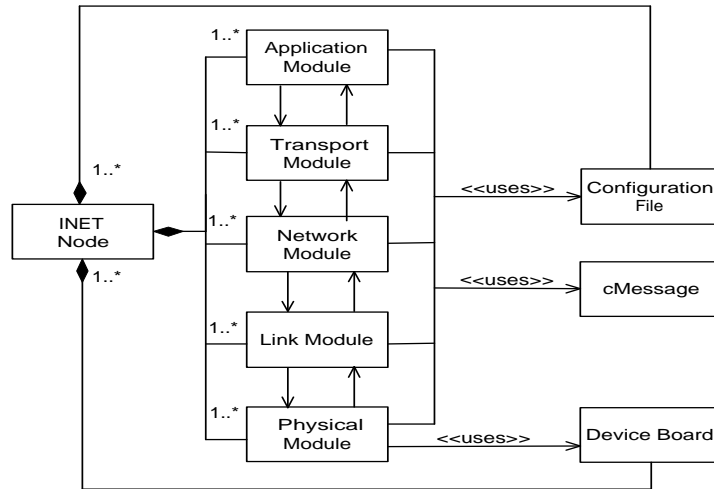


Fig. 3.4. INET node metamodel

represented via the ACOSO-based SO and the INET node metamodels, as depicted in Figure 3.5. However, the transition from an agent-based SO to an INET node must be guided by a preliminary mapping phase. Table 3.2 shows these guidelines for the transition from modeling to simulation for IoT system designers; the definition of automatic translation rules is beyond this thesis

scope but it is a future work (an automatic transition, whenever possible, may speed up the ACOSO-Meth application but keeping the same effectiveness). In detail, both ACOSO-based SOs and INET nodes are state-based entities whose evolution and interactions are driven by messages (ACOSO Events and INET cMessages, respectively) flowing among their components. In both cases, these components constitute modular and versatile architectures and allow the implementation of SO-based IoT systems of different scales in different application scenarios by just reusing or re-implementing some pluggable blocks. In particular, specific SO services can be implemented via UserDefined-Tasks within ACOSO-based SOs as well as through application level modules in INET nodes. New SO communication protocols can be introduced by developing related CommunicationAdapter in the ACOSO CMS or by adding a new transport/network/link module to the INET node stack. Likewise, SO sensing/actuation devices can be managed through the ACOSO DMS and its DeviceAdapters or through specific device boards connected to the INET nodes physical level. Finally, SO data coming from computation or communication activities as well as SO configuration setups and parameters are managed by the ACOSO KMS while they are stored and queried into configuration files in the INET node. In brief, simulating an ACOSO SO in OMNeT++ requires a minimal effort since the similarities between SOs/agents and OMNeT++ network nodes is straightforward. Indeed, only the SO application logic needs to be defined in terms of OMNeT++ application modules, while issues related to the transport, network and physical layers (including challenging aspects like the node energy management, the physical modeling of obstacles that interfere with wireless signal transmission, etc.) are directly managed by the simulator.

3.4 Implementation phase

To obtain the metamodel for supporting the implementation phase, the ACOSO-based SO metamodel has been implemented by using the JADE platform [107]. JADE has been selected mainly for the following reasons: (i) it is an FIPA-compliant, well-known and Java-based agent middleware; (ii) it is open-source, has a spread community and, over the years, has evolved (e.g., JADEX [112], JADE-LEAP [113]) to run atop heterogeneous computing systems such as Java Micro Edition-enabled and Android-supported devices, as well as on sensor nodes constituting heterogeneous WSNs (Figure 3.6); (iii) its middleware provides an effective agent-oriented management/communication infrastructure, that comprises an Agent Management System (AMS), ACL-based message transport system and Directory Facilitator (DF). In particular, DF supports agent service discovery, and has been extended with an agent-oriented interface [65] to allow SOs registration, indexing, and searching on the basis of their functional and/or non functional features (e.g., Location, FingerPrint, and provided Services) reported in the High-Level SO metamodel

Table 3.2. Mapping guidelines

SO functionality	ACOSO-based SO	INET node	Rationale
Service provision	Task Management Subsystem (TMS)	Application level module	Application logic defining specific SO services can be implemented within ACOSO UserDefined-Tasks coordinated by the TMS or within INET modules located at the application level
Communication	Communication Management Subsystem (CMS)	Physical to transport level module	Communication with SO itself or external entities carried out through ACOSO events managed by the ACOSO CMS (and its Communication-Adapters) or by INET cMessages among modules of every level
Sensing / Actuation	Device Management Subsystem (DMS)	Physical level module	SO sensing/actuation devices can be managed by the ACOSO DMS (and its DeviceAdapters) or at physical level module in the INET node
Data management	Knowledgebase Management Subsystem (KMS)	Configuration files	SO data can be stored and queried through database managed by the ACOSO KMS or in configuration files in the INET node

of Section 3.2. These features are represented through metadata descriptions in a JSON [114], which is lightweight format, easy to read and to manually/automatically written, analyzed and generated. Indeed, differently from general-purpose JADE agents, SOs have a strong situatedness and may seamlessly appear and disappear, but they may also evolve on the basis of some learning models or extemporary interactions with other SOs. An enhanced Directory Facilitator, providing a dedicated and dynamic SO discovery service, is thus fundamental. The metamodel shown in Figure 3.7 refers to JADE-based implementation of the ACOSO-based SO metamodel, named JACOSO. Considering the inheritance relationship from the ACOSO-based SO model and

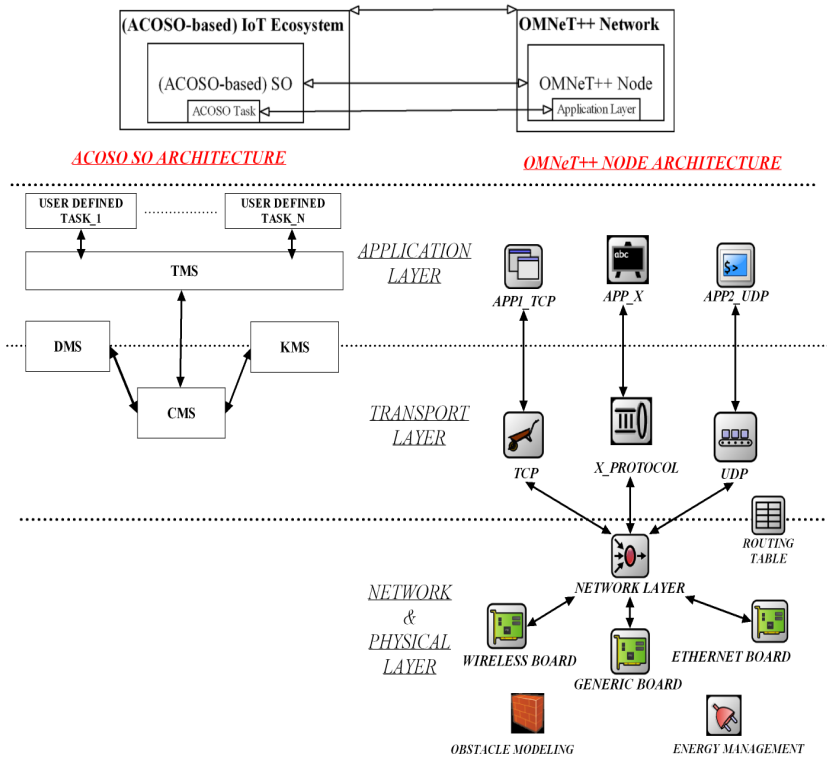


Fig. 3.5. Layered view of (a) the ACOSO SO Architecture and (b) the OMNeT++ Node Architecture

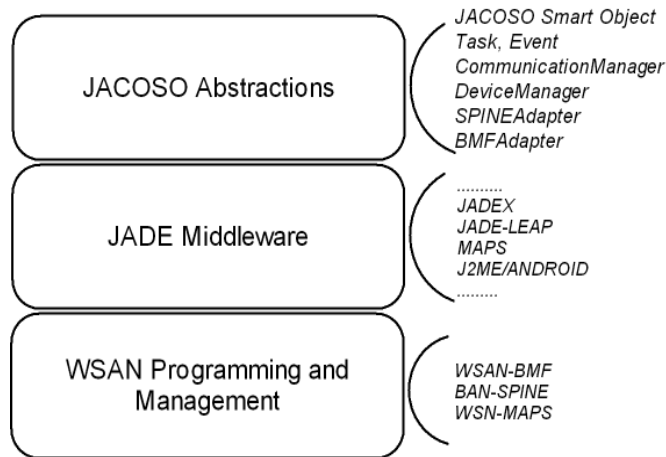


Fig. 3.6. JACOSO three-layered architecture

the JADE components, hereinafter we present only the implementation components that characterize the JACOSO SO metamodel with reference to the related macro-components:

- *SO Basic Info* is spread between the JADE-based agent itself and an internal knowledge base. The latter contains also the current values of the variables constituting the inference rules required for an SO decision-making process. Information, inference rule variables and configurations that need to be provided at the SO instantiation (e.g., for the JACOSO SO devices), are set by ConfiguratorTask.
- *SO Service* is defined as UserDefinedTasks implemented as JADE-based behaviors. The application logic encapsulated in UserDefinedTasks can exploit the JACOSO SO inference rules required for the SO decision-making process by interacting with InferenceRuleTask.
- *Augmentation* is handled by the DeviceManager which, by means of different DeviceAdapters, interfaces JACOSO SO with heterogeneous augmentation devices. In particular, BMFAdapter and SPINEAdapter allow the management of Wireless Sensor and Actuator Networks (WSANs) and Body Sensor Network (BSN) respectively through BMF (Building Management Framework) [115] and SPINE (Signal Processing In-Node Environment) [116] frameworks. BMF is a domain specific framework, expressly conceived for the management of WSN in the context of environment monitoring and building automation; while SPINE is designed for efficient management of BSNs. Both SPINE and BMF comprise networks of heterogeneous devices (e.g., Shimmers, Telos-B and MICA2 sensor motes, Android-based devices, and conventional computers) based on typical IoT standards (e.g., IEEE 802.15.4, ZigBee, 6LowPan, Bluetooth) [117] and they interact with JACOSO by means of the related deviceAdapters.
- *SO Communication* is handled by the CommunicationManager which enables JACOSO SO to flexibly support different communication patterns by just implementing appropriate CommunicationAdapters. In particular, ACLCommunicationAdapter allows a direct message passing of ACLMessages [92] between a sender and receiver; TopicPSAdapter, instead, realizes an asynchronous one-to-many communication in which ACLMessages sent by a publisher are only received from those who have subscribed the related topic and operate in the same platform. JACOSO guarantees high versatility, allowing the implementation of SOs and IoT systems of different scales and within different application scenarios just by re-using and/or re-implementing some components. Such components that can model new SO functionalities are indicated as hot-spots and are (i) ConfiguratorTask that sets up specific SO Basic Info, SO components and Tasks at the SO instantiation time; (ii) UserDefinedTasks that encapsulate a specific SO application logic; (iii) CommunicationManager, because it should be set up to handle new CommunicationAdapters realizing other communication patterns (e.g., web services and sockets) beside or instead of the existing

ones (direct message passing and publish/subscribe); and (iv) DeviceManager if other DeviceAdapters are introduced to interface JACOSO SO with specific devices. On the opposite, JACOSO architectural blocks that do not need to be changed are the so-called frozen-spots.

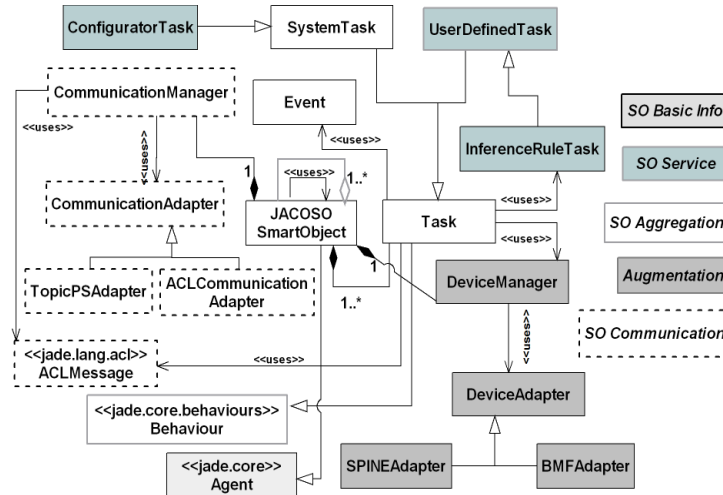


Fig. 3.7. Implementation phase: JACOSO SO Metamodel

3.5 Discussion

In the previous subsections a set of metamodels, each of which is functional to a different development phase, has been presented. Here, the relations among the entity concepts in the different phases will be explained. Indeed, from analysis to implementation, SO-related concepts evolve, being refined from high-level abstractions to implementable software components. In Table 3.3, such concepts are listed and hence their mapping to different development phases of analysis, design and implementation is presented. First, it should be noted that an SO at the analysis phase is described as a very abstract entity, becoming an agent only at the design phase: since the High-Level SO Metamodel is unbound from any paradigm, it may be used as a reference model, similar to AIOTI [38] and IoT-A [66] ones. Moreover, the abstract SO User introduced at the analysis phase is further replaced by an agentified user (ACOSO SO, if it aggregates and exploits other SOs) at the design phase and by a JADE agent at the implementation phase (JACOSO SO, if it aggregates and exploits other SOs). SO Basic Info, individually described in a High-Level SO Metamodel, is spread between the SO itself and its knowledge

base in the design and implementation phases. Regarding the augmentation, SO devices are simply reported at the analysis phase while their management (`DeviceManagementSubsystem` and `DeviceEvent`) and actual interfacing (`DeviceAdapters` and `DeviceManager`) are elicited respectively in the ACOSO-based and JACOSO SO metamodels. SO communication features, not explicitly highlighted at a High-Level SO metamodel, are introduced at the design and implementation phases.

Indeed, an ACOSO-based SO Metamodel presents a `CommunicationManagementSubsystem` exploiting `ExternalEvents` and `InternalEvents` while a JACOSO-based SO Metamodel introduces the `CommunicationManager`, customizable `CommunicationAdapters`, and FIPA-compliant `ACLMessages` infrastructures. Finally, an SO Service concept is first abstractly presented in terms of operations and QoS indicators at the analysis phase and then refined as an application-level `UserDefinedTask` and `ServiceEvents` at the design and implementation phases. With regard to the fulfillment of SLRs and TLRs discussed in Section 2.4, ACOSO-Meth provides a systematic and full-fledged approach (SLR₆) to the SO development, exploiting (i) the agent abstraction to virtualize and homogenize the different SOs to be developed (SLR₁); (ii) a flexible and modular communication infrastructure (comprising at design phase the `CommunicationManagementSubsystem` and at implementation phase the `Communication Manager` with its `CommunicationAdapters`) to enable voluntary communication among different paradigms and data formats (SLR₂ and SLR₅); (iii) a customizable augmentation infrastructure (comprising the `DeviceManagementSubsystem` at design phase and `DeviceManager` with its `DeviceAdapters`) to enable interoperability among heterogeneous IoT devices (TLR₁); (iv) well-known FIPA-compliant interfaces and ontology to straightforwardly access SOs functionality, historical and contextual information (leveraging at design phase on the `KBManagementSubsystem` and at implementation phase on SO internal knowledge bases, SLR₃ and SLR₄); (v) the ACOSO-middleware (in particular its domain-neutral metamodels and programming techniques) and the JADE facilities (e.g., AMS and DF) to speedup SOs prototyping and evolution (TLR₄), and support their augmentation variation (TLR₂) and decentralized management (TLR₃); and (vi) a revised scale concept to unambiguously characterize SO-based IoT systems and possibly enable their comparison (SLR₇ and TLR₅).

Table 3.3. Evolution of the SO main concepts from the analysis to the implementation phase

Concept	High-Level SO Metamodel	ACOSO-based SO model	Meta-JACOSO SO Metamodel
	(Analysis Phase) High-level (conceptual) entity	(Design Phase) ACOSO Agent, Behavior, Task	(Implementation Phase) JADE Agent, JADE Behaviors, Task
User	SO, human, digital system	Agentified user, ACOSO SO	JADE Agent, JACOSO SO
SO Basic Info	FingerPrint, Location, Status, PhysicalProperty	(Agent) SmartObject, KBManagementSubsystem	(JADE Agent) SmartObject, InferenceRuleTask variables
Augmentation	Device	DeviceManagementSubsystem, DeviceEvent	BMFAdapter, SPINEAdapter, DeviceManager
SO Communication	Not explicitly highlighted	CommunicationManagement Subsystem, InternalEvent, ExternalEvent	TopicPSAdapters, ACLCommunicationAdapter, CommunicationManager, ACL Messages
SO Service	Service	UserDefinedTask, ServiceEvent	UserDefinedTask, InferenceRuleTask, ServiceEvent

3.6 A methodology extension: towards Opportunistic IoT services

IoT services represent a novel class of customized, pervasive, cyberphysical services, promising to play a crucial role within IoT ecosystems: however, some essential concepts, as highlighted in Section 2.3.1, have been so far overlooked by the available IoT service models.

In such direction, this Section introduces a novel “Opportunistic” IoT service definition and proposes a novel approach [50, 118] to fully support IoT service development, comprising two main steps: (i) metamodeling, in which high-level representations are provided, mainly for descriptive purposes, to outline a service overview particularly suitable for the analysis phase; and (ii) operational modeling, in which services are formalized following specific notations to support the further phases of service design, verification and simulation. These two steps (based on the same concepts but presented from two different perspectives) are both centered around innovative cyberphysical IoT services involving heterogeneous entities, generally defined “IoT Entities”, within a certain “IoT Environment”, to be detailed later, as depicted in Figure 3.8. Similarly to models surveyed in Section 3.6, we consider IoT services as interfaces for making an IoT Entity’s functionality accessible by other IoT Entities. Conversely, our IoT service model is the first that explicitly considers the following “Opportunistic” properties, crucial to capture the real IoT service potentials but largely overlooked in the past:

1. *Dynamicity*, IoT services can be dynamically, and not a-priori, created/activated;
2. *Context-awareness*, any implicit/explicit information about the current location, identity, activity, and physical condition of the involved IoT entities should be considered;
3. *Co-location*, IoT services are created for being simultaneously exploited by different IoT entities sharing the same (cyberphysical) resources in the same location;
4. *Transience*, IoT services can last for a temporary time or till certain conditions are met.

