An Architecture for Open Smart Environments

Vom Fachbereich Informatik
der Technischen Universität Darmstadt
genehmigte

Dissertation

zur Erlangung des akademischen Grades
eines

Doktor-Ingenieurs (Dr.-Ing.)

von

Dipl.-Inform. Fernando Daniel Lyardet

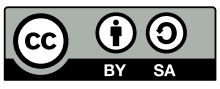
aus La Plata, Buenos Aires, Argentina

Referenten:	Prof. Dr. rer. nat. Max Mühlhäuser
	Prof. Dr. Gabriele Kotsis
	Prof. Dr. Dr. e.h. Lutz Heuser
Tag der Einreichung:	22.10.2021
Tag der Disputation:	16.03.2022

Darmstädter Dissertationen
D17

Fernando Daniel Lyardet: *An Architecture for Open Smart Environments*
Darmstadt, Technische Universität Darmstadt

Jahr der Veröffentlichung der Dissertation auf TUprints: 2023
Tag der mündlichen Prüfung: 16.03.2022



ABSTRACT

Ambient Intelligence's vision builds on the integration provided by Ubiquitous Computing and includes the people as the centrepiece and ultimate purpose technology serve. It is based on the increasing technological advances in embedded computational power, information and sensing capabilities embedded into everyday objects. Many projects in the last decade have pursued the realisation of environments sentient to human presence and proactive to support the user's needs. However, despite the current availability of technology, there is a notorious absence of large-scale settings. This absence raises questions about the complexity and effort required for top-down approaches to make smart environments available everywhere. There are too many scenarios to be modelled one by one in real life, and the scenarios, people, and technology change over time.

Several approaches have been pursued to integrate devices and services into federations interacting with humans as a coherent system. However, they need to address devices' cooperation across different environments and include the people as active participants. This results in isolated islands of integration with clearly defined boundaries, such as the smart home or office and between technology and people.

This thesis presents a novel paradigm called Open Smart Environments (OSEs) that are based on three main entities: knowledge-bundled devices and services called tangible or soft Smart Products, the Human as a first-class actor with abilities, preferences and skills, and the Environment as a representation of the 3D space where both Human and Smart Products are situated and carry out their daily tasks and activities. Besides the matching architectures and development, these three components require new adaptation mechanisms, component architectures and concepts for their realisation that are also proposed, implemented and demonstrated in this thesis. The proposed components provide the mechanisms to build intelligent environments that gradually grow in complexity and smartness. People and technology interact seamlessly with each other in and across space.

ZUSAMMENFASSUNG

Die Vision der Ambient Intelligence baut auf der Integration durch Ubiquitous Computing auf und stellt den Menschen in den Mittelpunkt, dem die Technologie letztlich dient. Sie basiert auf den zunehmenden technologischen Fortschritten bei der Integration von Rechenleistung, Informationen und Sensorik in Alltagsgegenstände. In den letzten zehn Jahren haben viele Projekte die Realisierung von Umgebungen angestrebt, die auf die menschliche Anwesenheit reagieren und proaktiv die Bedürfnisse des Benutzers unterstützen. Trotz der gegenwärtigen Verfügbarkeit von Technologie gibt es jedoch keine groß angelegten Umgebungen. Dieses Fehlen wirft Fragen über die Komplexität und den Aufwand auf, der für Top-Down-Ansätze erforderlich ist, um intelligente Umgebungen überall verfügbar zu machen. Es gibt zu viele Szenarien, als dass man sie eins nach dem anderen im wirklichen Leben modellieren könnte, und die Szenarien, Menschen und Technologien ändern sich im Laufe der Zeit.

Es gibt mehrere Ansätze, um Geräte und Dienste in Verbände zu integrieren, die als kohärentes System mit Menschen interagieren. Sie müssen jedoch die Zusammenarbeit der Geräte in verschiedenen Umgebungen berücksichtigen und die Menschen als aktive Teilnehmer einbeziehen. Dies führt zu isolierten Integrationsinseln mit klar definierten Grenzen, wie z.B. dem intelligenten Zuhause oder dem Büro und zwischen Technologie und Menschen.