By integrating such considerations with the background surveyed in the Section , we define an Opportunistic IoT Service as an *interface that allows an IoT entity to be engaged, under specific constraints and pre/post-conditions, in a temporary, contextualized and localized usage relationship. The service provision impacts the service provider(s), service consumer(s) (and, in some case, third parties indirectly involved to the service provisioning), by modifying their properties and/or their status.*

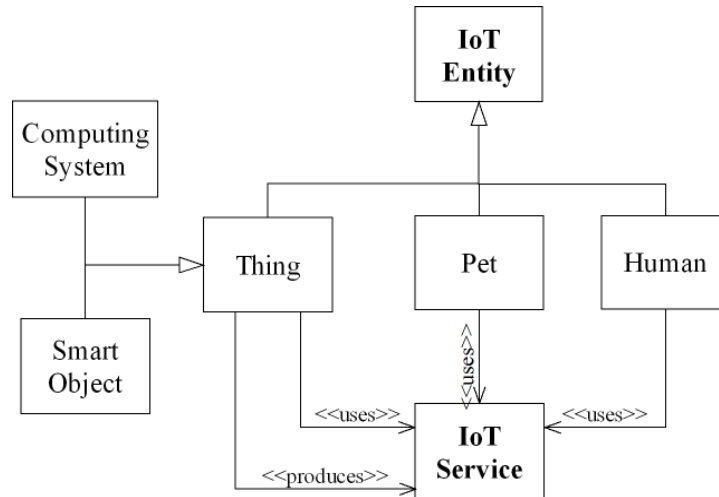


Fig. 3.9. IoT Entities and their roles in IoT Service provision

that, if violated, prevent the SO from working, such as electric voltage, and maximum SO work temperature), and Preference (helping choose between alternatives options, properties, modalities, e.g., a SmartCar with a preferred fuel brand) fields, where a preference is not necessarily stable over time and, as opposed to a Constraint, it can be disregarded; and (ii) in the Service category, the Service Model and Service Profile fields, with the same purposes of [45].

IoT Environment and Context Metamodeling

Differently from the conventional computing services, usually loosely impacted by context-awareness, co-location or transience, IoT Services are actually and opportunistically tightly related to the IoT Environment. It represents the physical environment without any augmentation (e.g., a parking area, an agricultural field, and an industrial warehouse) in which IoT Entities and Physical Elements (e.g., trees, unanimated obstacles, and weather phenomena) are co-located during the IoT Service provision. Context, instead, represents a set of dependencies among IoT services and both IoT Entities and the IoT Environment. Indeed, service provision is expected to exploit any implicit or explicit information regarding IoT Entity, IoT Environment, or other IoT Services. For example, an IoT Service can be influenced from an IoT Entity constraint or preference, as well as from the dimensions of the physical environment.

IoT Service Metamodeling

Each IoT Service is featured by a Service Profile and a Service Model (extending the one reported in [2]), such that it can be accurately described, automatically discovered, consumed or composed. The Service Profile contains the

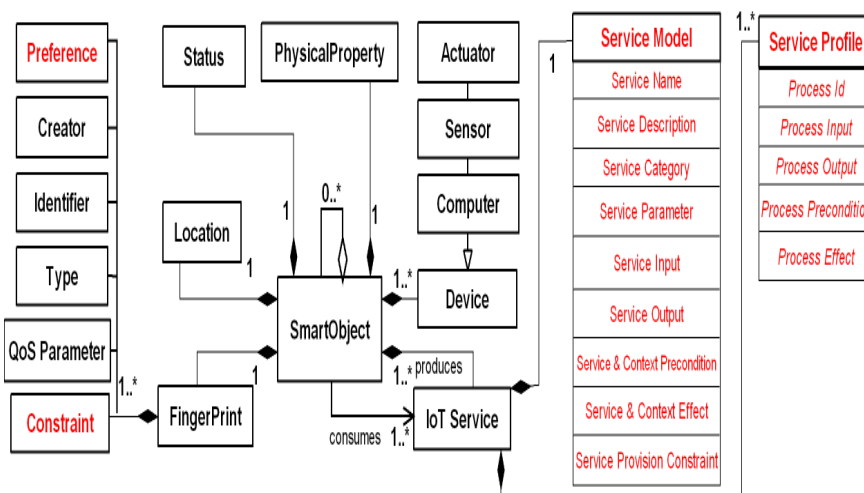


Fig. 3.10. Smart Object modeling and main features related to IoT Service provision (in red the extensions with respect to [3])

main attributes describing the IoT Service itself and the relationships between the service provision and the involved IoT Environment. In detail:

- Service Name: it refers to the name of the IoT Service that is being offered. It can be used as services identifier;
- Service Description: it provides a brief human-readable description of the IoT Service;
- Service Category: it refers to an entry in some IoT Service ontology or taxonomy (e.g., monitoring, and payment);
- Service Parameter: it describes the quality parameters provided by the IoT Service (e.g., latency, and precision);
- Service Input: information required for the IoT Service execution;
- Service Output: information generated as output of the IoT Service execution;
- Service Precondition & Service Context Precondition: functional and IoT Entity-related conditions required for a valid IoT Service execution;
- Service Effect & Service Context Effect: events involving IoT Entities which result from the IoT Service execution;
- Service Provision Constraint: IoT Entity's constraint that is relevant to the IoT Service execution.

The Service Model, instead, contains details about a process, namely the operation(s) concretely implementing the IoT Service. In detail:

- Process Id: it identifies the process;
- Process Input: it specifies the information that the Process requires for its execution;

- Process Output: it specifies the information generated from the Process execution;
- Process Precondition: it specifies the condition under which the Process has place;
- Process Effect: events or changes to the state of IoT Entities that result from the Process execution.

IoT Service Operational Modeling

For a number of reasons, IoT services promise to be notably more complicated, heterogeneous and large-scale than conventional ones. First, the IoT service deployment phase is obviously notably complex, time-consuming, and error-prone, comprising not only software distribution but also the configuration of (even thousands of) heterogeneous devices according to their specific resources and surrounding environment [103, 119]. Second, IoT service provisions cannot underestimate several issues related to the network size, density, and topology, as well as failures and changes to service working conditions, that are difficult to be described through static metamodels [109, 119]. Third, IoT services require to completely adhere to their expected provisions, since they perform cyberphysical actuation in time-sensitive, critical environments [44]. It follows that the static and descriptive, yet accurate and expressive, IoT service metamodels need to be complemented by operational IoT service models for paving the way toward verification and simulation phases.

Considering that IoT Entities and Service interactions are typically (asynchronously) event-driven and time-dependent (namely influenced by the current state and previous history), IoT systems may be formally modeled as DESs (Discrete Event Systems)[120]. Indeed, DES formalization allows descriptive models to be mapped into operational representations, enabling the subsequent verification and simulation by means of different computing tools. Essential elements in DESs are the (discrete) Event set Ev and the (discrete) States Space Ss . Ss comprises all the services states (e.g., activation, ready, execution, and aborted) that can be reached according to the possible events (e.g., input received, computed value out of threshold, physical constraint violated, etc.) included in Ev . Doing so, it is possible to model, verify and simulate IoT Services by taking into account relevant elements defining their Service Profile and Service Model (e.g., service/process input, output, preconditions, and effects), as well as important IoT Entity features (e.g., constraints, and preferences locations). In Figure 3.11, an example Opportunistic IoT Service S involving an IoT Entity E and an IoT Environment Env is modeled as a DES through a finite state automaton. Essential elements in DESs are the (discrete) event set Ev and the (discrete) state space Ss . In the proposed example, Ss comprises five service states (service activated, ready, in execution, terminated, and aborted) that are reached according to seven possible events, related to S , E and Env , which constitute Ev . Indeed, service S is activated only if functional preconditions p_1 and p_2 (indicated as servicePreconditions,

following the service model notation of [45]) are satisfied and it is executed only if entity E provides an input within 30 seconds. In such case, the service execution impacts E and Env . If E provides no input within 30 seconds (representing by over-lining the related event ev_4) after the service execution or, in the meanwhile, its physical constraint c_1 is violated, then S is aborted. Any context change occurring during the service execution requires a new input from E .

Petri nets and their extensions (e.g., for dealing with real time and stochastic systems) represent an excellent model for DESs and provide a well-established suite of tools for their formal verification [121, 122]. Future works will also explore advanced operational models for large-scale collective adaptive systems, such as the work in [119].

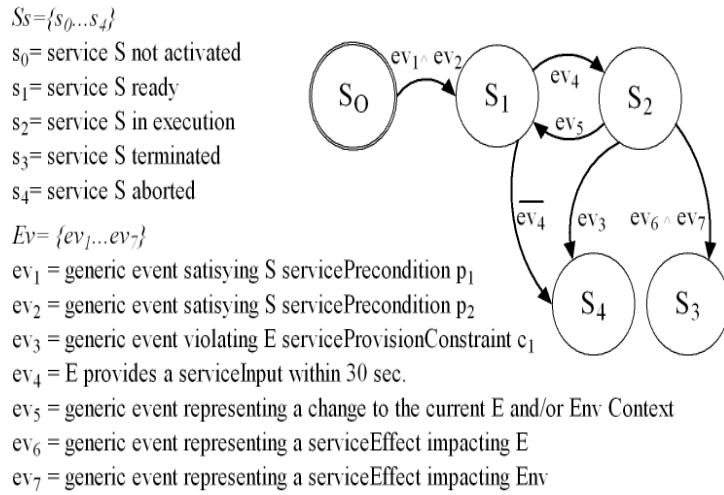


Fig. 3.11. Abstract IoT Service modeled through an FSA

3.7 Summary

This chapter has introduced the application domain-neutral, agent-oriented and metamodel-based ACOSO-Meth methodology for SO-based IoT systems development. By exploiting the ABC paradigm (whose features allow to straightforwardly instill smartness and autonomy within a single SO and thus realize cognitive and autonomic IoT systems) and the ACOSO middleware's agent-oriented modeling and programming techniques, ACOSO-Meth significantly facilitates and speeds up all the development phases leading to the actual and full-fledged engineering of an IoT system. Three levels of SO meta-models (respectively linked to the analysis, design and implementation phases)

have been proposed, providing a seamless support among the different phases of SO development process. In addition, since many design decisions affecting the final configurations of under development SOs and IoT systems can be taken as a result of simulations, a hybrid approach for straightforwardly simulating an ACOSO SO in OMNeT++ has been presented. Furthermore, considering that services are the real IoT drivers, an extension of the ACOSO-Meth has been presented, proposing a novel full-fledged approach that support opportunistic IoT Service development by means of descriptive metamodels and operational models. They allow testing (and thus, better designing and understanding) IoT services before their actual implementation so as to inspect important issues related to the network size, density, and topology, as well as failures and changes to service working conditions.

The effectiveness and efficiency of ACOSO-Meth in supporting the development of IoT ecosystems of different complexities and scales will be proved in the next chapter of this Thesis by means of four use cases referring to different application contexts.

Smart and Interoperable IoT Ecosystems

This chapter emphasizes how the proposed ACOSO-Meth is actually able to support the development of heterogeneous IoT ecosystems and related services in different application contexts. Indeed, one of the main goal of ACOSO-Meth is to provide a domain-neutral methodology that can seamlessly support the analysis, design, and implementation phases through a modular and versatile approach.

Therefore, some interesting use cases improving the current state-of-the-art have been (fully or partially, according to their specific needs) realized following the ACOSO-Meth development approach, and they have been reported hereinafter.

4.1 Smart Unical

The application of ACOSO-Meth for engineering a complex Smart University Campus IoT ecosystem, specifically prototyped at the University of Calabria and named Smart UniCal, is presented in the following. Several references to Smart University/Smart Campus scenarios are available in the literature [123], [124], [125], [126] and [127] and, regardless of particular goals or implementations, they all present “comfortable and user-tailored environments, rich in innovative services”. Our Smart UniCal system (Figure 4.1) is an aggregated SO composed by a Smart Bridge (dotted yellow bordered area) and Smart Departments (yellow bordered area), spanning multiple adjacent buildings, which contains smart rooms such as Smart Lab and Smart Office. Smart UniCal SOs have been characterized respectively in “L”arge (i.e., the SmartBridge), “M”edium (i.e., the SmartDIMES) and “S”mall scale (i.e., the SmartSenSysCalLab) SOs, according to the considerations reported in Section 2. In particular, Table 4.1 reports the list of services provided by Smart UniCal SOs:

- SmartBridge provides a cyberphysical service for a structural health monitoring [128] purpose;



Fig. 4.1. The Smart UniCal infrastructure: the SmartBridge part (dotted yellow bordered area) which crosses the SmartDIMES (yellow bordered area with building identification codes)

- SmartDIMES (Department of Informatics, Modeling, Electronics and Systems Engineering), namely a Smart Department, provides a cyberphysical service to remotely control department spaces and facilities, e.g., HVAC (Heating, Ventilation and Air Conditioning) systems and lights, aiming to save energy; and
- SmartSenSysCaLab, namely a Smart Lab, provides cyberphysical services to laboratory users who are supported in their daily activities.

It should be noted that the domain-neutrality of the ACOSO-Meth and ACOSO middleware allows supporting the development of the different kinds of Smart Unicals SOs and related services by keeping the same methodological approach and by exploiting the same metamodels and programming techniques. In the following, the descriptions of the SmartBridge in the analysis, design and implementation phases are provided according to the ACOSO-Meth. Finally, the Smart UniCal performance evaluation is presented.

Table 4.1. Smart Objects constituting Smart Unical along with their provided services

Scale	Smart Object	Service	Description
S	SmartSenSysCaLab	smartWellness	Correct lifestyle suggestions
		smartComfort	Workplace conditions improvement
M	Smart DIMES	smartMonitoring	Indoor environmental monitoring
L	Smart Bridge	smartVibration	Bridge vibrations monitoring

4.1.1 Analysis phase

Smart Bridge (Figure 4.2) is a large-scale SO physically based on the “Pietro Bucci” bridge, which crosses the Unical campus for 1.22 Km, linking together all the 14 university departments (spread among different building units called cubes). Its Administrator can query its status, specifically the currently recorded vibration, or use the smartVibration service for monitoring the bridges structural health [128]. Service smartVibration allows the analysis of the vibrations generated by the transit of vehicles and pedestrians upon the bridge. If the sensed vibrations reach warning thresholds, the service notifies such event to the Administrator. In order to provide such service, Smart-Bridge is augmented through different devices, including accelerometer sensor nodes and laptop base stations, coordinated by a PC acting as a main coordinator. In more details, smartVibration service has two basic operations: getVib that exploits accelerometer sensors on the bridge to accurately sense the vibrations, and vibAnalysis that exploits SmartBridges computing devices to elaborate the raw vibration data acquired and to compare them with the defined thresholds. The getVib operation has a response time in the order of second while vibAnalysis detects all the vibrations exceeding the warning thresholds with 100% accuracy.

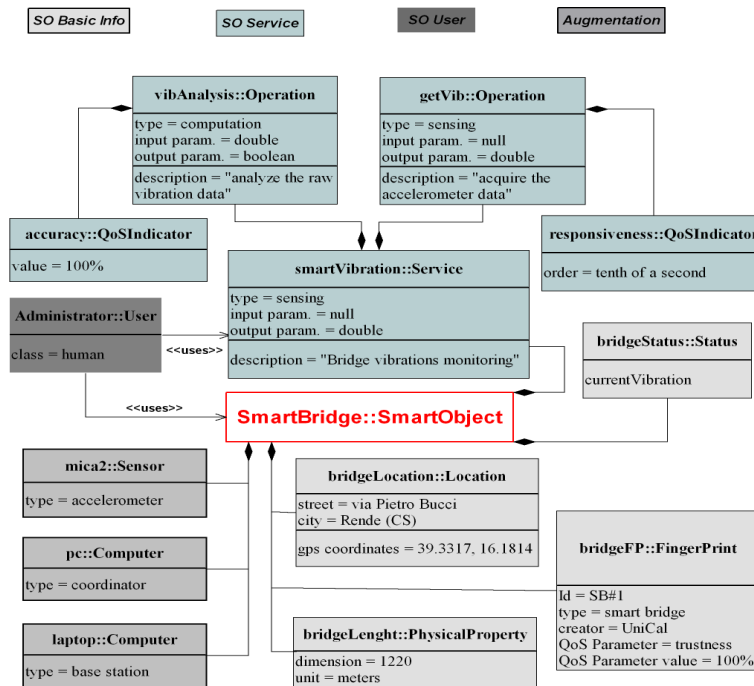


Fig. 4.2. High-Level SmartBridge Metamodel at analysis phase.

4.1.2 Design phase

SmartBridges High-Level metamodel is refined at the design phase, resulting in Smart Bridges ACOSO-based metamodel as shown in Figure 4.3. In particular, SmartBridge is modeled as an ACOSO-based agent and SO Users as generic agents. The smartVibration service and the vibAnalysis and getVib related operations are modeled as UserDefinedTasks (smartVibrationTask, vibAnalysisTask and getVibTask respectively) driven by the corresponding ServiceEvents (getVibEvent and vibAnalysisEvent). The vibrationSensingEvent, instead, allows interfacing the accelerometer sensors with SmartBridge, e.g., providing the raw vibration sensed data.

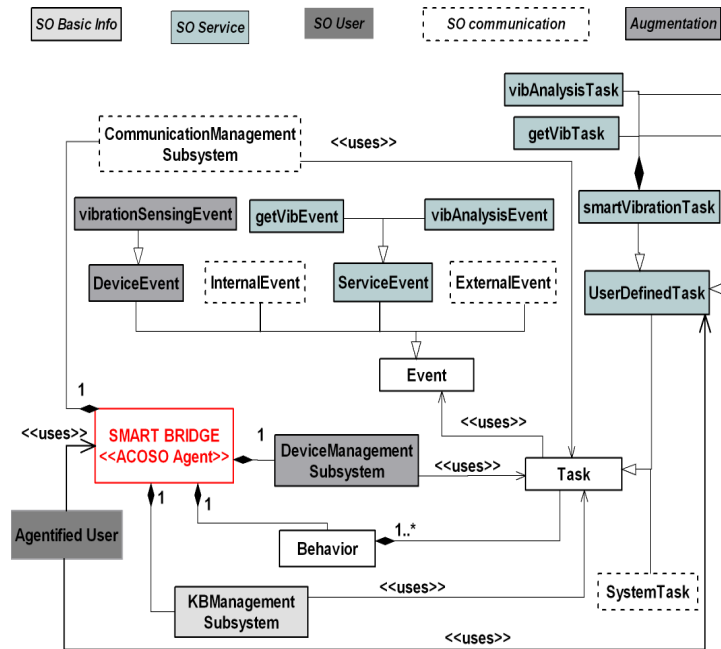


Fig. 4.3. ACOSO-based SmartBridge Metamodel at design phase.