In dieser Arbeit wird ein neuartiges Paradigma vorgestellt, das als Open Smart Environments (OSEs) bezeichnet wird und auf drei Hauptkomponenten basiert: wissensbasierte Geräte und Dienste, die als greifbare oder weiche intelligente Produkte bezeichnet werden, der Mensch als erstklassiger Akteur mit Fähigkeiten, Vorlieben und Fertigkeiten und die Umgebung als Darstellung des 3D-Raums, in dem sich sowohl der Mensch als auch die intelligenten Produkte befinden und ihre täglichen Aufgaben und Aktivitäten ausführen. Neben den passenden Architekturen und Entwicklungen erfordern diese drei Komponenten neue Anpassungsmechanismen, Komponentenarchitekturen und Konzepte für ihre Realisierung, die in dieser Arbeit ebenfalls vorgeschlagen, implementiert und demonstriert werden. Die vorgeschlagenen Komponenten bieten die Mechanismen zum Aufbau intelligenter Umgebungen, die allmählich an Komplexität und Intelligenz gewinnen. Menschen und Technologie interagieren nahtlos miteinander im und durch den Raum.

ACKNOWLEDGMENTS

I want to express my gratitude to Prof. Dr. rer. nat. Max Mühlhäuser and Prof. Dr. Dr. e.h. Lutz Heuser for their encouragement and support. I also want to thank my family and friends who always believed in me.

CONTENTS

LIST OF FIGURES

LIST OF TABLES

1 INTRODUCTION

A *smart environment* is a space populated with sensors and computer-augmented objects that communicate and cooperate in providing greater comfort to its inhabitants through better services and automated assistance. Advances in the miniaturisation of sensors and microprocessors have made it possible to "weave" computing power into almost any physical object [221]. Such objects shall be henceforth generically described as *products*. Products add new ways to collect, interpret and react to user actions.

In this vision, these new, augmented products would not have sophisticated controls and extensive user manuals but would interact with users seamlessly, similar to conventional products already used daily. Such an exciting world of unobtrusive assistance or *agency* at the user interface triggered several projects to turn the vision of smart environments into a reality [203]. However, despite the initial success of several different projects, smart environments have yet to become part of everyday life. The absence of such advanced living spaces can be attributed to the technical constraints and limitations of how they are built today.

The predominant top-down, as-a-system approach to designing, developing, and deploying creates islands of integration, such as the "smart kitchen" or the "smart living room", that are hard to extend and interconnect. Furthermore, purpose-oriented smart environments do not address that living spaces are never finished –a location may keep a given configuration for a while. Still, like any human space, it is subject to changes and mutations of its purposes and composition as people shape their places according to their needs [69, 91].

For example, modern cars today provide a prime example of smart environments with their advanced mechanisms and assistive technologies. Standard equipment detects when a driver may need to rest or alert the driver about upcoming dangerous obstacles beyond the range of the human eye. Personalised comfort and driving settings are remembered whenever the driver is back in the car. On the one hand, these features showcase how the combination of embedded sensor

technology with processing capabilities can improve people's safety and augment their driving experience. On the other hand, it demonstrates the limitation of highly integrated and centralised systems. In the car example, its features, possible extensions, and interactions are fixed at design time. To upgrade their automobiles' smart environment capabilities, drivers must buy new cars [155, 196].

For smart environments to become pervasive, a new approach must be developed to be seamlessly and transparently integrated into everyday life. This thesis aims to create a model, an architecture, and the technologies to support the creation of smart environments driven by the products people gradually introduce in their habitats. In this thesis, we define a new term, open smart environments (OSEs), that describes a set of essential components configured for new services and incremental levels of intelligence in the environment.

1.1 MAIN CHALLENGES ADDRESSED IN THIS THESIS

LIGHTWEIGHT SMART ENVIRONMENT INFRASTRUCTURES Provide architecture and supporting technology to facilitate the development of smart environments with minimal infrastructure requirements such as servers and custom software development. Allow smart environments that grow with the products people use to define their living spaces.

SPATIAL AWARENESS Enhance the capabilities of smart environments to augment people's experience in that environment. Develop support for activities that take place across spatially separated points. Smart environments should determine how they can help in at least four ways: *planning* (answering the question "How do I get to XXX?"), *way-finding* (guiding people through their way to the desired destination), *searching* (providing the best or more appropriate route to reach a goal), and *monitoring* (guarding people against accessing restricted areas).

EXTENSIBILITY AND INTEROPERABILITY Provide an approach and mechanisms to accommodate new, unforeseen technologies and applications. New hardware and software components created by independent vendors could be seamlessly integrated without commitment to a specific toolkit or technology implementation.

OPEN INTERFACES Remove implementation-specific barriers. Architectures frequently require a commitment to an implementation (e.g. software library, communication middleware, or operating system) instead of an interface. Although the wide variety of existing hardware and software services can be more easily accommodated, the interaction contracts are hidden. Extensibility is only achieved via a centralised controller, becoming a bottleneck for future growth.

1.2 MAIN CHALLENGES NOT ADDRESSED IN THIS THESIS

USABILITY Design adaptive user interfaces. The computer's move from the virtual to the physical world constitutes a new challenge for computer science. The interface was a visible concrete artefact on the desktop where users must perform operations. Now, computer science must deal with where that previously visible interface might have gone. Usability in ubiquitous computing systems introduces several challenges, such as how affordances of augmented artefacts are affected, how the interaction takes place, and how to provide cues of what is going on to the user (i.e. make certain aspects visible to help people rationalise the actions of the machine system which, incidentally, creates an opposing force to the intended invisibility)

SECURITY AND PRIVACY Avoid compromising information about a smart environment's user base. Smart environments require information about inhabitants and preferences to adapt and provide proactive assistance. However, this information deeply invades the private sphere of a human being and requires stringent technical measures to keep such information beyond the reach of unknown parties.

1.3 CONTRIBUTIONS

This thesis has taken several steps to address the main challenges listed in Section 1.1. Altogether, we make three contributions facing the challenges outlined in the enumeration.

1. **Model for scalable, open smart environments (OSE)** Propose a model for building smart environments incrementally as people buy products and install them in their living and working spaces. This model allows third-party vendors and developers to create products that communicate and integrate with other products and people in the surrounding environment.

2. **Technology support and concepts for OSE implementation** Allow smart environment creation with near-zero infrastructure by transferring traditional, locally- installed support infrastructures and services onto a set of cloud-based services.

3. **Support for multi-user and multi-activity smart environments** Introduce a new architecture to seamlessly support different reasoning algorithms that provide for the expansion of the services offered by a smart environment.

1.4 THESIS OUTLINE

This thesis is organised as follows:

CHAPTER 2 - RELATED WORK Provide an overview of related work in ambient intelligence. The pre-existing body of work is reviewed before motivating the issues addressed in this thesis.

CHAPTER 3 - OPEN SMART ENVIRONMENTS MODEL Present a novel model for building smart environments bottom-up, from the products people gradually add to their living places. This chapter also introduces the ambient properties concept, a taxonomy of an environment's smartness properties, examples, scenarios, component interactions, and topologies to achieve each proposed property.

CHAPTER 4 - BOOTSTRAPPING OPEN SMART ENVIRONMENTS The core components are introduced for building open smart environments. Smart products, the main building block for OSEs, are detailed with two examples. A novel mechanism for browsing and connecting to local objects called Me2it is discussed. Then, ubiquitous explanations for smart products are introduced as a novel architectural approach to providing user assistance. Complex procedures call for support for complex behaviour in smart products in a customised workflow engine. Interoperability between smart products from different vendors is critical for adapting smart environment technology. An extension to Mundo, *Mundo Nubes*, is also outlined with a cloud-based service for semantic disambiguation.

CHAPTER 5 - HUMAN AND SPACE IN OPEN SMART ENVIRONMENTS Create models for people, space, and indoor navigation in OSEs. The model for people allows different systems to share their knowledge about the humans they must support and combine the actions of people and technology to achieve other goals.

CHAPTER 6 - HIGHER AMBIENCE IN OPEN SMART ENVIRONMENTS Explain the new concepts necessary to extend OSEs and support higher context-awareness and adaptation capabilities. First, this chapter explains the concept of *embedded preferences for products* (EP4P) to enable smart interactions between smart products and their surroundings throughout their life cycle. Afterwards, a granular model for assessing and adapting higher abstraction levels of context-awareness called embedded situations is presented. Finally, the concepts of Areas of Influence and places are integrated to support multiple concurrent activities in an environment.

CHAPTER 7 CONCLUSION AND OUTLOOK Summarize the thesis, conclude, and suggest possible future work. Evaluate scientific contributions presented in this thesis instead of summarising all the evaluations in a single, dedicated chapter.