4.1.3 Implementation phase

Smart Bridges ACOSO-based metamodel is refined at the implementation phase, resulting in Smart Bridges JACOSO-based metamodel (Figure 4.4). In particular, in this phase the generic agentified SmartBridge is specialized into a JADE-based agent, as well as the agentified SO User. ACLCommunicationAdapter allows SmartBridge exploiting a direct ACL-based messages exchange mechanism. SmartBridgeInferenceRuleTask contains both inference

rules required for a SmartBridge decision-making process and current values of the variables constituting such inference rules. UserDefinedTasks (smartVibrationTask, vibAnalysisTask and getVibTask) implementing smartVibration and related events (vibAnalysisEvent and getVibEvent) are modeled as JADE Behaviour, while BMFAdapter interfaces SmartBridge with its devices. Interaction diagram of Figure 4.5 illustrates the methods realizing the Smart Bridges Smart Vibration service.

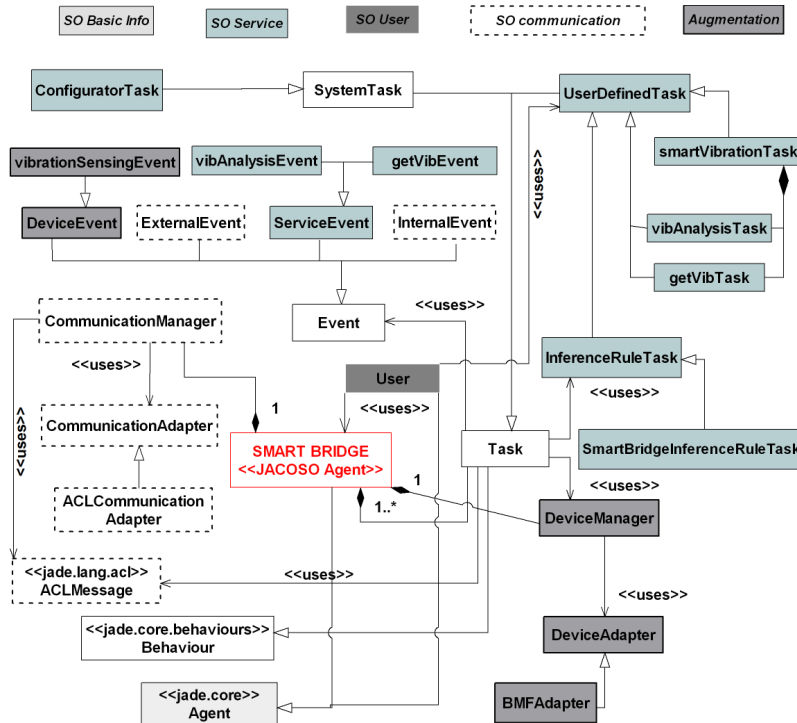


Fig. 4.4. JACOSO-based SmartBridge Metamodel at implementation phase.

4.1.3.1 Technical implementation details

In the following, some key technical implementation details of the Smart UniCal system, related to the used IoT devices and to the implemented JACOSO-based software components, are described. In particular, Figure 4.6 shows some technical deployment snapshot of the IoT devices of SmartBridge, SmartDIMES and SmartSenSysCalLab.

Table 4.2 shows the characteristic of main hardware/software devices. Such heterogeneous IoT devices along with Android-based devices and conventional computers adopting different technologies (e.g., IEEE 802.15.4, Wifi,

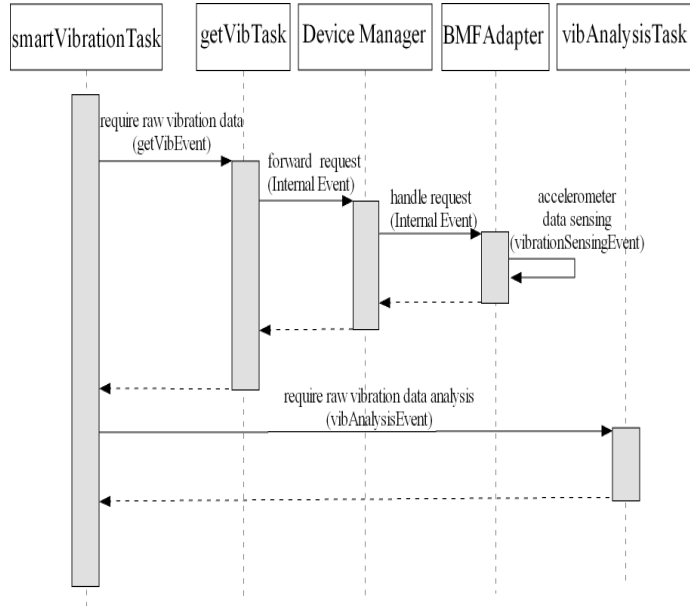


Fig. 4.5. Interaction diagram of the Smart Bridges Smart Vibration service.

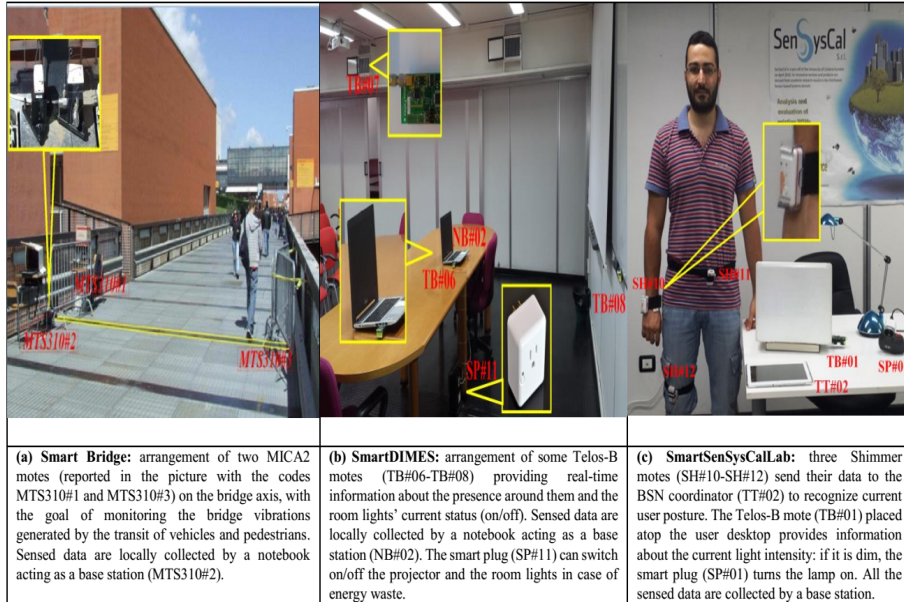


Fig. 4.6. Snapshot of the IoT devices of (a) SmartBridge, (b) SmartDIMES and (c) SmartSenSysCalLab

and Bluetooth) and managed by different frameworks (i.e., SPINE [116] and BMF [115]) have been made interoperable through the related deviceAdapters provided by the ACOSO middleware, independently from low-level network protocols or different communication paradigms through the exploitation of the related communicationAdapters. We focus on the implementation of the following services: (1) smartVibration service of SmartBridge, (2) smartMonitoring service of SmartDIMES, and (3) smartWellness and the smartComfort services of SmartSenSysCalLab.

1. smartVibration is based on the data gathered by 90 Crossbow MICA2 devices. Every 27 meters, two of them are deployed facing each other and laying on a metallic beam that transversely passes through the axis of the bridge (Figure 18(a)). Such network of Crossbow MICA2 devices is managed by 9 notebooks (placed in rooms in front of the bridge such that each notebook can manage data of its closest 10 motes), hosting the BMF application and collecting the data, while a central PC acts as a main coordinator and hosts the SmartBridge SO application. Each notebook works in a different subnetwork and all the notebooks interact with the main coordinator through an IP-based WiFi UniCal Intranet; Crossbow MICA2 devices, instead, are connected to their associated notebook through the 802.15.4 wireless protocol. Totally, 20 non-overlapping subnetworks have been used to realize this service.
2. smartMonitoring is based on 43 Telos-B-based indoor environmental sensors, i.e., humidity, temperature, light, and presence sensors, and on 20 Telos-B-based actuators (i.e., smart plugs) deployed within 18 DIMES rooms (Figure 18(b)). In detail, at least two devices (one sensor and one actuator) have been deployed for each of the 18 monitored environments, located in different cubes. Each monitored environment is associated to a laptop (the environments located at the same floor of the same cube share the same one) hosting a base station and running the BMF application (totally, ten laptops have been used and interconnected through 10 overlapping subnetworks). Each laptop interoperates with the associated sensors/actuators through the BMFAdapter: it allows the collection of sensed data from the sensing devices to the base station, and the forwarding of commands from a base station to actuating devices. The SmartDIMES application is hosted in a separate laptop. The SmartDIMES administrator, through such application, can transparently manage all the environments and send both request and configuration messages to the deployed Telos-B motes and smart plugs.
3. smartWellness provides customized and real-time hints to SenSysCal lab users by displaying notifications on their personal smartphones and/or laptop monitors. Data coming from 12 light/presence Telos-B sensors (one for each of the ten SenSysCal desktops, one at the entrance and one in the middle of the lab) and from 30 users wearable Shimmer sensors (three for each user, placed at user wrist, waist and leg) are forwarded by means of

BMFAdapter (environment data) and of SPINEAdapter (wearable data), to a base station. The base station is a laptop running the SmartSenSysCalLab application that collects the overall data, elaborates them and sends back customized notifications to users (specifically, on their Twitter profile or on the computer screen placed on their desktop). The same base station, in the context of a smartComfort service and through BMFAdapter, periodically queries the light intensity value to every Telos-B sensor deployed atop one of the 12 user desktops. In case of poor lighting, the corresponding desktop lamp is switched on through its smart plug. The aforementioned devices that contribute to realize the SmartSenSysCalLab services (Figure 18(c)) are connected to the local laboratory subnetwork.

Table 4.2. Main hw/sw characteristics of the IoT devices used to implement Smart Unical SOs and their service

Device	Main characteristics	SO/Services
MICA2	<p><i>OS:</i> TinyOS. <i>CPU:</i> Atmel Atmega 128L (8 bit bus, 8MHz clock). <i>Memory:</i> 4K Ram 128K Flash 512K EEPROM. <i>Radio:</i> 802.15.4 compatible CC2420. Expansion board (2-axis accelerometers). <i>Battery:</i> 2X AA batteries (4000-5000 mAh in total, depending on the cell).</p>	SmartBridge/ smartVibration
Telos-B	<p><i>OS:</i> TinyOS. <i>CPU:</i> TI MSP430F1611 (16-bit bus, 4-8MHz clock). <i>Memory:</i> 10K Ram 48K Flash 1M EEPROM. <i>Radio:</i> 802.15.4 compatible CC2420. On-board sensors (humidity, temperature light sensors). <i>Battery:</i> 650 mAh.</p>	SmartDIMES/ smartMonitoring SmartSenSysCalLab/ smartComfort
Shimmer	<p><i>OS:</i> TinyOS. <i>CPU:</i> TI MSP430F1611 (16-bit bus, 4-8MHz clock). <i>Memory:</i> 10K Ram 48K Flash. <i>Radio:</i> 802.15.4 compatible CC2420. On-board sensors (3-axis accelerometer) <i>Battery:</i> 650 mAh.</p>	SmartSenSysCalLab/ smartWellness

4.1.3.2 Performance evaluation

As presented in Section 4.1.3, ACOSO-Meth is based on a JACOSO SO Meta-model for system implementation, deployment and execution. In the following, the Smart UniCal performance evaluation is presented to assess the suitability of the ACOSO-Meth implementation phase in actually supporting efficient small-, medium- and large-scale IoT systems. Indeed, SmartBridge, SmartDIMES and SmartSenSysCalLab SOs and their aggregated IoT devices, are evaluated when providing their specific services (see below). However, in order to define the proper scenario size (number of SOs and their distribution in different subnetworks) that effectively enables the developed services, preliminary tests were conducted to evaluate SOs performance, thus analyzing possible bottlenecks (Figure 4.7 in the information exchange (IE) phase).

Please note that the deployment stage and performance evaluation require a significant effort, especially due to the number of SOs involved: fortunately,

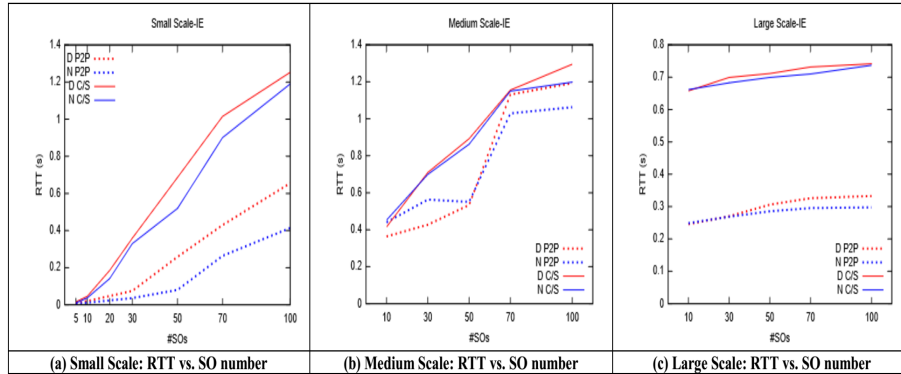


Fig. 4.7. SmartSenSysCal, SmartDIMES, and SmartBridge performance evaluation considering different communication paradigms (C/S or P2P) and MGR models (D or N)

we can leverage on our previous experience in the fields of WSNs and cyber-physical systems [115, 116, 81, 12] to speed up the identification of operation modalities and performance indices, as well as the SO monitoring and data gathering. In particular, we considered SOs exchanging 2KB fixed length simple FIPA-compliant data messages by following either a Client/Server (C/S) or a Peer-to-Peer (P2P) paradigm. As some services are intrinsically centralized or distributed, they can be implemented following either a C/S or a P2P paradigm. Moreover, we considered IoT device data sources (or simply data sources) with either stochastic normal distribution (N, with mean = 0.5 msg/s and variance = 0.2 msg/s) or deterministic (D, 1 msg/s) message generation rate (MGR) models. Given the communication paradigms and MGR models, we focused on two fundamental network-oriented performance indices for distributed SOs when providing specific services collaborating with each other: (i) message delivery ratio (MDR); and (ii) round trip time (RTT). In Figure 4.7, however, only the RTT values calculated in small- (SmartSenSysCalLab), medium- (SmartDIMES), and large-scale (SmartBridge) scenario are shown, as the MDR values are always 100%, being JADE communications based on TCP connections, thus fully reliable. Figure 4.7(a) highlights how the increase of the involved SOs in the small-scale scenario adversely affects RTT, which rapidly grows due to the network congestion. In Figure 4.7(b), differently from Figure 4.7(a), where SOs are supported by just one network within a squared grid of side 10 m, SOs are now distributed in 10 different sub-networks within a squared grid of side 250 m. In such a deployment area, it happens that adjacent networks interfere with each other, since their coverage radii overlap. Nevertheless, a better SO distribution implies less congestion and lower RTT. For example, the RTT of 100 SOs distributed in 10 subnetworks is definitively lower than the RTT of the same number of SOs deployed in one network. Finally, in the large-scale scenario of Figure 4.7(c), the SOs are distributed in

20 non-overlapping subnetworks within a squared grid of side 1000 m. Differently from the small- and medium-scale scenarios, in Fig 4.7(c), it can be noted that increasing the number of SOs has a little impact on RTT. Compared to the same configuration of a medium-scale scenario, RTT values are lower. In fact, in a large-scale scenario, networks are deployed in a wider area. Thus, they do not interfere with each other and consequently both congestion and RTT decrease. For example, given 50, 70 and 100 SOs, RTT values in the large-scale scenario significantly decrease by comparing those in the medium-scale scenario. The aforementioned SOs performance evaluation has provided important insights to define, for each SO and for each SO service, the best operation modalities in the Smart UniCal ecosystem. Such modalities are detailed in Table 4.3 by specifying: the number of embedded devices ($\#EDev$), number of involved subnetworks ($\#SubNets$), evaluation time ($EvTime$), message length (ML) and deterministic message generation rate ($D-MGR$). Given the SOs and SO service operation modalities as shown in Table 4.3, the Smart UniCal performances have been evaluated in terms of MDR and RTT, IoT devices energy and memory consumption (base stations are powerful and less constrained than motes and they are typically plugged to the mains electricity, such that they can be easily recharged), and the provided results reported in Table 4.4. Such performance indices have been chosen in order to characterize SO performance both functionally and non-functionally: indeed, they provide insights about services reliability and responsiveness (according to the performance indices previously outlined to describe the IE phase), but also about the required resources (energy and memory, in particular). The latter is a relevant aspect considering that most of the IoT devices are resource-constrained. The results reported in Table 4.4 confirm the RTT trends shown in Figure 4.7 and JADE message systems high reliability, being based on the TCP protocol. Then, the increase of $\#EDev$ adversely affects both RTT, which grows due to the network congestion, and energy consumption, especially if also the evaluation time increases (in the case of smartVibration services, the residual energy is only slightly nicked since MICA2 capacity is bigger than those of Shimmer and Telos-B ones).

Table 4.3. Operation modalities of the Smart Unical Smart Objects

SO	Service	#EDev	#SubNets	Operation Modality	EvTime (h)	ML (KB)	D-MGR (msg/s)
SmartSenSysCallLab	smartComfort	25	1	Periodically (every minute), each desktop light intensity is acquired and sent to the base station	8	2	1
	smartWellness	42		Periodically (every minute), environmental and body data are acquired and sent to the base station			
SmartDIMES	smartMonitoring	73	10	Periodically (every 30 seconds) environmental data acquired and sent to the base station	12	2	1
SmartBridge	smartVibration	100	20	Both periodically (every minute) and occasionally data are acquired and sent to the base station	12	2	1

Table 4.4. Smart Unical performance evaluation

Service	MDR	RTT (s)	Residual Energy	Residual RAM-ROM
smartComfort (SmartSenSysCalLab)	100%	0.046	85%	63%-28%
smartWellness (SmartSenSysCalLab)	100%	0.059	57%	56%-17%
smartMonitoring (SmartDIMES)	100%	0.513	48%	60%-26%
smartVibration (Smart Bridge)	100%	0.507	87%	78%-12%

Moreover, EDev deployment on different SubNets affects RTT more than #SubNets. In particular, by comparing the RTT values of SmartSenSysCalLab and SmartDIMES, it should be noted that when #EDev scarcely doubles, RTT increases tenfold; however, if there is no overlapping among the SubNets, then the performances are quite stable, even if #EDev and #SubNets increase, as in the case of SmartBridge. SO lifetimes varies depending on the provided service, devices batteries, operation modalities and scenario configurations as reported in Tables 4.2, 4.3. In particular, we have defined the Residual Energy of an SO X providing a service s by exploiting (all or a set of) its different D_i devices as

$$RE(Xs) = \min\{batteryD_1..batteryD_n\} \quad (4.1)$$

where $batteryD_i$ is the amount of power currently left in a D_i s battery that enables its correct working [129] in the context of service provision. Given such definition, $RE(Xs)$ can vary from 100% (all SO devices involved in the service provision are full of energy) to 0% (at least one SO device has an energy shortage preventing it from correct working). With regard to the Smart Unical and testing, for the sake of simplicity, each SO in providing only a single service, SO service provision varies from 18 hours (SmartSenSysCalLab providing only the SmartWellness service) to 92 hours (SmartBridge providing only the SmartVibration service). Finally, memory consumption results highlight that IoT devices have enough free memory to deploy other in-node services or customized extensions.