1.5 PUBLICATIONS

Parts of this thesis have been published in different conferences, workshops, book chapters and journals. The first step concerned work on ad-hoc interaction between devices called *iClouds* [100, 101], a project directed by Dr Andreas Heinemann. The need for a different approach to supporting context awareness in applications was first discussed in [179]. The indoor navigation support has been published in [135, 141], different aspects of building smart products and lessons learned in [19, 21, 22, 114, 129], the ubiquitous explanations [139], and the architecture for reasoning modules and different modes of situation assessment [137].

In addition to scientific publications, the author also co-organized several workshops to foster the discussion of scientific issues related to smart products, smart environments, and the Internet of things: AmIBlocks'07 [131], AmIBlocks'08 [132, 134], AmIBlocks'09 [133], and TouchTheWeb'2010 [138].

One patent was awarded in 2018 (EU: **EP3144918A1** / US: **9.947.219**) about IoT camera-based traffic analytics for Smart Cities and two further applications in process on IoT applications and embedded preferences.

Different implementation aspects and prototyping of this thesis have been carried out as part of several master theses at TU-Darmstadt. I had the privilege of serving as a supervising tutor. Special thanks to the students who devoted their energy, programming skills, and motivation [27, 60, 89, 122, 195, 204, 205].

Finally, a considerable body of work was devoted to the design of Voice User Interfaces (VUIs) [190, 192, 193] however, it will not include in this thesis.

2

RELATED WORK

As motivated in Chapter 1, making smart environments grow organically based on the objects and services we deploy is crucial to transforming them into an everyday reality. This chapter provides an overview of fundamental concepts about the origins, definition, current trends and state of the art in Smart Environments. This chapter is organised as follows: first, we provide a rationale for the origins and conceptualisation of Smart Environments (Section 2.1.1). We then present a review and characterisation of the Smart Environments research and application scenarios (Section 2.2.2). We further continue our study by analysing the different design concerns, and we compare the other technology choices to address them. Afterwards, we summarise a set of requirements to be met for Smart Environments to be considered as such, and we conclude this chapter with a short discussion (Section 2.4.9)

2.1 INTRODUCTION

2.1.1 *Ubiquitous Computing*

The vision of Ubiquitous Computing (UC) [77, 220] proposes a world of omnipresent information technology in which computation is seamlessly "woven" into the environment to support people in their everyday activities. Enriching everyday surroundings with information

technology is thereby done in such a way that computers become omnipresent but at the same time also invisible to people. UC calls for computers to disappear into the background to become integral parts of the environment. People might not even be able to identify the abundant information technology surrounding them.

A critical aspect that UC introduces is the focus on people: Computers embedded into the everyday environment are supposed to sense the context of people and adapt their application behaviour automatically based on a user's current surroundings.

2.1.2 *Context Awareness*

In the scope of this dissertation, context is an essential concept because context-awareness provides the means for products to cooperate, perceiving the current state of the environment [65, 66]. The term "context-aware computing" became popular in the middle of the 90s when researchers started to develop applications that incorporated sensed information, such as the GPS location of users. These applications used the location as auxiliary information to improve interaction with the user by adapting it to the user's needs. Context-awareness became a synonym with the property of systems that adapt to sensed information and, therefore, an essential aspect of ubiquitous computing systems. There is a considerable body of work on context-awareness and its applications, but it is beyond the scope of this thesis.

2.1.3 *Smartness*

According to the Merriam-Webster dictionary, the quality of being "smart" is "having intelligence". Moreover, intelligence is described as *"the ability to comprehend; to understand and profit from experience"*[Webster] and *"the ability to adapt to, shape, and select environments to accomplish one's goals within the context of one's society and culture"* [201].

The notion of *smartness* or *intelligence* we adopt in this thesis is the quality attributed to technical artefacts and systems with at least some of the following qualities:

- **Anticipation:** seem to *know things* and as such can free users from having to "spell everything out" to perform a task, requiring little or no attention to control its operation [113].
- **Perception:** with enough knowledge to observe something and take action based on such observation [175].
- **Intelligibility:** able to explain on its own so that users don't have to Figure out what happens when something goes in an unexpected direction [125].

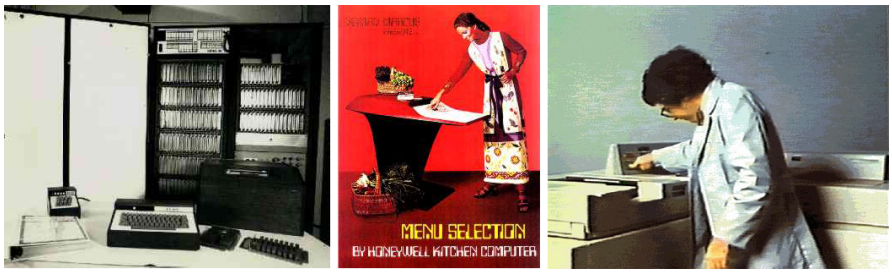


Figure 2.1: Early examples of computerised assistance. From left to right: the ECHO IV Computer (1968), the Honeywell H316 Kitchen Computer (1969), and the Xerox 1790 (1974)

2.1.4 *Smart Environments: Origins and Conceptualization*

The vision of Smart Environments is to take advantage of the embedding of computing and communication capabilities into nearly everything, namely the environment, objects, or even clothing, and achieve integration of such entities to react proactively, adapt and offer new services to the people in their living environments [56, 187]. Much of the current research in Smart Environments is also known under a different term that Phillips Research has raised: *"Ambient Intelligence" (AmI)* [167]. The term AmI captures a similar research vision of a user experience reified out of invisible, connected, and sensing entities: *"people living easily in digital environments in which the electronics are sensitive to people's needs, personalized to their requirements, anticipatory of their behaviour and responsive to their presence"* [167].

Although the concept of *Smart Environments* is relatively new, the concept of technology assistance in people's daily life has been explored for a long time. Some of the first concepts were introduced several decades ago in science fiction series and movies. In those early prototypes, human characters would engage interacting with them as another active character in the story, embodied in the Figure of a single sentient entity (e.g. a main *"computer"*) that can see and listen to what people do everywhere and can be spoken to, gesture to, and interact within other complex ways.

Perhaps the first example linking people's life and well-being to a technological future is Walt Disney's Experimental Community of Tomorrow or EPCOT, an initiative promoted by its founder, was initially set to become *"both a laboratory for future technology and a home for the citizens of tomorrow"*[68, 96].

There have also been some early real-world prototypes developed in the late '60s, some even commercially available. In these early examples, the focus was instead on automation than embedding. The home computer offered to complete the house family finances "automatically", store recipes, compute shopping lists, track family inventory, control home temperature, turn appliances on and off, and predict the weather [199]. Figure 2.1 shows a few examples of those

early systems: first, the Electronic Computing Home Operator (ECHO) IV [97, 199]. Among its many features, the ECHO IV had a message centre intended for family members to leave messages for each other. Second, the Honeywell's 4Kb RAM H316 "Kitchen Computer" from 1969 was the first of such systems to be commercially available. The H316 could store recipes and be able to provide cooking tips [97]. Finally, the first (known) product with computerised assistance was presented in 1974: the Xerox 1790 copier machine [202].

New technological capabilities have triggered many developments related to Smart Environments [69]. However, more than technology is needed for a new field to emerge and establish. Other factors, such as changes in public policies and human factors, shaped a need for such developments. In the first case, a good example is a move towards decentralised health care, and the resulting development of new technologies for autonomous health assistance [56]. In the second case, among the human factors involved in the perceived value of everyday chores [203]: *"traditionally there has been a negative value placed on domestic and other forms of manual labour necessary to the maintenance of infrastructures, such that those with the means to do so delegate that work to others - traditionally servants and other workers, and more recently machines. We can see at least some of the visions that motivate Smart Environments as part of this tradition"*.

2.1.5 *Challenges*

The first generation of smart environments relied on carefully crafted top-down scenarios augmenting existing objects and physical premises with sensors and computing power, and a controller software to interpret those inputs and generate responses to the foreseen scenarios. Arguably a rather monolithic approach that reflected the existing limitations at the time.

In the meantime, the Internet of Things (IoT) [31] has become ubiquitous, augmenting physical things objects with computing power and connectivity. Today, many connected devices and platforms exist in a broad spectrum of applications such as automotive, healthcare, parking, agriculture, utilities, urban lighting, transportation, industrial control, and buildings[2, 3]. Some reports count more than 300 IoT platforms already created and more to come[212]. Not surprisingly, leading vendors like Amazon (AWS IoT), Cisco (Jasper) [4], IBM (Watson) [5], Apple (HomeKit) [6], Google (Brillo, Weave, AndroidThings) [7–9], Microsoft (Azure IoT) [10], and Qualcomm (AllJoyn) [11] have entered the market with varying degrees of success and some have resorted to restricted availability[25] or even disappeared already (Brillo, Weave)[12].