4.2 Smart Digital Libraries

Digital Libraries (DLs) are distributed software infrastructures that aim at collecting, managing, preserving, and using digital objects (or resources) for the long term, and providing specialized services on such resources to its users [130]. Currently, DLs include not only books and digitalized textual documents, but also a wide range of digital objects: text document, image, audio, video, software, etc. [131]. In the IoT context, SOs represent a novel type of digital resources, since during their lifecycle they can produce continuous

streams of geolocalized and contextual data also related to their use and their surrounding environment, providing different cyberphysical services to their users. Including SOs, the newest type of digital objects, into DLs could have a twofold implication:

- on the one hand, it would allow DLs to acquire a plethora of novel content creators/consumers, since it is expected that billions of SOs will soon impact our daily life;
- on the other hand, it would allow end users to acquire a valuable tool to simplify the complex SOs management, exploiting the functionalities that DLs provide for their traditional contents.

The SOs inclusion into DLs as novel first-class objects to be collected, managed, and preserved is therefore notably promising yet challenging. In such direction, this case study presents an approach for the inclusion of SOs into DLs which would enable effective discovery, querying and management of SOs based on typical DL tools and facilities [67, 68, 69]. To the best of our knowledge, this approach, although currently focused just to the analysis phase, represents the first research effort for the integration of SOs into DLs, thus seeking towards novel cyberphysical DLs, or Smart DLs (SDLs).

4.2.1 Analysis phase

The proposed inclusion approach is based on the ACOSO High-Level SO metamodel because, as already presented in Section 3.2, it describes all the cyberphysical SOs characteristics (geophysical, functional, and non-functional) through a set of metadata categories that can characterize an SO in any application domain of interest (e.g., Smart Cities, Smart Factories, Smart Grid). In particular, when instantiated, the ACOSO High-Level SO metamodel generates a well-defined but flexible SO metadata model, structured so as to be easily manually or automatically generated, queried and managed. Moreover, it is technology-neutral and can be implemented by using any data modeling language (e.g., XML, JSON). To exemplify the proposed approach, the instantiation of an SO (SmartDesk) metamodel and related metadatamodel (following the JSON format) are reported respectively in Figures. 4.8, 4.9.

The Smart Desk [68] is an SO located in the SenSysCal Lab at the University of Calabria (other specific information are equivalently reported in Figures 4.8 and 4.8) and supports its user Antonio during his daily working activities. In particular, the Smart Desk is equipped with a presence sensor and provides a sensing service to check whether a user is at the desk (PresenceDesk Service) and an actuation service to send messages, targeting the desk user, onto the desk monitor (Visualization Service). There is only one QoS Param defined which is the level of trust (in the range 0..1) of the Smart Desk. According to the SmartDesk metamodel of Figure 4.8, the JSON document reported in Figure 4.9 has six members associated with each of the six

categories of metadata previously described (FingerPrint, PhysicalProperty, Location, Status, Device, and Service).

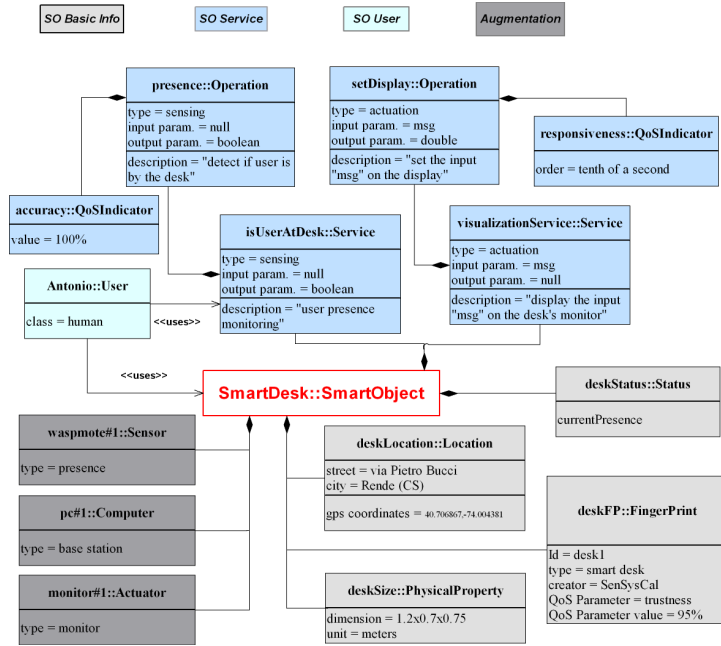


Fig. 4.8. SmartDesk Model

In order to foster the seamless integration of SOs into existing DLs, the proposed SO metamodel and metadata model have been made compliant with the Digital Library Reference Model (DLRM) proposed by the DL.org community [132], that is currently the main reference model for architecting DLs. In particular, the DLRM states that a DL is similar to an Organization, which foundations are six core concepts or domains: Content, User, Functionality, Policy, Quality, and Architecture, as depicted in 4.10.

The first five of them capture the features characterizing the DL and its expected services. The Architecture, instead, captures the systemic properties underlying the expected services. The cornerstone of the DLRM as well as the shared concept between the six DL domains is the Resource, which models any element easily identifiable through an unique Resource ID. As long as the Resource complies with the established specifications defined into the Resource Format (an arbitrarily complex and structured schema usually drawn from an ontology to guarantee a uniform interpretation), it may be accessed, queried and managed. An SO, as compliant with the definition (and also rationale) provided in the DLRM, can be straightly included as Resource in a DL, uniquely identified through its FingerPrint, easily accessed, queried

```

{
  "fingerPrint":{
    "identifier":{"id":"desk1",
      "creator":{"name":"Sensyscal",
        "type":{"primaryType":"desk",
          "QoS_parameter":"trustness",
          "QoS_parameter value":"0.95"}
        }
    }
  },
  "status":{"currentPresence":""},
  "physical_properties":{
    "dimension":"1.20x0.7x0.75",
    "unit":"m"
  },
  "devices": [
    {"pc#1":{"type":"base station"},
    {"waspmote#1":{"type":"presence"},
    {"monitor#1":{"type":"monitor"}
  ],
  "location":{
    "street":"Via P. Bucci",
    "city":"Rende (CS)",
    "gps coordinates":"40.706867,-74.004381"
  }
  "services": [
    {"name":"iUserAtDesk",
    "type":"sensing",
    "input param":"null",
    "output param":"boolean",
    "description":"user presence monitoring",
    "operations": [
      {"id":"presence",
        "type":"sensing",
        "input param":"null",
        "output param":"boolean",
        "description":"visualize information message on the display"
      },
      {"id":"setDisplay",
        "type":"actuation",
        "input param":"msg",
        "output param":"null",
        "description":"display the input msg on the desk's monitor",
        "operations": [
          {"id":"setDisplay",
            "type":"actuation",
            "input param":"msg",
            "output param":"null",
            "description":"set the input msg on the display"}
        ]
      }
    ]
  },
  "QoSIndicator" "accuracy": 100%
  "QoSIndicator" "order": tenth of second
}

```

Fig. 4.9. JSON representation of a smart desk according to the SO metadata model

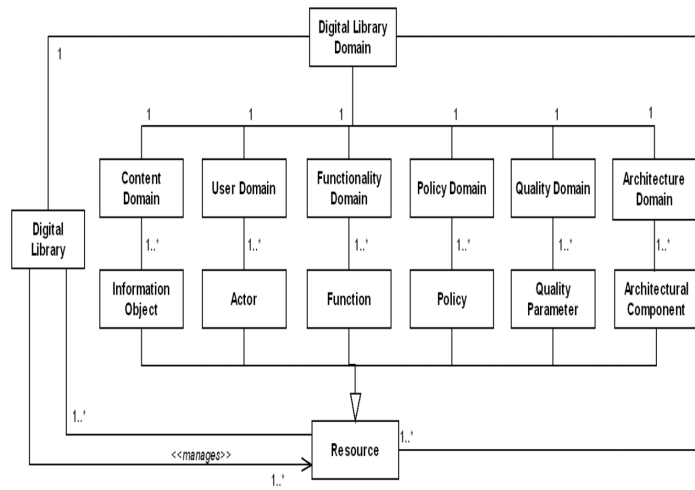


Fig. 4.10. Digital Library Main Concepts

and managed by the DL entities as long as it complies with presented SO metadata model. Table 4.5 shows a map of the aforementioned DL concepts with the ACOSO High-Level SO metamodel and specifically for the Smart Desk case study. Similarly, Tables 4.6-4.10 specifically map the DL domains' main concepts.

Table 4.5. Mapping between the Resource DLRM Concepts, the Smart Object and the SmartDesk ones

DLRM Concepts	General Smart Object Concepts	SmartOffice Concepts
Resource	Instance of Smart Object	Instance of SmartOffice
Resource ID	SmartObject FingerPrint	SmartOffice FingerPrint: <ul style="list-style-type: none"> • Identifier: Office1 • Type: SmartOffice • Creator: SenSysCal • Location: University of Calabria, 41c • QoSParameter: 0,95 Trustness
Resource Format	Smart Object metamodel	SmartOffice metamodel

The Content Domain (see Table 4.6) represents the various aspects related to the modeling of information managed in the DL universe to serve the information needs of the active entities interacting within the DL, namely the Actors. The main Resource of Content Domain is the Information Object, which is an information item that seamlessly provides data to the DL Actors by means of the functionality offered by the DL itself. Such interactions, organized according to different types of criteria, are displayed in different Views and recorded by the Action Log, so allowing the Actor profiling. Specifically, the SO is a novel Information Object that contributes to the production and consumption of content that will be handled by the DL Actors through the SO Services and the SO service requests could be monitored by the Action Log over time so allowing the Actor profiling. Moreover, such content is suitable for being contextualized or displayed in different Views.

The User Domain (see Table 4.7) represents the various aspects related to the modeling of entities, either human or machines, interacting with any DL system. In particular, the DL End-Users are the ultimate clients the Digital Library is going to serve. Specifically, an SOs play a dual role within the DL reference model, and specifically in the end-user domain: in fact, SOs are both content creators, because they produce or update data and information, and

Table 4.6. Mapping between the DLRM Content Domain Concepts, the Smart Object and the SmartDesk ones

DLRM Concepts	General Smart Object Concepts	SmartOffice Concepts
Information Object	Smart Object	SmartOffice
Action Log	When and how a specific SO Service has been used	“When the LightService has been used?”
Actor Profiling	Who used specific a SO Service	“Who used the LightService?”

content consumers, as it often happens that they are themselves users of other SOs or Resources in general.

Table 4.7. Mapping between the DLRM User Domain Concepts, the Smart Object and the SmartDesk ones

DLRM Concepts	General Smart Object Concepts	SmartOffice Concepts
Actor	Users (Smart Objects, Humans, Digital Systems)	SmartOffice, Smart Desk, Smart Whiteboard, Smart Projector and Antonio
Content creator	Smart Object	SmartOffice w.r.t. Light Services user
Content consumer	Smart Object, Smart Object Users	SmartOffice w.r.t. Presence Services user

The Functionality Domain (see Table 4.8) represents the various aspects related to the modeling of facilities/services provided in the DL universe to serve Actor needs. A Function is a particular operation that can be realized on a Resource upon an Actor request. Functions can be specialized in two main classes: the Access Resource Function and the Manage Function. The first family of functions aims at finding Resources compliant to certain (static or dynamic) features (Discovery), querying them (Search-Browse), retaining the content retrieved through specific mechanisms (Acquire) and finally displaying it (Visualize). The Manage Function, instead, supports the production (Create), publication (Publish), updating (Update), configuration (Personalize) and other basic operations related to the Resource lifecycle. It should be noted that these functionalities are directly provided to the DL Actors by the DL for each included Resource, on the basis of the information structured following the given Resource Format. Likewise, these functionalities are not provided directly by the SO but by the DL, on the basis of information structured in the proposed SO metadata model.

Table 4.8. Mapping between the DLRM Functionality Domain Concepts, the Smart Object and the SmartDesk ones

DLRM Concepts	General Smart Object Concepts	SmartOffice Concepts
Access Resource Functions (discovery-search-query-visualize)	The User exploits the DL Discovery Function to discover a specific service; hence, the User submit a request to the Service through the DL Query-Function	The User queries the DL for a specific services: the DL Discovery Function finds that the inserted criteria match with the LightService metadata; so the request is carried out by the Query Function
Manage Functions (create-publish-update-personalize)	The User specifies through the Personalize Function how to display the SO Services usage	The User specifies through the Personalize Function the desired view (daily or monthly) for displaying the LightService request output

The Policy Domain (see Table 4.9) represents a set of guiding principles designed to organize actions in a coherent way and to help in decision making. In particular, the User Policy defines possible User actions on the Resource. The proposed SO metadata model is neutral with respect to the concept of Policy. Few changes to the SO metadata model could be carried out to regulate the interactions between the SO user and the SO services, according to what is present in the reference DL model respectively with the User Policy and Content Policy. In particular, one could implement the concept of Policy by directly associating it to the SO User or SO Service entity, or binding it outside of the SO metadata model, at the level of the DL.

Table 4.9. Mapping between the DLRM Policy Domain Concepts, the Smart Object and the SmartDesk ones

DLRM Concepts	General Smart Object Concepts	SmartOffice Concepts
User Policy	SO Services enabled on the basis of User degree of reliability	SmartOffice w.r.t. to LightService and VisualizationService access

The Quality Domain (see Table 4.10) captures the aspects that permit considering DL systems from a quality point of view, with the goal of judging and evaluating them with respect to specific facets. It represents the various

aspects related to features and attributes of Resources with respect to their degree of excellence. In particular, the DLRM provides Quality Parameter on the Resources (Generic Quality Parameter), on the Information Object (Content Quality Parameter), and on the User (User Quality Parameter). The proposed SO metadata model already contains two elements that refer to the SO quality (QoS Parameter) and the quality of the SO Services (QoS Indicator), in full agreement with the DL reference model that provides Quality Parameters on the Resources (Generic Quality Parameter) and on the Information Object (Content Quality Parameter). Regarding the User, the DLRM presents a User Quality Parameter that could be easily imported into the SO metadata model, for example by assigning each SO User a reliability value, on the basis of which it is possible to define Policy and granting special rights or access privileges to the SO Services.

Table 4.10. Mapping between the DLRM Quality Domain Concepts, the Smart Object and the SmartDesk ones

DLRM Concepts	General Smart Object Concepts	SmartOffice Concepts
Generic Quality Parameter	SO QoS Parameter	Trustness value of QoSParameter
Content Quality Parameter	SO Service QoS Indicator	Accuracy value of PresenceServices QoSIndicator

The Architecture Domain (see Table 4.11) represents the various aspects related to the software systems that concretely realize the DL universe. In particular, it offers useful insights about how to develop new efficient DL systems and how to improve current ones. The inclusion of an SO within the DL architecture presented in the DLRM may involve (i) the insertion of an architectural Running Component, which represents a running instance of a Software Component active on a Hosting Node, suitably designed, based on the SO characteristics and equipped with specific interfaces, or (ii) the creation of a new component, currently not present in the reference architecture, delegated to the SO virtualization [133].

Table 4.11. Mapping between the DLRM Architecture Domain Concepts, the Smart Object and the SmartDesk ones

DLRM Concepts	General Smart Object Concepts	SmartOffice Concepts
Inclusion strategy	Proxy-based inclusion	Proxy-based inclusion

4.3 Opportunistic IoT services

The modeling approach described in Section 3.6 has been applied for showing its flexibility and effectiveness to the “Crowd safety” [118], “SmartConnectivity” [50] and the “SmartHealth” [134] opportunistic IoT services, which refer to different IoT ecosystems (respectively a Smart City and a Smart Workshop) and pursue different goals (i.e., public safety, connectivity restoring and health monitoring). In particular, these services exploit a smartphone-based mobile IoT Gateway as fundamental building block, since it can continuously collect data from heterogeneous wireless IoT devices and forward them over different communication interfaces and standards.

4.3.1 Smartphone-based Mobile IoT Gateway

Within IoT ecosystems, interoperability among different standards and communication technologies is still a significant challenge that we have started to address by proposing a smartphone-based mobile gateway acting as a flexible and transparent interface between different IoT devices and the Internet [134, 135]. Since smartphones are always connected, have a mass diffusion, are equipped with several communication interfaces (e.g., Wi-Fi, NFC, Bluetooth) and have significant storage and computing capability [136], they are ideal candidates to carry out the delicate task of linking the world of the Internet and the world of “things”, resulting as a fundamental block in making the so called Opportunistic IoT services. We developed an App based on Android OS able to activate all the communication interfaces available on common smartphones in order to collect data from different IoT devices as shown in Figure 4.11. In particular, since smartphones available on the market do not include the ZigBee radio interface, we equipped our smartphones with a Micro SD ZigBee card [137] in order to add a new radio interface well suited for communication with environmental IoT devices such as the widely used Wasp mote [138]. Furthermore, biomedical and lifestyle data can be acquired by setting the communication on the standard Bluetooth radio interface to exchange data with IoT devices such as bangles, pedometers and scales or integrating the SPINE-android framework [116] within the smartphone-centric application to communicate with specific Shimmer [139] wearable sensors well designed to acquire high quality, biophysical and movement data in real-time. The presented solution can continuously collect and forward data coming from wireless IoT devices and sensors transmitting over different communication interfaces and standards; moreover it can send control messages or data flows, such as video streaming, to neighboring IoT devices in an opportunistic fashion. It has been exploited in the use cases presented in the next section.

The general architecture of the smartphone-centric mobile gateway application (Figure 4.12) is mainly constituted by

1. a Management GUI through which the user can receive notifications from IoT devices and sensor nodes and manage them (Graphical User Inter-



Fig. 4.11. Communication scenario for testing the smartphone-centric application.

face module of Figure 4.13 is an implementation of Management GUI. It handles an easy interaction with users, displays all measurements and all settings of the software modules);

2. an Application Services to start a set of different services according to a specific application scenario (Application Services module of Figure 4.13 is an implementation of Application Services. It allows for the starting of different bound services and to manage them through a specific GUI in order to show all the performed measurements and to send the control commands to the IoT devices);
3. a Communication, Coordination and Management Brain (CCMB) able to acquire data from different interfaces and to control several devices. In particular the CCMB module consists of three main logical blocks that can interact with each other as shown in Figure 4.12:
 - The Communication block handles the reception and transmission of messages over the air, and manages the radio duty cycling. It is formed by a series of decoders for incoming packets and a series of encoders for outgoing packets. Each message received or sent is initially handled by the radio controller that provides a common interface on a specific radio adapter that can be dynamically loaded to support several communication technologies such as Bluetooth SMART, ANT+, IEEE 802.15.4, ZigBee, NFC, Wi-Fi, Z-wave. Communication Service Engine module of Figure 4.13 is an implementation of the Communication Block. It creates five bound services (i.e., ZigBee, SPINE, Wi-Fi, Bluetooth and 3G/LTE) in this specific implementation, by handling the logical functionality of access to data and measures. The bound services are based on a standard client-server communication model by

allowing other components and applications to be connected, to send requests and receive replies.

- The IoT Device Management block acts as an interface to the IoT devices by creating periodic timers when the remote sensing operation on the IoT devices is required by a specific user or when a specific command message needs to be send to an IoT device. The controller within this block can handle a variety of IoT devices regardless of their hardware specifications through the appropriate interfaces by implementing the specific high level protocol for each device. This ensures modularity and efficiency. The controller also uses a BufferPool to store the readings that become available for further applications. Finally, the Device Registry contains a list of each active IoT device to connect to in order to receive information or to send control messages. IoT Device Management module of Figure 4.13 is an implementation of the IoT Device Management block. It is in charge of the data interpretation exchange, control commands execution, and dynamic adapters loading of discovered devices. It is formed by three software sub-modules (i) the Protocol Device - it implements the data exchange protocol to which the specific request is addressed (e.g., reset sensor, read accelerometer measure, reset gateway). Furthermore it addresses the request to the specific sensor/platform/gateway by correctly interpreting each specific data frame structure; (ii) IoT Device Board Controller - it is in charge of the dynamic device adapter loading by setting the specific data structure for each new device, adding it to the protocol device module; (iii) Message Handler - it handles the communication between services and GUI by favoring the message passing (e.g. measures, errors, requests failure) from the IoT devices to the graphical interface. It also allows the diffusion of control messages from the GUI to the IoT devices.
- The Coordination Manager block, which derives from [140], is in charge of the management of the interaction between the IoT device Management and Communication modules (the Manager module of Figure 4.13 is an Coordination Manager implementation. It includes the functional software modules. It starts all software blocks by also handling the logic functionalities of the application at a high level). It includes the following blocks (i) Event dispatcher: it triggers when a particular event occurs, such as the discovery of new nodes or a particular alarm (e.g. temperature monitoring and the alarm is triggered when the temperature falls under a specific threshold); (ii) Function Datamodel: it represents the available functionalities for each specific IoT device, such as the independent specification of sampling rates for multiple IoT devices; (iii) Message Datamodel: it is used to forward a message after a particular event or any user action (e.g. it is possible to receive a message with the average value of the performed measurements or a message that contains the new discovered IoT devices or a generic

error message); (iv) IoT network control API : it represents the interface for developers through which it is possible to control, configure and reset the IoT Device.

All the data from these different interfaces are firstly stored on a local SQLite DataBase (see Figure 4.13) before being sent to a remote server throughout a 3G/4G interface in order to be further processed and made available to different experts in specific fields. To support the opportunistic interaction during the mobility of the gateway, a timed scanning procedure has been implemented in order to periodically discovery new IoT devices for data exchange according to the supported bound services.

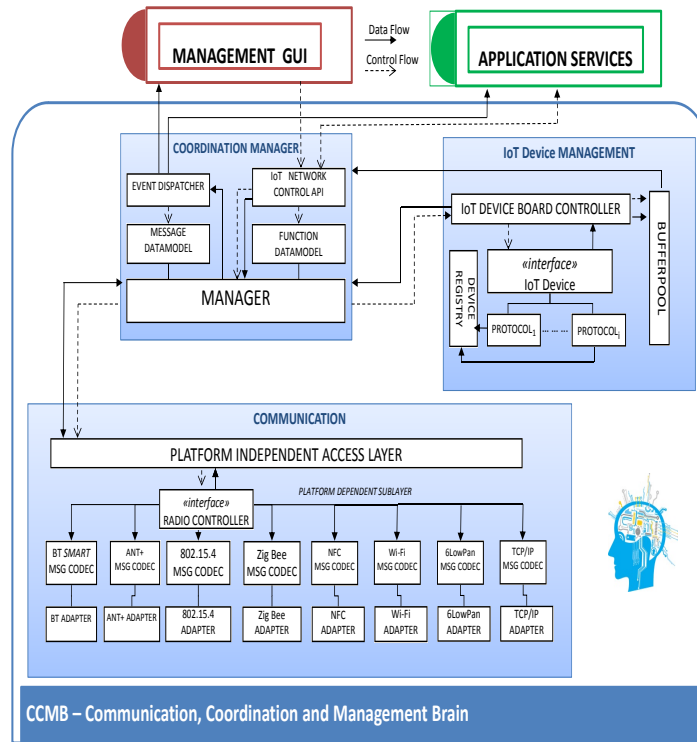


Fig. 4.12. Software architecture of the smartphone-centric gateway.

To evaluate the performances of the proposed solution, we implemented a testbed and conducted several tests. In detail, we used three smartphones (with different hardware and software capabilities listed in Table 4.12) which collect and forward data received from different sensors and IoT devices on different communication interfaces (listed in Table 4.13) for several periods, 6 minutes long. In particular, to better evaluate the system performances, we

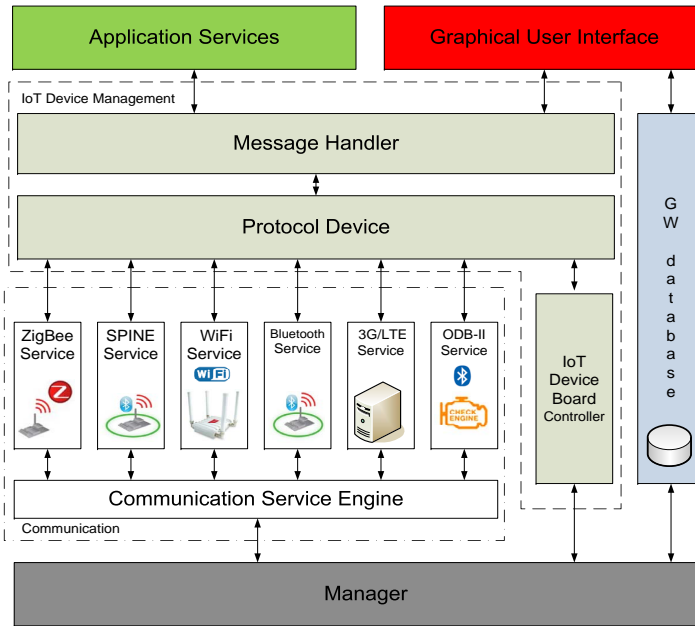


Fig. 4.13. Specific implementation of the smartphone centric gateway application.

decided to repeat the test 10 times with the aim of averaging the traffic load distribution due to the natural asynchronous. We set the confidence interval level to 0:95 and we excluded the first 60 seconds from the statistical error computation in order to verify the correctness of the statistical analysis for the obtained results also reducing the systems transient effects.

Table 4.12. Smartphones used for the testbed.

	Samsung Galaxy S2	Samsung Galaxy S3	Samsung Galaxy S4
CPU	Dual-Core 1.2GHz	Quad-Core 1.4GHz	Quad-Core 1.9GHz
RAM	1GB	1GB	2GB
Battery	1800mAh	2100mAh	2600mAh
Operating System	Android 2.3.3 Gingerbread	Android 4.0.4 Ice Cream Sandwich	Android 4.2.2 Jelly Bean

We evaluated the smartphone performances in terms of energy consumption, memory and CPU usage to further discuss the efficient use of such integrated communication architecture. The GUI of the implemented IoT mobile gateway application is shown in Figure 4.14. It is worth noting that none of the used devices represents the cutting edge in the mobile phones market and none has any specific add-ons, so that they can be considered as representative of a wide range of current common user devices and customers.

Table 4.13. IoT devices connected to the Smartphone-centric application through-out several interfaces.

Interfaces		
Bluetooth	SD-Zigbee	Wi-Fi
2 Garmin Vivofit [141] 1 Scale Beurer 74822 BF 3 Shimmer	3 Waspote XBee ZB-Pro	1 Samsung Smart TV 1 Samsung Air Conditioner

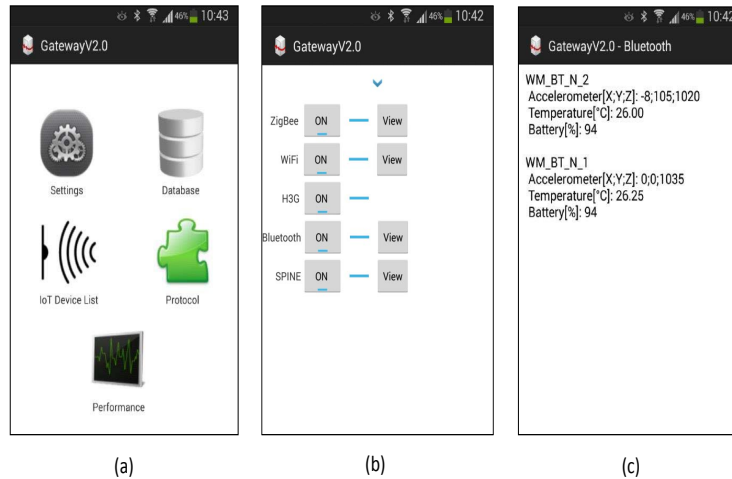
**Fig. 4.14.** Screenshots of the mobile gateway application: a) Main GUI, b) Multiple interfaces choice and activation, c) Data received on a specific interface.

Figure 4.15 shows that the IoT gateway application has a reduced average CPU load regardless of the different tested smartphones; certainly the greater the computational capacity of the device, the greater the percentage of CPU load, however since the average CPU load value is around 15% in case of the most performing smartphone (Galaxy S4), we can argue that the implemented IoT gateway application is fully supported. Regarding to the memory usage, significant results are shown in Figure 4.16, showing that a maximum memory amount of about 85MBytes is required to run the IoT gateway application on the Galaxy S4 smartphone; this value is fully reasonable since that smartphone is equipped with 2GB of memory. Regarding the energy consumption we conducted a 30 minutes long test to make more evident the battery level decrease. Starting with a different battery level for each smartphone model, we experienced quite similar behaviour in terms of battery discharge speed as shown in Figure 4.17. This result is mostly due to the fact that, even if the most performing smartphone is equipped with a more powerful battery,

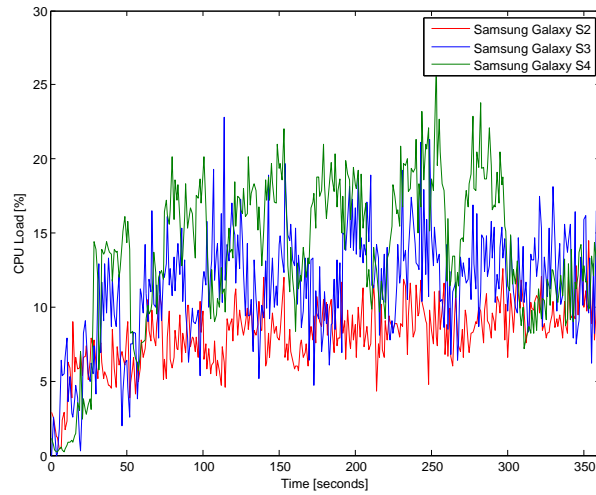


Fig. 4.15. CPU load by activating all the interfaces.

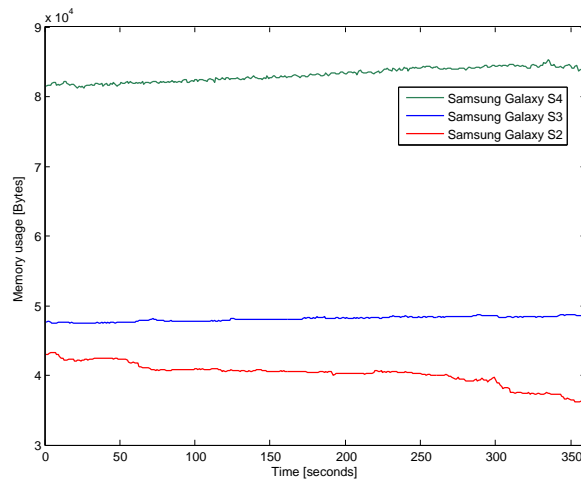


Fig. 4.16. Memory usage by using all the interfaces.

it also has a bigger screen and a higher CPU load due to more performing hardware characteristics offsetting the benefits of having a bigger battery.

In conclusion, the obtained results confirmed the lightweight of the implemented application, that can easily run on common smartphones for several hours. Therefore, such kind of smartphone-based mobile IoT gateway results

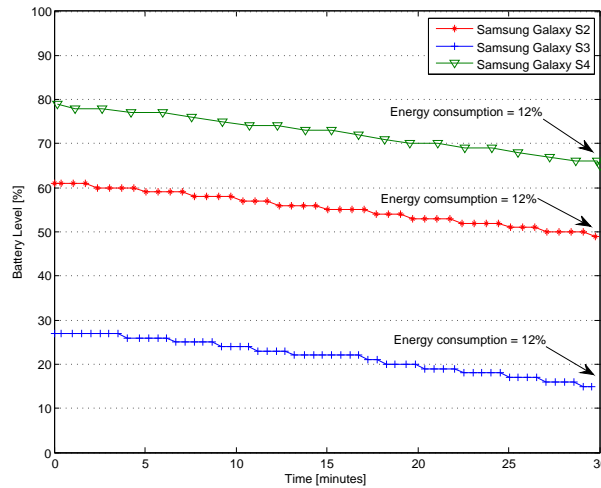


Fig. 4.17. Energy consumption.

suitable to be exploited in real IoT ecosystems, like the one described hereinafter.

4.3.2 Smart City scenario: Crowd Safety service

The Crowd Safety IoT Service considers a mass public event [142], such as the Vienna marathon, and aims at (i) alerting people located nearby overcrowded zones, where any small incident can create a dangerous panic situation; (ii) proposing alternative paths according to the users preferences/constraints (e.g., a tourist, an elder, a biker can receive different suggestions for the same destination customized on their preferences). In details, SOs deployed around the city (e.g., smart traffic lights, and smart lamps) monitor through their embedded devices the flow of athletes and audience, and allow estimating the city zones density. The Crowd Safety IoT Service thus alerts citizens located nearby overcrowded zones by sending a notification on their personal devices. The same alerted citizens can hence specify their destination and receive customized, context-aware, and real-time hints on the best path to be followed. The Crowd Safety is clearly an opportunistic IoT Service because it exposes the four aforementioned opportunistic properties of:

1. *Dynamicity*, since it is activated only if a zones density level exceeds a threshold continuously for a certain amount of time;
2. *Co-located*, since it exploits multiple SOs at the same time for contemporary serving multiple citizens located nearby the overcrowded zones;
3. *Transient*, since it lasts only for an emergency situation and until the citizen is near an overcrowded zone;

4. *Context-aware*, since it considers athletes and audience positions and environmental elements (e.g., a bridge) for determining density and risk levels, as well as citizens positions and their preferences for providing alerts and customized hints.

The opportunistic “Crowd Safety” IoT Service is described according to high-level metamodels (Figure 4.18) and operational models (Figure 4.19). Citizens and Things (namely, IoT Entities) located in Vienna and deployed on its monitored Streets, Squares and Bridges (IoT Environment) are differently involved in the Crowd Safety IoT Service. This comprises three processes for mapping each zone to a risk level (Density calculation), alerting citizens located near overcrowded zones (User Alert), and, if required by the same alerted citizens, providing customized alternative paths for a certain destination (Path Suggestion). The Crowd Safety IoT Service and related processes are better detailed through a Service Profile and a Service Model. The former provides functional specifications (e.g., a citizens position is determined with a precision of 50 meters and they are notified within 10 seconds from their detection near an overcrowded zone), while the second specific preconditions can trigger certain events concretely implementing the Crowd Safety IoT Service (e.g., how a city zone gets matched with its density level). A (simplified but enough expressive) operational model describing the Crowd Safety IoT Service according to the Petri net formalism is depicted in Figure 4.19. In detail, Service Space S_s comprises five service states while six events in Event set Ev represent service preconditions (e.g., the density level should exceed a warning threshold for a period before the zone is considered as being overcrowded) and effects (alert notifications or path suggestions are sent to a citizen who is near a dangerous zone). Even at a first glance, it is evident to see the matching between the concepts of Figures. 4.18 and 4.19. For example, S_0 , S_3 and S_4 depicted in Figure 4.19 are the homonyms processes constituting the Crowd Safety Service Model in Figure 4.18, which encodes, among others, ev_3 as Process Precondition and ev_4 as Process Effect. However, as previously motivated, the metamodels in Figure 4.18 accomplish a descriptive functionality while operational model in Figure 4.19 allows performing the formal verification and simulation of the service.