Each new platform introduces its own IoT infrastructure, protocols, and interfaces, with incompatible standards, formats, and seman-

tics which creates closed ecosystems [106, 157]. This technological fragmentation makes systems less interoperable, that is, capable of performing a given task or exchanging information in real-time without a third party[51] and using the information exchanged. Instead, systems become heavily dependent on custom integration when connecting applications so that the other can access data from one system. However, integration involves a third party[176], a mediator or middleware that translates the receiving system's data and APIs. Data and actions do not happen directly between the entities involved in this scenario.

2.1.6 *Composing IoT Systems, Interoperability Challenges and Technology State of the Art*

Integrating systems and devices within the same stack by a single vendor is relatively straightforward but limited to the domains supported by that vendor. As sensing technologies and hardware devices become commoditised, independent providers seek value creation through vertical integration, which often results in the processing of the data generated by their devices being restricted to their platforms and clouds, with little or no compatibility with products from other brands. For instance, a smartwatch developed in Apple WatchOS cannot interact with an environmental sensor without the sensor vendor's proprietary application. This vertical integration leads to the formation of "Intranets-of-Things," [92] which become islands of integration that are difficult to integrate using generic methods.

To tackle the interoperability challenge [23, 33, 34, 61, 95, 164, 197], there is a significant amount of research on different approaches, which identify a standard set of three levels for horizontal interoperability: technological, syntactic, and semantic [92, 118].

Technological interoperability refers to the physical connection of both components and their ability to transmit information on the lower layers of the OSI model and the corresponding protocols. Devices supporting multimode radio equipment represent the most common technical approach towards integrating heterogeneous devices that utilise different networking and communication means. However, according to [106] , technological interoperability still represents a significant barrier in IoT, with up to 60% of the overall potential value locked due to the need for compatible solutions.

Smartphones are a good example: they deploy a *Cellular Radio Interface*for connecting to 2G, 3G, 4G and 5G networks. A *Wi-Fi Radio Interface* that allows the phone to connect to Wi-Fi networks, enabling users to access the internet and make calls and send texts over a Wi-Fi network. A*Bluetooth Radio Interface:* to connect wirelessly to other devices, such as headphones, speakers, and smartwatches; a *NFC Radio Interface* to communicate with other devices, such as contactless pay-

ment terminals, smart cards, and other phones.; a *GPS Radio Interface* to receive signals from GPS satellites, allowing the phone to determine its location and provide location-based services; and some also include a *FM Radio Interface* to listen to FM radio stations without using data or the internet. A smartphone can operate across different networks and communicate with different devices and other phones thanks to the multiple interfaces.

Deploying multiple wireless technologies in a device can potentially expose more attack points where malicious entities could inject unauthorised code and sniff network traffic. The research community addresses different concerns, such as hardware protection mechanisms, cryptographic protocols, secure boot and trusted environment execution to safeguard the system(e.g. [126], [93]).

Syntactic interoperability refers to the agreement between components on the data format, encodings, and API. There are several standard technologies and platforms, including messaging protocols such as CoAP, XMPP, AMQP, MQTT, DDS, Hy-LP, and DPWS and UPnP, and OSGi [94]. However, these solutions offer only inter-domain compatibility, and they usually act as closed silos with a narrow application focus, imposing specific data formats and interfaces. Mechanisms for achieving horizontal interoperability include gateway proxies for messaging protocols [24], which convert messages automatically from one messaging protocol to the compatible format of another protocol. For instance, among RESTful HTTP, CoAP, XMPP, MQTT, and DDS [94].

The approach goes further than translating or converting between message formats since the protocols mentioned above have different operation modes: CoAP operates similarly with HTTP. XMPP requires a resource server, while MQTT imposes a central broker that administrates the communication. The messages from the three distinct settings are parsed through a message broker that implements the multiprotocol proxy's core functionality, translating the messages from one protocol to another and managing the communication flow. The messages can also be maintained locally with a topic router, easing the discovery process of the communicating system components.