4.3.3 Smart Workshop scenario: SmartConnectivity and SmartHealth services

Industrial environments can be featured by harsh ambient, featured by excessively values of humidity and temperature, power and gas leakages, etc. These factors negatively impact the reliability of data transmission (and, consequently, a production process), and, most importantly, can endanger the operators’ health. In the context of an industrial Smart Workshop comprising different industrial areas, two opportunistic IoT Services, namely the SmartConnectivity [50] and the SmartHealth services [134], exploit operators’

4.3. Opportunistic IoT services

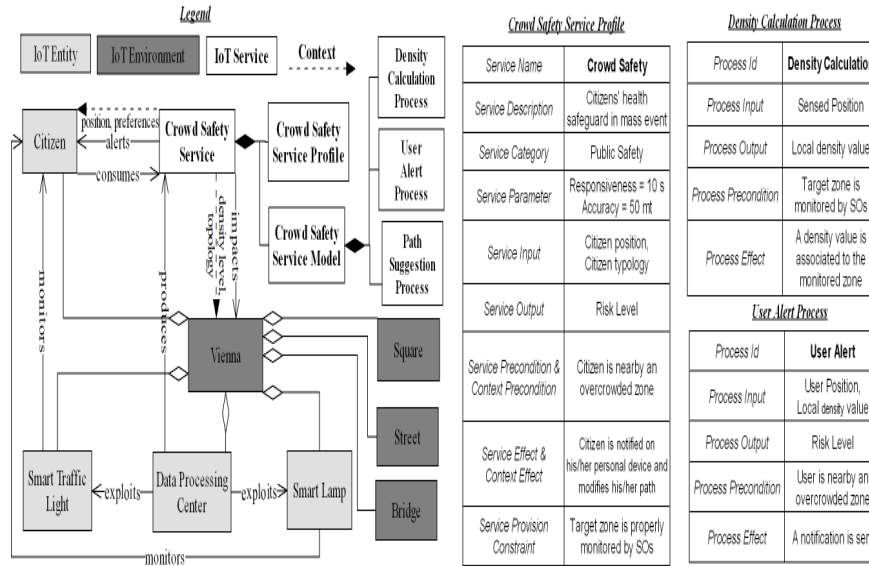


Fig. 4.18. Metamodeling of the Crowd Safety opportunistic IoT service

$Ss = \{s_0 \dots s_4\}$
 S_0 = density calculation
 S_1 = pre-alert
 S_2 = waiting for approaching citizens
 S_3 = user alert - sending notification
 S_4 = path calculation
 $Ev = \{ev_1 \dots ev_6\}$
 ev_1 = density level exceeds warning threshold
 ev_2 = density level still exceeds warning threshold after 1 min.
 ev_3 = citizen approaching overcrowded zone
 ev_4 = alert notification sent
 ev_5 = citizen requires path hints
 ev_6 = path suggestions sent

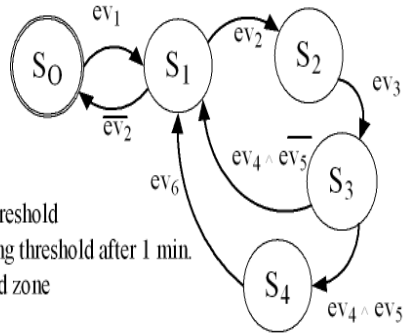


Fig. 4.19. Simplified FSM describing Crowd Safety IoT Service

smartphones respectively with the aims to (i) perform temporary networking for enhancing the workshop connectivity against node failures, and (ii) collect and analyze data from both wearable medical and environmental IoT devices for constantly monitor workers' health and the ambient conditions, potentially reporting dangerous situations. In detail, each operator brings a a mobile smartphone-based IoT gateway like the one described in Section 4.3.1, thus exploiting (i) its WiFi interface for carrying data among the disconnected machines, thus avoiding network fragmentation and implementing the SmartConnectivity service; and (ii) its ZigBee and Bluetooth interfaces for interacting respectively with environmental (temperature, humidity, gas sensors) and wearable medical IoT devices (pulsioximeters, blood and pressure monitors, Fitbit devices), thus implementing the SmartHealth service. Hence, in performing both the tasks of data gathering and data collection, operators' smartphones act as opportunistic, multi-technology, mobile gateways, since they flexibly and transparently interface different IoT devices, otherwise non interacting. Differently from the Crowd Safety, the opportunistic services designed for the Smart Workshop are described only according to the high-level metamodels in order to provide a preliminary overview for the initial service analysis phase. Figure 4.20 indeed reports IoT Entities (the workshop operators along with things, such as the IoT devices) and their role in the SmartConnectivity and the SmartHealth provision within the Smart Workshop environment.

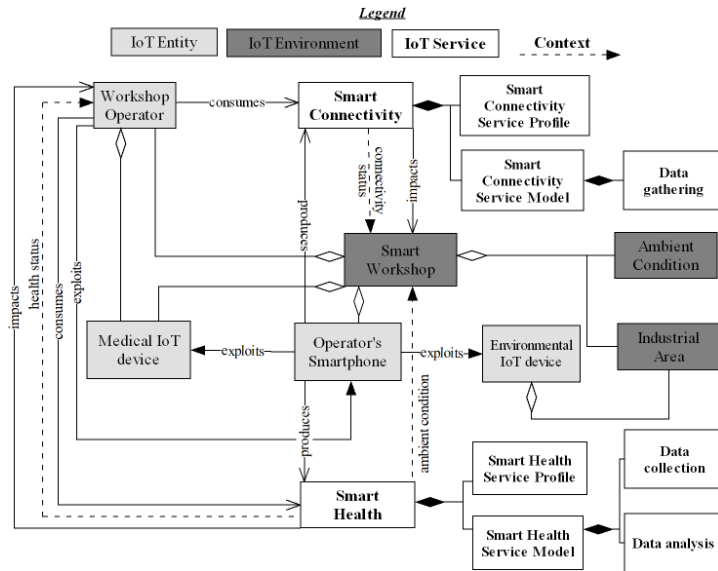


Fig. 4.20. Metamodeling of the SmartHealth and SmartConnectivity opportunistic IoT services

4.4 Summary

This chapter has presented four IoT ecosystems with related use cases, showing how the systematic application of ACOSO-Meth has significantly facilitated and speeded up their development in different phases. In particular: (a) the High-Level SO Metamodel supported the abstract analysis of the main Smart UniCal and SmartDesk features and functionalities, as well as the modeling of the Opportunistic IoT services for the Smart City and Smart Workshop scenarios and finally the SOs inclusion into DLs as novel first-class objects to be collected, managed, and preserved; (b) the agent-oriented design provided the adequate flexibility and effectiveness to fulfill the fundamental requirements at both System- and things- levels of Smart UniCal; finally, (c) the JADE-based implementation allows a rapid and efficient prototyping of the Smart UniCal; this just demands the only effort of programming (by extension) the application specific JACOSO hot-spots that represent only 25% of the overall lines of code constituting the Smart UniCal software. It follows that, because of such modular and extensible approach, most of the code does not need to be customized according to the particular application requirements, but it can be directly reused. Such benefits are not affected by the hand-made transitions among the analysis, design and implementation phases: an automatic transition, whenever possible, may speed up the ACOSO-Meth application while keeping the same effectiveness.

Conclusions, Publications and Future Work

The *Internet of Things* (IoT) can be depicted as a heterogeneous, dense, and open ecosystem rich in contextualized interactions among different entities, such as humans, conventional computing systems and *Smart Objects* (SOs). Because of these features, IoT promises to impact every application context (e.g., transportation, industry, public safety), thus opening novel markets and definitively blurring the line between the physical and virtual worlds. However, although several IoT systems and applications have been already developed, the IoT is still far from unfolding all its potential. Main factors for this delay (which is contributing, by the way, to the proliferation of poorly interoperable “Intranet of Things”) are the lack of standards, the heterogeneity featuring IoT building blocks, and their challenging and articulated development processes. With regard to the last issue, this Thesis provided an application-neutral, full-fledged methodology, named *ACOSO-Meth (Agent-based COoperating Smart Objects Methodology)*, for supporting all the development phases of autonomic and cognitive IoT ecosystems and related services, from analysis to implementation.

During the development of this Thesis, three main contributions have been provided to the IoT research area.

The first important contribution is the definition of a comparison framework comprising IoT fundamental development requirements, raised from a thorough state-of-the-art analysis of IoT platforms, architectures, middleware and methodologies. In particular, we grouped such requirements in System-level requirements (namely, related to the whole distributed system and its development), and Things-level requirements (particularly referring to “things” such as SOs, RFiD, mobile devices, etc.). These requirements recur at the same time and with a substantial prominence within the IoT systems, and allow accommodating all their most important features. Among them, particular emphasis has been given to a novel “scale characterization” requirement, meeting the need of unambiguously classify IoT systems and SOs according to multiple factors, not exclusively attributable to geographical factors. This

comparison framework has inspired the ACOSO-Meth development but can be reused to analyze future work in the field. To the best of our knowledge, such a review work was missing before, and, therefore, it can be considered as the first contribution of this Thesis.

As second contribution, the ACOSO-Meth, designed for fulfilling the aforementioned fundamental System- and Thing-level requirements, has been presented. Based on the jointly exploitation of metamodels, agent-based programming abstractions and middleware, ACOSO-Meth is the first application-neutral, agent-based methodology able to support the main engineering phases of IoT systems and applications. In detail, metamodels (completely decoupled from any specific application) allow highlighting IoT entities main features, functionalities and relationships at different degrees of granularity. Going further the analysis phase, the agent based computing paradigm has a paramount importance because key features of autonomy, proactiveness, intelligence and sociability are necessary, along with agent-based middleware, for designing and implementing autonomic and cognitive SOs and IoT systems. Just autonomic and cognitive properties will be crucial in the future IoT characterization, enabling billion (or, even, trillion) of SOs and IoT systems to perform several self-management actions without a steady human intervention. ACOSO-Meth has been exploited to support the development (from the high-level system analysis to the concrete JADE-based implementation) of the Smart UniCal IoT ecosystem, a complex case study comprising heterogeneous SOs of different scales, deployed in a real scenario (the University of Calabria) and providing cyberphysical services related to structural, indoor space and wellness monitoring. In particular, the JADE-based implementation allowed a rapid and efficient prototyping of the Smart UniCal ecosystem; this demanded the only effort of programming by extension the application-specific JACOSO hotspots, which represent just 25% of the overall lines of code constituting the Smart UniCal software. It follows that, because of such modular and extensible approach, most of the code has been directly and effectively reused, regardless the particular application requirements. The benefits becoming from a full-fledged engineering of the Smart Unical in terms of a multi-agent system, and resulted in a significantly facilitated and speeded up development process, were not affected by the hand-made transitions among the analysis, design and implementation phases: an automatic transition, whenever possible, may speed up the ACOSO-Meth application while keeping the same effectiveness. Furthermore, the ACOSO-Meth approach, and specifically the ACOSO High-Level SO metamodel, has been exploited to drive the inclusion of SOs into Digital Libraries (DLs, namely distributed software infrastructures providing specialized services on a wide range of digital resources) as novel first-class objects to be collected, managed, and preserved. Indeed, during their lifecycle, SOs can produce continuous streams of geolocalized and contextual data and provide different cyberphysical services to their users. Instantiated on the case study of a SmartDesk supporting its user's working activities, the technology-neutral ACOSO High-Level SO metamodel has gen-

erated a well-defined but flexible SO metadata model (specifically following the JSON format), structured so as to be easily manually or automatically created, queried and managed through the facilities provided by the DLs. To the best of our knowledge, this approach represents the first research effort towards the integration of SOs into novel cyberphysical DLs, or Smart DLs (SDLs).

Finally, a further (preliminary) contribution of this Thesis pertains the definition of novel IoT services according to their opportunistic properties, i.e., dynamicity, context-awareness, co-location and transience. Services notably contributed to the spread of Internet, and, likewise, they promise to represent the real drivers for the IoT. Currently available IoT service models, however, consider static environments with established interactions, such that service provisioning is typically customized just according to a users current position or a sensed phenomenon. Such limitations prevent them for being concretely applied because the subsequent crucial phases of their automatic verification, execution and simulation are not properly supported. Indeed, deploying an IoT service is obviously notably complex, time-consuming, and error-prone, while its provision must to completely adhere to the expected terms, even in case of unexpected issues (e.g., network congestion or failures, changes to service working conditions) that are difficult to be described through static metamodels. This motivated us in discussing about a novel, yet actually exploitable, IoT service model, and therefore we have promoted our vision of *Opportunistic IoT Service* and a full-fledged approach to its descriptive metamodeling (particularly suitable for supporting the analysis phase) and operational modeling (particularly suitable for supporting formal verification, execution and simulation). For the first purpose, the ACOSO-Meth has been extended, and specifically the High-Level SO metamodel refined by introducing some categories particularly related to the IoT services. For the second purpose, considering that IoT services interactions are typically (asynchronously) event-driven and time-dependent (namely influenced by the current state and previous history), we have modeled the whole IoT systems as Discrete Event System (DES) and adopted the finite state automaton (FSA) notation to design service operational models. To show the effectiveness and flexibility of the proposed IoT service modeling approach in different contexts, two use cases have been reported, which are related to the Industrial IoT and Smart City, and featured by different scales, purposes, and requirements. This research line is still in an initial phase, so it not presented any tried-and-true solution, but some novel elements to advance the state-of-the-art and some interesting insights for the future work.

5.1 Publications related with this Thesis

This Thesis is based on several articles published in international workshops, conferences, books, and journals. The research work related to this Thesis has resulted in 15 publications, including:

- 3 journal articles (3 with ISI impact factor);
- 10 conference papers;
- 2 book chapters;

In the following, the publications are organized according to their publication venues and a brief description of each publication is provided.

5.1.1 Journal Articles

- **Modeling and Simulating Internet-of-Things Systems: A Hybrid Agent-Oriented Approach.** [35]:

Fortino, G., Gravina, R., Russo, W., and Savaglio, C. Modeling and Simulating Internet-of-Things Systems: A Hybrid Agent-Oriented Approach. *Computing in Science & Engineering*, 19(5):68-76. 2017.

This paper is a significant extension of [109], [110] and [103]. In particular, it presents relevant mapping guidelines that provide a guidance to IoT system designers during the transition from modeling to simulation.

- **Enabling IoT Interoperability through Opportunistic Smartphone-based Mobile Gateways.** [134]:

Aloi, G., Caliciuri, G., Fortino, G., Gravina, R., Pace, P., Russo, W., and Savaglio, C. Enabling IoT interoperability through opportunistic smartphone-based mobile gateways. In *Journal of Network and Computer Applications*, 81:74-84. 2017.

This paper extends the contents of [135] adding (i) new discussions on the requirements for the interoperability of the proposed opportunistic mobile smartphone-based gateway, (ii) more details on the architecture of the developed application, (iii) realistic use cases based on funded ongoing research projects, and (iv) new results in high load data traffic scenario.

- **Agent-Oriented Cooperative Smart Objects: from IoT System Design to Implementation.** [3] :

Fortino, G., Russo, W., Savaglio, C., Shen, W., and Zhou, M. Agent-Oriented Cooperative Smart Objects: from IoT System Design to Implementation. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, PP(99):1-18. 2018.

The paper first elicits and discusses main IoT ecosystem development requirements, then proposes a full-fledged approach to their development based on ACOSO-Meth and its related middleware. Finally, the application of ACOSO-Meth for the development of a complex IoT ecosystem highlights the effectiveness and the efficiency of the proposed approach.

5.1.2 Book Chapters

- **Towards Multi-layer Interoperability of Heterogeneous IoT Platforms: The INTER-IoT Approach.** [63]:

Fortino, G., Savaglio, C., Palau, C. E., de Puga, J. S., Ghanza, M., Paprzycki, M., Montesinos, M., Liotta, A., and Llop, M. Towards Multi-layer Interoperability of Heterogeneous IoT Platforms: The INTER-IoT Approach. In *Integration, Interconnection, and Interoperability of IoT Systems*, 199-232, Springer, Cham. 2018.

This work presents the INTER-IoT systemic approach, which is being created within the INTER-IoT project together with necessary software tools and end-user applications. Solutions for overcoming interoperability problems across the communication/software stacks of heterogeneous IoT platforms are discussed, aiming at facilitating the reuse and integration of current IoT platforms with existing and future (even standard) IoT ecosystems.

- **Towards Cyberphysical Digital Libraries: Integrating IoT Smart Objects into Digital Libraries.** [69]:

Fortino, G., Rovella, A., Russo, W., and Savaglio, C. Towards Cyberphysical Digital Libraries: Integrating IoT Smart Objects into Digital Libraries. In *Management of Cyber Physical Objects in the Future Internet of Things*, 135-156. Springer International Publishing. 2015.

This paper is an extension of our previous works [68] and [67]. In particular, to foster the SOs inclusion into DLs, a mapping between Digital Library Reference Model (DLRM, namely the main reference model for architecting DLs) and the proposed SO metadata model is presented, showing the alignment between their key concepts and thus supporting the idea of treating SOs as novel and valuable first-class DL resources.

5.1.3 Conference Papers

- **Opportunistic Cyberphysical Services: A Novel Paradigm for the Future Internet of Things.** [?]:

Fortino, G., Savaglio, C., Zhou, M. Opportunistic Cyberphysical Services: A Novel Paradigm for the Future Internet of Things. *The 4th IEEE World Forum on the Internet of Things (WF-IoT 2018)*, February 2018.

This paper summarizes our previous contributions in the novel research context of IoT services [50], [118] and promotes our vision of “Opportunistic IoT Services” along with a full-fledged approach to their modeling. Its effectiveness and flexibility is illustrated by means of two case studies, related to the Industrial IoT and Smart City scenarios.

- **Agent-based Computing in the Internet of Things: a Survey.** [29]:

Fortino, G., Savaglio, C., Ghanza, M., Paprzycki, M., Badica C., and Ivanovic, M. Agent-based computing in the Internet of Things: a survey. *International Symposium on Intelligent*

and Distributed Computing, 307-320. Springer, Cham. October 2017.

This paper shows how Agent-Based Computing (ABC) paradigm has been effectively exploited for modeling, programming and simulating IoT systems. Hence, main ABC's concepts, metaphors, techniques, and methods applied so far for the IoT system development are surveyed, and a critical analysis of their common misapplications and misconceptions reported.

- **A Mobile Multi-technology Gateway to Enable IoT Interoperability.** [135]:

Aloi, G., Caliciuri, G., Fortino, G., Gravina, R., Pace, P., Russo, W., and Savaglio, C. A mobile multi-technology gateway to enable IoT interoperability. In *Internet-of-Things Design and Implementation (IoTDI), 2016 IEEE First International Conference on*, 259-264. IEEE. 2016.

This paper introduces a mobile gateway solution to support IoT interoperability through a multi-standard, multi-interface and multi-technology smartphone, that becomes a universal interface between the Internet and the Things. The feasibility of the proposed solution has been tested by implementing a specific testbed and evaluating gateway performances in continuously collecting and forwarding data coming from heterogeneous wireless IoT devices and sensors.

- **Modeling Opportunistic IoT Services in Open IoT Ecosystems.** [118]:

Fortino, G., Savaglio, C., Zhou, M. Modeling Opportunistic IoT Services in Open IoT Ecosystems. *Proc. 18th Workshop Objects to Agents (WOA17)*, 90-95. July 2017.

This paper notably extends [50] by proposing detailed descriptive meta-models (suitable to the analysis phase) and operational models (suitable to the implementation and verification phases) aiming to fully support IoT service development according to opportunistic properties, i.e., dynamicity, context-awareness, co-location and transience. The application of the proposed modeling approach is shown in a concrete case study (public safety during a mass events evolution).

- **Toward Opportunistic Services for the Industrial Internet of Things.** [50]:

Fortino, G., Savaglio, C., Zhou, M. Toward Opportunistic Services for the Industrial Internet of Things. *Proceedings of 13th IEEE Conference on Automation Science and Engineering (CASE)*, 825-830. IEEE. August 2017.

This paper surveys the state-of-the-art of both developer- and enterprise-oriented IoT service models, analyses their limitations, and presents a first contribute towards the definition of a novel "Opportunistic IoT Service" paradigm. A particular emphasis is given to the Industrial IoT (IIoT) context, reporting a case study related to an opportunistic IoT service that provides a reliable data transmission in a smart factory.

Towards Interoperable, Cognitive and Autonomous IoT Systems: An Agent-based Approach. [103]:

Savaglio, C., Fortino, G., and Zhou, M. Towards interoperable, cognitive and autonomous IoT systems: An agent-based approach. In *Internet of Things (WF-IoT), 2016 IEEE 3rd World Forum on*, 58-63. IEEE. December 2016.

This paper first discusses the importance of the synergic application of paradigms such as agent-based computing, cognitive networks and autonomous computing, in order to pave the way toward an interoperable IoT ecosystem, and then demonstrates how the ACOSO middleware (whose performance verification has been made through the OMNeT++ simulator) can support the development of distributed and self-steering IoT systems.

- **Simulation of Agent-Oriented Internet of Things Systems.** [110]:

Fortino, G., Russo, W., and Savaglio, C. Simulation of Agent-Oriented Internet of Things Systems. *Proc. 17th Workshop Objects to Agents (WOA16)*, 8-13. 2016.

This paper presents a preliminary work related to the simulation of agent-oriented IoT systems in small-medium-large scale scenarios through the OMNeT++ simulation platform. A specific attention has been devoted to the “inter-Things” communications phase, by characterizing the three simulation scenarios according to their SO population and related distribution in a different number of subnetworks, and, therefore, by testing different transport protocols, performance metrics, communication parameters, and message exchange patterns.

- **Agent-oriented Modeling and Simulation of IoT Networks.** [109]:

Fortino, G., Russo, W., and Savaglio, C. Agent-oriented modeling and simulation of IoT networks. *2016 Federated Conference on Computer Science and Information Systems (FedC-SIS)*, 90-95. IEEE. September 2016.

This paper proposes the agent-oriented modeling of IoT networks (through the ACOSO-based Smart Object model) and their simulation (through the INET extension for the OMNeT++ network simulator) with the goal of evaluating performances and validating network design choices.

- **Autonomic and Cognitive Architectures for the Internet of Things.** [36]:

Savaglio C., and Fortino, G. Autonomic and Cognitive Architectures for the Internet of Things. In *International Conference on Internet and Distributed Computing Systems*, 9258: 39-47. G. Di Fatta, G. Fortino, W. Li, M. Pathan, F. Stahl, and A. Guerrieri, Eds. Springer International Publishing 2015.

This paper reviews the current trends in IoT management architectures, inspecting the underlying motivations and framing the current state-of-the-art of the most relevant autonomic and cognitive architectures. Indeed, the countless challenges and opportunities that the development of such an ecosystem entails require a marked intervention on the current Internet

architectural frameworks and models, primarily as regards the management function.

- **Towards a Development Methodology for Smart Object-oriented IoT Systems: A Metamodel Approach.** [104]:

Fortino, G., Guerrieri, A., Russo, W., and Savaglio, C. Towards a development methodology for smart object-oriented IoT systems: A metamodel approach. In *Systems, Man, and Cybernetics (SMC), 2015 IEEE International Conference on*, 1297-1302. IEEE. October 2015.

This paper presents a preliminary software engineering approach aiming to support a systematic development of SOSs-based IoT systems, from the high-level design to its agent-based implementation. Based on metamodels defined at different levels of abstraction and instantiated on an example case study, this work represents the first building block toward the definition of a generic, agent-based, full-fledged methodology resulting in the ACOSO-Meth [3].

5.2 Future Work

Some research directions related with this Thesis and deserving further efforts are still being explored. Therefore, in the following, on-going activities (short-term perspective and technological approach) and future research lines (medium-long term perspective and methodological approach) are presented.

An interesting activity is currently devoted to the development of the following strategic Smart UniCal services: an RFID-based people counting/identification system (smartTrack service) for SmartBridge, an RFID-based inventory system for SmartDIMES valuable stuff (smartInventory service), and an NFC-based system to automatically record SenSysCal users' attendances and their daily timetable (smartAttendance service).

Another relevant on-going work is the definition of well-formalized and automatic translation rules for driving the transition from IoT system modeling to simulation. By automatically mapping an agent-based SO to an INET node, scalability and efficiency of our hybrid simulation approach of IoT ecosystems could be notably enhanced.

Furthermore, we are actually working on re-investing some of the research contribution presented in this thesis, especially the smartphone-based mobile IoT gateway and the guidelines provided by the ACOSO-Meth, in the European H2020 INTER-IoT project (that already supported most of the presented work and exploited several of the provided contributions).

On the basis of the achieved results and on the on-going research activities, a number of new research directions have been envisaged.

With respect to ACOSO-Meth, future work is already planned towards four main lines: (i) extending the current ACOSO-based approach to sup-

port the BDI (Belief-Desire-Intention) paradigm, the topic-based communication among different platforms by means of dedicated “mediator” agents, and creation of a tool for the automatic models instantiation and code generation; (ii) integrating Cloud/Edge computing with the ACOSO-based approach to enhance system scalability and enable more critical real-time system responses; (iii) defining mechanisms to automatize, where possible, the transition among the different phases constituting the methodology, thus speeding up the ACOSO-Meth application while keeping the same effectiveness, and (iv) using data validation models for IoT domains.

A promising future work intend to implement the proposed approach inclusion for the SOs in DLs through the exploitation of a real DL management system such as Fedora and/or DSpace.

Finally, future research efforts will focus on an integrated framework for supporting the formal verification, simulation and implementation of IoT services before their deployment. Formal methods and verification tools, e.g., Petri nets and their extensions for dealing with real time and stochastic systems, will be explored and applied. Additionally, we plan to investigate the application of Aggregate Computing techniques to foster the “collective adaptive” character of opportunistic IoT services.

References

- [1] Stefan Meissner, Dan Dobre, Matthias Thoma, and Gregorio Martin. Internet of things architecture iot-a project deliverable d2. 1-resource description specification.
- [2] Jian Huang, F Bastani, I-L Yen, Jing Dong, Wenke Zhang, F-J Wang, and H-J Hsu. Extending service model to build an effective service composition framework for cyber-physical systems. In *Service-Oriented Computing and Applications (SOCA), 2009 IEEE International Conference on*, pages 1–8. IEEE, 2009.
- [3] Giancarlo Fortino, Wilma Russo, Claudio Savaglio, Weiming Shen, and Mengchu Zhou. Agent-oriented cooperative smart objects: from iot system design to implementation. *IEEE Transactions on System, Man and Cybernetics*, PP.(9):1–18, 2017.
- [4] Kashif Sana Dar, Amir Taherkordi, and Frank Eliassen. Enhancing dependability of cloud-based iot services through virtualization. In *Internet-of-Things Design and Implementation (IoTDI), 2016 IEEE First International Conference on*, pages 106–116. IEEE, 2016.
- [5] Sonja Meyer, Klaus Sperner, Carsten Magerkurth, and Jacques Pasquier. Towards modeling real-world aware business processes. In *Proc. of the Second Intl. Workshop on Web of Things*, page 8. ACM, 2011.
- [6] Maryam Davoudpour, Alireza Sadeghian, and Hossein Rahnama. “canthings” (context aware network for the design of connected things) service modeling based on timed cpn. In *Semantic Computing (ICSC), 2015 IEEE International Conference on*, pages 127–130. IEEE, 2015.
- [7] Klaus Sperner, Sonja Meyer, and Carsten Magerkurth. Introducing entity-based concepts to business process modeling. In *International Workshop on Business Process Modeling Notation*, pages 166–171. Springer, 2011.
- [8] Daniele Miorandi, Sabrina Sicari, Francesco De Pellegrini, and Imrich Chlamtac. Internet of things: Vision, applications and research challenges. *Ad Hoc Networks*, 10(7):1497–1516, 2012.
- [9] Gerd Kortuem, Fahim Kawsar, Daniel Fitton, and Vasughi Sundramoorthy. Smart objects as building blocks for the internet of things. *Internet Computing, IEEE*, 14(1):44–51, 2010.

- [10] Friedemann Mattern and Christian Floerkemeier. From the internet of computers to the internet of things. In *From active data management to event-based systems and more*, pages 242–259. Springer, 2010.
- [11] Jayavardhana Gubbi, Rajkumar Buyya, Slaven Marusic, and Marimuthu Palaniswami. Internet of things (iot): A vision, architectural elements, and future directions. *Future Generation Computer Systems*, 29(7):1645–1660, 2013.
- [12] Terrell R Bennett, Claudio Savaglio, David Lu, Hunter Massey, Xianan Wang, Jian Wu, and Roozbeh Jafari. Motionsynthesis toolset (most): a toolset for human motion data synthesis and validation. In *Proceedings of the 4th ACM MobiHoc workshop on Pervasive wireless healthcare*, pages 25–30. ACM, 2014.
- [13] Pankesh Patel and Damien Cassou. Enabling high-level application development for the internet of things. *Journal of Systems and Software*, 103:62–84, 2015.
- [14] Nicholas R Jennings. Agent-based computing: Promise and perils. 1999.
- [15] Giancarlo Fortino, Antonio Guerrieri, Wilma Russo, and Claudio Savaglio. Integration of agent-based and cloud computing for the smart objects-oriented iot. In *Computer Supported Cooperative Work in Design (CSCWD), Proceedings of the 2014 IEEE 18th International Conference on*, pages 493–498. IEEE, 2014.
- [16] Jeffrey O Kephart and David M Chess. The vision of autonomic computing. *Computer*, 36(1):41–50, 2003.
- [17] Carolina Fortuna and Mihael Mohorcic. Trends in the development of communication networks: Cognitive networks. *Computer networks*, 53(9):1354–1376, 2009.
- [18] Luigi Atzori, Antonio Iera, and Giacomo Morabito. The internet of things: A survey. *Computer networks*, 54(15):2787–2805, 2010.
- [19] Giancarlo Fortino, Antonio Guerrieri, and Wilma Russo. Agent-oriented smart objects development. In *Computer Supported Cooperative Work in Design (CSCWD), 2012 IEEE 16th International Conference on*, pages 907–912. IEEE, 2012.
- [20] Kai Lin, Min Chen, Jing Deng, Mohammad Mehedi Hassan, and Giancarlo Fortino. Enhanced fingerprinting and trajectory prediction for iot localization in smart buildings. *IEEE Transactions on Automation Science and Engineering*, 13(3):1294–1307, 2016.
- [21] Michael Wooldridge. *An introduction to multiagent systems*. John Wiley & Sons, 2009.
- [22] Alessandro Ricci and Andrea Santi. Agent-oriented computing: Agents as a paradigm for computer programming and software development. In *Proc. of the 3rd Intl Conf. on Future Computational Technologies and Applications. Wilmington: Xpert Publishing Services*, pages 42–51, 2011.
- [23] Stefan Poslad. Specifying protocols for multi-agent systems interaction. *ACM Transactions on Autonomous and Adaptive Systems (TAAS)*, 2(4):15, 2007.
- [24] ACL Fipa. Fipa acl message structure specification. *Foundation for Intelligent Physical Agents*, <http://www.fipa.org/specs/fipa00061/SC00061G.html> (30.6. 2004), 2002.
- [25] Charles M Macal and Michael J North. Tutorial on agent-based modeling and simulation. In *Simulation Conference, 2005 Proceedings of the Winter*, pages 14–pp. IEEE, 2005.

-
- [26] Federico Bergenti, Marie-Pierre Gleizes, and Franco Zambonelli. *Methodologies and software engineering for agent systems: the agent-oriented software engineering handbook*, volume 11. Springer Science & Business Media, 2006.
- [27] Michael J Wooldridge and Nicholas R Jennings. Software engineering with agents: Pitfalls and pratfalls. *IEEE Internet Computing*, 3(3):20–27, 1999.
- [28] Hyacinth S Nwana and Divine T Ndumu. A perspective on software agents research. *The Knowledge Engineering Review*, 14(2):125–142, 1999.
- [29] Claudio Savaglio, Giancarlo Fortino, Maria Ganzha, Marcin Paprzycki, Costin Bădică, and Mirjana Ivanović. Agent-based computing in the internet of things: A survey. In *International Symposium on Intelligent and Distributed Computing*, pages 307–320. Springer, 2017.
- [30] Antonio Liotta. The cognitive net is coming. *IEEE Spectrum*, 50(8):26–31, 2013.
- [31] Michael N Huhns and Munindar P Singh. Service-oriented computing: Key concepts and principles. *IEEE Internet computing*, 9(1):75–81, 2005.
- [32] Huawei Publications. Using internet of things to create product-service hybrids, <http://e.huawei.com/us/publications/global/ictinsights/>.
- [33] Gheorghe H Popescu. The economic value of the industrial internet of things. *Journal of Self-Governance and Management Economics*, 3(2):86–91, 2015.
- [34] Matthias Thoma, Sonja Meyer, Klaus Sperner, Stefan Meissner, and Torsten Braun. On iot-services: Survey, classification and enterprise integration. In *Green Computing and Communications (GreenCom), 2012 IEEE International Conference on*, pages 257–260. IEEE, 2012.
- [35] Giancarlo Fortino, Raffaele Gravina, Wilma Russo, and Claudio Savaglio. Modeling and simulating internet-of-things systems: A hybrid agent-oriented approach. *Computing in Science & Engineering*, 19(5):68–76, 2017.
- [36] Claudio Savaglio and Giancarlo Fortino. Autonomic and cognitive architectures for the internet of things. In *International Conference on Internet and Distributed Computing Systems*, pages 39–47. Springer, 2015.
- [37] Integrating the Physical with the Digital World of the Network of the Future SENSEI project. www.sensei-project.eu.
- [38] AIOTI (Alliance for IoT Innovation) project. www.aioti.eu.
- [39] Future Internet ware (FIWARE) project. www.fiware.org.
- [40] IEEE P2413 Standard for an Architectural Framework for the IoT. standards.ieee.org/develop/project/2413.html.
- [41] Semantic Sensor Network XG Final Report W3C Semantic Sensor Network Incubator Group. <http://www.w3.org/2005/incubator/ssn>.
- [42] Anis Charfi, Benjamin Schmeling, Francesco Novelli, Heiko Witteborg, and Uwe Kylau. An overview of the unified service description language. In *Web Services (ECOWS), 2010 IEEE 8th European Conference on*, pages 173–180. IEEE, 2010.
- [43] Business Process Model. Notation (bpmn) version 2.0. *OMG Specification, Object Management Group*, 2011.
- [44] Suparna De, Payam Barnaghi, Martin Bauer, and Stefan Meissner. Service modelling for the internet of things. In *Computer Science and Information Systems (FedCSIS), 2011 Federated Conference on*, pages 949–955. IEEE, 2011.
- [45] David Martin, Mark Burstein, Jerry Hobbs, Ora Lassila, Drew McDermott, Sheila McIlraith, Srini Narayanan, Massimo Paolucci, Bijan Parsia, Terry Payne, et al. Owl-s: Semantic markup for web services. *W3C member submission*, 22:2007–04, 2004.

-
- [46] Stephan Haller, Alexandru Serbanati, Martin Bauer, and Francois Carrez. A domain model for the internet of things. In *Green Computing and Communications (GreenCom), 2013 IEEE and Internet of Things (iThings/CPSCoM), IEEE International Conference on and IEEE Cyber, Physical and Social Computing*, pages 411–417. IEEE, 2013.
- [47] Mohd Anuaruddin Bin Ahmadon and Shingo Yamaguchi. On service personalization analysis for the internet of me based on pn2. In *Consumer Electronics (ICCE), 2016 IEEE International Conference on*, pages 413–416. IEEE, 2016.
- [48] Richard Zurawski and MengChu Zhou. Petri nets and industrial applications: A tutorial. *IEEE Transactions on industrial electronics*, 41(6):567–583, 1994.
- [49] Li Da Xu, Wu He, and Shancang Li. Internet of things in industries: A survey. *IEEE Transactions on industrial informatics*, 10(4):2233–2243, 2014.
- [50] Giancarlo Fortino, Claudio Savaglio, and Mengchu Zhou. Toward opportunistic services for the industrial internet of things. In *Proceedings of 2017 13th IEEE Conference on Automation Science and Engineering (CASE)*, pages 825–830, 2017.
- [51] Rahul Ravi and Li-Chung Wu. *Demystifying Industry 4.0 Implications of Internet of Things and Services for the Chemical Industry*. PhD thesis, 2016.
- [52] Paul Daugherty, Prith Banerjee, Walid Negm, and Allan E Alter. Driving unconventional growth through iiot, 2015. www.accenture.com/us-en/labs-insight-industrial-internet-of-things.
- [53] Alexander Salinas Segura and Norbert Vicari SAG. Internet of things architecture iot-a project deliverable d6. 2–updated requirements list. 2011.
- [54] Giancarlo Fortino, Antonio Guerrieri, Wilma Russo, and Claudio Savaglio. Middlewares for smart objects and smart environments: overview and comparison. In *Internet of Things Based on Smart Objects*, pages 1–27. Springer, 2014.
- [55] Soma Bandyopadhyay, Munmun Sengupta, Souvik Maiti, and Subhajit Dutta. Role of middleware for internet of things: A study. *International Journal of Computer Science and Engineering Survey*, 2(3):94–105, 2011.
- [56] B Clifford Neuman. Scale in distributed systems. *Readings in Distributed Computing Systems.*, pages 1–28, 1994.
- [57] Elisabetta Cortese, Filippo Quarta, Giosue Vitaglione, and P Vrba. Scalability and performance of jade message transport system. In *AAMAS workshop on agentcities, Bologna*, volume 16, page 28, 2002.
- [58] Stefano Galzarano, Claudio Savaglio, Antonio Liotta, and Giancarlo Fortino. Gossiping-based aodv for wireless sensor networks. In *Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on*, pages 26–31. IEEE, 2013.
- [59] Moez Esseghir and Nizar Bouabdallah. Node density control for maximizing wireless sensor network lifetime. *International Journal of Network Management*, 18(2):159–170, 2008.
- [60] Eiko Yoneki. Evolution of ubiquitous computing with sensor networks in urban environments. In *Ubiquitous computing conference, metapolis and urban life workshop proceedings*, pages 56–59, 2005.
- [61] Debasis Bandyopadhyay and Jaydip Sen. Internet of things: Applications and challenges in technology and standardization. *Wireless Personal Communications*, 58(1):49–69, 2011.

-
- [62] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Geor-gakopoulos. Context aware computing for the internet of things: A survey. *IEEE Communications Surveys & Tutorials*, 16(1):414–454, 2014.
- [63] Giancarlo Fortino, Claudio Savaglio, Carlos E Palau, Jara Suarez de Puga, Maria Ghanza, Marcin Paprzycki, Miguel Montesinos, Antonio Liotta, and Miguel Llop. Towards multi-layer interoperability of heterogeneous iot plat-forms: The inter-iot approach. In *Integration, Interconnection, and Interoper-ability of IoT Systems*, pages 199–232. Springer, 2018.
- [64] Fahim Kawsar, Tatsuo Nakajima, Jong Hyuk Park, and Sang-Soo Yeo. De-sign and implementation of a framework for building distributed smart object systems. *The Journal of Supercomputing*, 54(1):4–28, 2010.
- [65] Giancarlo Fortino, Marco Lackovic, Wilma Russo, and Paolo Trunfio. A dis-covery service for smart objects over an agent-based middleware. In *Inter-national Conference on Internet and Distributed Computing Systems*, pages 281–293. Springer, 2013.
- [66] Alessandro Bassi, Martin Bauer, Martin Fiedler, Thorsten Kramp, Rob Van Kranenburg, Sebastian Lange, and Stefan Meissner. *Enabling things to talk*. Springer, 2016.
- [67] Giancarlo Fortino, Anna Rovella, Wilma Russo, and Claudio Savaglio. On the classification of cyberphysical smart objects in the internet of things. In *UBICITEC*, 2014.
- [68] Giancarlo Fortino, Anna Rovella, Wilma Russo, and Claudio Savaglio. In-cluding cyberphysical smart objects into digital libraries. In *International Conference on Internet and Distributed Computing Systems*, pages 147–158. Springer, 2014.
- [69] Giancarlo Fortino, Anna Rovella, Wilma Russo, and Claudio Savaglio. To-wards cyberphysical digital libraries: Integrating iot smart objects into digital libraries. In *Management of Cyber Physical Objects in the Future Internet of Things*, pages 135–156. Springer, 2016.
- [70] Christos Goumopoulos and Achilles Kameas. Smart objects as components of ubicomp applications. *International Journal of Multimedia and Ubiquitous Engineering*, 4(3):1–20.
- [71] Patricia Derler, Edward A Lee, and Alberto Sangiovanni Vincentelli. Modeling cyber-physical systems. *Proceedings of the IEEE*, 100(1):13–28, 2012.
- [72] Takumi Kato, Ryo Chiba, Hideyuki Takahashi, and Tetsuo Kinoshita. Agent-oriented cooperation of iot devices towards advanced logistics. In *Computer Software and Applications Conference (COMPSAC), 2015 IEEE 39th Annual*, volume 3, pages 223–227. IEEE, 2015.
- [73] Michele Ruta, Floriano Scioscia, Giuseppe Loseto, and Eugenio Di Sciascio. Semantic-based resource discovery and orchestration in home and building automation: A multi-agent approach. *IEEE Transactions on Industrial Inform-atics*, 10(1):730–741, 2014.
- [74] Xiangyu Zhang, Rajendra Adhikari, Manisa Pipattanasomporn, Murat Kuzlu, and Saifur Rahman Bradley. Deploying iot devices to make buildings smart: Performance evaluation and deployment experience. In *Internet of Things (WF-IoT), 2016 IEEE 3rd World Forum on*, pages 530–535. IEEE, 2016.
- [75] Alexis Morris, Paolo Giorgini, and Sameh Abdel-Naby. Simulating bdi-based wireless sensor networks. In *Proceedings of the 2009 IEEE/WIC/ACM International Joint Conference on Web Intelligence and Intelligent Agent Technology-Volume 02*, pages 78–81. IEEE Computer Society, 2009.

- [76] Artem Katasonov, Olena Kaykova, Oleksiy Khriyenko, Sergiy Nikitin, and Vagan Y Terziyan. Smart semantic middleware for the internet of things. *ICINCO-ICSO*, 8:169–178, 2008.
- [77] Vagan Terziyan, Olena Kaykova, and Dmytro Zhovtobryukh. Ubiroad: Semantic middleware for context-aware smart road environments. In *Internet and web applications and services (icw)*, 2010 fifth international conference on, pages 295–302. IEEE, 2010.
- [78] Panagiotis Vlacheas, Raffaele Giaffreda, Vera Stavroulaki, Dimitris Kelaidonis, Vassilis Foteinos, George Poullos, Panagiotis Demestichas, Andrey Somov, Abdur Rahim Biswas, and Klaus Moessner. Enabling smart cities through a cognitive management framework for the internet of things. *IEEE communications magazine*, 51(6):102–111, 2013.
- [79] Franco Cicirelli, Antonio Guerrieri, Giandomenico Spezzano, Andrea Vinci, Orazio Briante, and Giuseppe Ruggieri. isapiens: A platform for social and pervasive smart environments. In *Internet of Things (WF-IoT), 2016 IEEE 3rd World Forum on*, pages 365–370. IEEE, 2016.
- [80] Giancarlo Fortino, Antonio Guerrieri, Michelangelo Lacopo, Matteo Lucia, and Wilma Russo. An agent-based middleware for cooperating smart objects. In *International Conference on Practical Applications of Agents and Multi-Agent Systems*, pages 387–398. Springer, 2013.
- [81] Franco Cicirelli, Giancarlo Fortino, Antonio Guerrieri, Giandomenico Spezzano, and Andrea Vinci. Metamodeling of smart environments: from design to implementation. *Advanced Engineering Informatics*, 2016.
- [82] Qihui Wu, Guoru Ding, Yuhua Xu, Shuo Feng, Zhiyong Du, Jinlong Wang, and Keping Long. Cognitive internet of things: a new paradigm beyond connection. *IEEE Internet of Things Journal*, 1(2):129–143, 2014.
- [83] Luciano Baresi, Antonio Di Ferdinando, Antonio Manzalini, and Franco Zambonelli. The cascadas framework for autonomic communications. *Autonomic Communication*, pages 147–168, 2009.
- [84] Maciej A Bossak. Simulation based design. *Journal of Materials Processing Technology*, 76(1):8–11, 1998.
- [85] Stamatis Karnouskos and Thiago Nass De Holanda. Simulation of a smart grid city with software agents. In *Computer Modeling and Simulation, 2009. EMS’09. Third UKSim European Symposium on*, pages 424–429. IEEE, 2009.
- [86] Luca Costantino, Novella Buonaccorsi, Claudio Cicconetti, and Raffaella Mambrini. Performance analysis of an lte gateway for the iot. In *World of Wireless, Mobile and Multimedia Networks (WoWMoM), 2012 IEEE International Symposium on a*, pages 1–6. IEEE, 2012.
- [87] Gabriele D’Angelo, Stefano Ferretti, and Vittorio Ghini. Multi-level simulation of internet of things on smart territories. *Simulation Modelling Practice and Theory*, 2016.
- [88] Wang Zhiliang, Yang Yi, Wang Lu, and Wang Wei. A soa based iot communication middleware. In *Mechatronic science, electric engineering and computer (MEC), 2011 International conference on*, pages 2555–2558. IEEE, 2011.
- [89] Dominique Guinard and Vlad Trifa. Towards the web of things: Web mashups for embedded devices. In *Workshop on Mashups, Enterprise Mashups and Lightweight Composition on the Web (MEM 2009), in proceedings of WWW (International World Wide Web Conferences)*, Madrid, Spain, April 2009.
- [90] Sarfraz Alam, Mohammad MR Chowdhury, and Josef Noll. Senaas: An event-driven sensor virtualization approach for internet of things cloud. In *Networked*

-
- Embedded Systems for Enterprise Applications (NESEA), 2010 IEEE International Conference on*, pages 1–6. IEEE, 2010.
- [91] Teemu Leppänen, Jukka Riekk, Meirong Liu, Erkki Harjula, and Timo Ojala. Mobile agents-based smart objects for the internet of things. In *Internet of Things Based on Smart Objects*, pages 29–48. Springer, 2014.
 - [92] ACL Fipa. Fipa acl message structure specification. *Foundation for Intelligent Physical Agents*, <http://www.fipa.org/specs/fipa00061/SC00061G.html> (30.6. 2004), 2002.
 - [93] Flavio Bonomi, Rodolfo Milito, Jiang Zhu, and Sateesh Addepalli. Fog computing and its role in the internet of things. In *Proceedings of the first edition of the MCC workshop on Mobile cloud computing*, pages 13–16. ACM, 2012.
 - [94] Franco Cicirelli, Giancarlo Fortino, Andrea Giordano, Antonio Guerrieri, Giandomenico Spezzano, and Andrea Vinci. On the design of smart homes: A framework for activity recognition in home environment. *Journal of medical systems*, 40(9):200, 2016.
 - [95] Dirk Slama, Frank Puhmann, Jim Morrish, and Rishi Bhatnagar. Enterprise internet of things, 2015. <http://enterprise-Internet of Things.org/book/enterprise-Internet of Things/>.
 - [96] Tom Collins. A methodology for building the internet of things.
 - [97] Franco Zambonelli. Towards a general software engineering methodology for the internet of things. *arXiv preprint arXiv:1601.05569*, 2016.
 - [98] Nikolaos Spanoudakis and Pavlos Moraitis. Engineering ambient intelligence systems using agent technology. *IEEE Intelligent Systems*, 30(3):60–67, 2015.
 - [99] Bogdan Manate, Florin Fortis, and Philip Moore. Applying the prometheus methodology for an internet of things architecture. In *Proceedings of the 2014 IEEE/ACM 7th International Conference on Utility and Cloud Computing*, pages 435–442. IEEE Computer Society, 2014.
 - [100] OMG UML. Unified modeling language. *Infrastructure Specification, version*, 2(1), 2007.
 - [101] Paolo Bresciani, Anna Perini, Paolo Giorgini, Fausto Giunchiglia, and John Mylopoulos. Tropos: An agent-oriented software development methodology. *Autonomous Agents and Multi-Agent Systems*, 8(3):203–236, 2004.
 - [102] Inmaculada Ayala, Mercedes Amor, and Lidia Fuentes. The sol agent platform: Enabling group communication and interoperability of self-configuring agents in the internet of things. *Journal of Ambient Intelligence and Smart Environments*, 7(2):243–269, 2015.
 - [103] Claudio Savaglio, Giancarlo Fortino, and Mengchu Zhou. Towards interoperable, cognitive and autonomic iot systems: An agent-based approach. In *Internet of Things (WF-IoT), 2016 IEEE 3rd World Forum on*, pages 58–63. IEEE, 2016.
 - [104] Giancarlo Fortino, Antonio Guerrieri, Wilma Russo, and Claudio Savaglio. Towards a development methodology for smart object-oriented iot systems: A metamodel approach. In *Systems, Man, and Cybernetics, 2015 IEEE Intl. Conf. on*, pages 1297–1302. IEEE, 2015.
 - [105] OM Group et al. Software & systems process engineering metamodel specification 2.0.
 - [106] Andreas Varga et al. The omnet++ discrete event simulation system. In *Proceedings of the European simulation multiconference (ESM2001)*, volume 9, page 65, 2001.

References

- [107] Fabio Bellifemine, Agostino Poggi, and Giovanni Rimassa. Jade—a fipa-compliant agent framework. In *Proceedings of PAAM*, volume 99, page 33. London, 1999.
- [108] IoT-A. <http://www.iot-a.eu/public/public-documents>, 2014.
- [109] Giancarlo Fortino, Wilma Russo, and Claudio Savaglio. Agent-oriented modeling and simulation of iot networks. In *2016 Federated Conference on Computer Science and Information Systems (FedCSIS)*, pages 1449–1452. IEEE, Sept 2016.
- [110] Giancarlo Fortino, Claudio Savaglio, and Wilma Russo. Simulation of agent-oriented internet of things systems. In *Proc. 17th Workshop From Objects to Agents*, pages 8–13, 2016.
- [111] A Varga. Inet framework for the omnet++ discrete event simulator, 2012.
- [112] Lars Braubach, Alexander Pokahr, and Winfried Lamersdorf. Jadex: A short overview. In *Main Conference Net. ObjectDays*, volume 2004, pages 195–207, 2004.
- [113] Antonio Moreno, Aïda Valls, and Alexandre Viejo. *Using JADE-LEAP implement agents in mobile devices*. Universitat Rovira i Virgili. Departament d’Enginyeria Informàtica, 2003.
- [114] Doug Crockford. Google tech talks: Javascript: The good parts, 2009.
- [115] Giancarlo Fortino, Antonio Guerrieri, Gregory MP O’Hare, and A Ruzzelli. A flexible building management framework based on wireless sensor and actuator networks. *Journal of Network and Computer Applications*, 35(6):1934–1952, 2012.
- [116] Giancarlo Fortino, Roberta Giannantonio, Raffaele Gravina, Philip Kuryloski, and Roozbeh Jafari. Enabling effective programming and flexible management of efficient body sensor network applications. *IEEE Transactions on Human-Machine Systems*, 43(1):115–133, 2013.
- [117] Zhengguo Sheng, Shusen Yang, Yifan Yu, Athanasios Vasilakos, Julie Mccann, and Kin Leung. A survey on the ietf protocol suite for the internet of things: Standards, challenges, and opportunities. *IEEE Wireless Communications*, 20(6):91–98, 2013.
- [118] Giancarlo Fortino, Wilma Russo, Claudio Savaglio, Mirko Viroli, and Mengchu Zhou. Modeling opportunistic iot services in open iot ecosystems. In *Proc. 18th Workshop From Objects to Agents*, pages 90–95, 2017.
- [119] Jacob Beal, Danilo Pianini, and Mirko Viroli. Aggregate programming for the internet of things. *Computer*, 48(9):22–30, 2015.
- [120] Christos G Cassandras and Stephane Lafortune. *Introduction to discrete event systems*. Springer Science & Business Media, 2009.
- [121] Rodrigo Roman, Jianying Zhou, and Javier Lopez. On the features and challenges of security and privacy in distributed internet of things. *Computer Networks*, 57(10):2266–2279, 2013.
- [122] Ya Shi, Cong Tian, Zhenhua Duan, and Mengchu Zhou. Model checking petri nets with msvl. *Information Sciences*, 363:274–291, 2016.
- [123] Abu Zafar Abbasi, Zubair A Shaikh, et al. Building a smart university using rfid technology. In *Computer Science and Software Engineering, 2008 International Conference on*, volume 5, pages 641–644. IEEE, 2008.
- [124] Hsing-I Wang. Constructing the green campus within the internet of things architecture. *International Journal of Distributed Sensor Networks*, 2014, 2014.

-
- [125] Marian Cață. Smart university, a new concept in the internet of things. In *RoEduNet International Conference-Networking in Education and Research (RoEduNet NER), 2015 14th*, pages 195–197. IEEE, 2015.
- [126] Thanos G Stavropoulos, Ageliki Tsioliariidou, George Koutitas, Dimitris Vrakas, and Ioannis Vlahavas. System architecture for a smart university building. In *International Conference on Artificial Neural Networks*, pages 477–482. Springer, 2010.
- [127] Michele Nati, Alexander Gluhak, Hamidreza Abangar, and William Headley. Smartcampus: A user-centric testbed for internet of things experimentation. In *Wireless Personal Multimedia Communications (WPMC), 2013 16th International Symposium on*, pages 1–6. IEEE, 2013.
- [128] Alessandro Sabato, Maria Q Feng, Yoshio Fukuda, Domenico Luca Carní, and Giancarlo Fortino. A novel wireless accelerometer board for measuring low-frequency and low-amplitude structural vibration. *IEEE Sensors Journal*, 16(9):2942–2949, 2016.
- [129] Tomás Sánchez López, Damith C Ranasinghe, Mark Harrison, and Duncan McFarlane. Adding sense to the internet of things. *Personal and Ubiquitous Computing*, 16(3):291–308, 2012.
- [130] Gobinda G Chowdhury and Sudatta Chowdhury. *Introduction to digital libraries*. Facet publishing, 2003.
- [131] Tefko Saracevic. Digital library evaluation: Toward an evolution of concepts. *Library trends*, 49(2):350, 2000.
- [132] Leonardo Candela, G Athanasopoulos, D Castelli, K El Raheb, P Innocenti, Y Ioannidis, A Katifori, A Nika, G Vullo, and S Ross. The digital library reference model. *DL.org Project Deliverable*, 2011.
- [133] Dieter Uckelmann, Mark Harrison, and Florian Michahelles. An architectural approach towards the future internet of things. In *Architecting the internet of things*, pages 1–24. Springer, 2011.
- [134] Gianluca Aloï, Giuseppe Caliciuri, Giancarlo Fortino, Raffaele Gravina, Pasquale Pace, Wilma Russo, and Claudio Savaglio. Enabling iot interoperability through opportunistic smartphone-based mobile gateways. *Journal of Network and Computer Applications*, 81:74–84, 2017.
- [135] Gianluca Aloï, Giuseppe Caliciuri, Giancarlo Fortino, Raffaele Gravina, Pasquale Pace, Wilma Russo, and Claudio Savaglio. A mobile multi-technology gateway to enable iot interoperability. In *Internet-of-Things Design and Implementation (IoTDI), 2016 IEEE First International Conference on*, pages 259–264. IEEE, 2016.
- [136] Gianluca Aloï, Marco Di Felice, Valeria Loscri, Pasquale Pace, and Giuseppe Ruggeri. Spontaneous smartphone networks as a user-centric solution for the future internet. *IEEE Communications Magazine*, 52(12):26–33, 2014.
- [137] Micro SD ZigBee card. <http://www.spectec.com.tw/sdz-539.html>.
- [138] Waspote Ultra Low power sensors. <http://www.libelium.com/products/waspote/>.
- [139] Shimmer wearable sensors. <http://www.shimmersensing.com/>.
- [140] Francesco Aiello, Fabio Luigi Bellifemine, Giancarlo Fortino, Stefano Galzarano, and Raffaele Gravina. An agent-based signal processing in-node environment for real-time human activity monitoring based on wireless body sensor networks. *Engineering Applications of Artificial Intelligence*, 24(7):1147–1161, 2011.
- [141] Garmin Vivofit. <http://www.garmin.com/itit/esplora/sport-e-fitness/>.

References

- [142] Roberto Casadei, Danilo Pianini, and Mirko Viroli. Simulating large-scale aggregate mass with alchemist and scala. In *Computer Science and Information Systems (FedCSIS), 2016 Federated Conference on*, pages 1495–1504. IEEE, 2016.