Semantic interoperability refers to the agreement between components on the same information model, ontologies, and provided communication interfaces. The W3C defines semantic interoperability as enabling different agents, services, and applications to exchange information, data, and knowledge in a meaningful way, on and off the Web [216]. The Web of Things (WoT) [210] is an important initiative that leverages semantic web technologies to expose data and systems through APIs to help solve technological fragmentation [117]. The Thing Description (TD) [215] is a notable feature of the WoT approach, describing the metadata and interfaces of (physical) Things in a machine-interpretable format. TD has been built upon W3C's extensive work on RDF [170], JSON-LD [107], and Linked Data [214]

and defines a domain-agnostic vocabulary to describe any Thing in its properties, events, and actions. Other semantic schemas can further enrich the properties description, such as the community-driven iot.schema.org.

In summary, achieving interoperability in IoT involves addressing technological, syntactic, and semantic challenges. The research community has proposed various approaches to tackle these challenges, such as deploying multimode radio equipment, using gateway proxies for messaging protocols, and leveraging semantic web technologies like the WoT. These solutions provide a foundation for achieving interoperability and unlocking the potential value of IoT.

2.1.7 *Automatic Composition Through Planning and Scheduling*

Planning is an essential component of rational behaviour, *"the reasoning side of action"* [85]: *"It is both the organisational process of creating and maintaining a plan; and the psychological process of thinking about the activities required to create a desired future on some scale. As such, it is a fundamental property of intelligent behaviour"* and a critical feature towards the *smartness* we need to embed into products. This reasoning process is essential to creating and refining a plan or integrating it with other plans; that is, it combines forecasting of developments with preparing scenarios of how to react to them. The following definitions of planning and plan are derived and adapted from Russell& Norvig [180], and Ghallab et al. [85].

Definition: *"**Planning** is a reasoning process of a problem-solving nature that selects and organises the available set of actions by anticipating their expected outcomes to identify a solution over an abstract set of possible plans. Such a solution is called a Plan"*.

Definition: *"A **Plan** is a specification of what action sequences should be performed by the available services and devices to achieve the desired goal, within given constraints"*.

This thesis focuses on scenarios for environments that may change dynamically, where the calculated strategy often needs to be revised online, as well as the knowledge of the environments and relevant policies.

A typical planner takes three inputs: a description of the *initial state* of the world, a description of the *desired goal*, and a set of *possible actions*, all encoded in a formal language such as STRIPS (Stanford Research Institute Problem Solver) [78]. The planner produces a sequence of steps that lead from the initial state to a state meeting the goal. An alternative language for describing planning problems is that of hierarchical task networks. A set of tasks is given, and each task

can be either realised by a primitive action or decomposed into a set of other tasks.

In STRIPS, planning the problem can be defined as a five-tuple $\langle S, S_0, G, A, \Gamma \rangle$, where S is the set of all possible states in the world, $S_0 \subset S$ is the *initial state* of the world, the set of *possible actions* the planner can perform in attempting to change one state to another state in the world. The translation relation $\Gamma \subseteq \times S \times A \times S$ defines the precondition and effects for the execution of each action.

Finally the *desired goal* is a formal description of the final state the planner aims to achieve and is specified as predicates that must be true in the final state. For example, in a planning problem for an smart home IoT system, the goal could be to achieve a certain state of the environment or devices such as *Goal: {lights-off, temperature-comfortable}*. Here, *"lights-off"* and *"temperature-comfortable"* are predicates that describe the desired final state of the environment. The planner will generate a sequence of actions to achieve this goal, such as turning off the lights and adjusting the thermostat.

2.1.8 *Ai Planning in Service Composition*

Automated planning techniques can allow ambient designers to decouple the intentional (what can I do?) from the operational (how do I do it?). For instance, people at home would be able to express their intention in a very coarse manner "lower the music", or "I want to see Netflix", and the system would be aware of the home's context (with all its devices and services available) can react executing a set of actions to achieve the desired request.

Different techniques and solutions have been devised over the years to assist people in achieving their requests by instrumenting their living space and making it smart, taking advantage of the flexibilities an ai-planner can provide [111, 124, 147, 148, 158, 162, 168, 208]. As early as 2003[99], AI planners were proposed to automate the operation and configuration of multimedia systems using an ai-solver to search for a target goal. A universe of available components would be semantically described in the proposed method. The ai-solver would take a universe description and search for a solution that achieves the user-defined target. When changes were introduced to the environment, the universe of known semantic descriptions would be updated accordingly, thus allowing the system to adapt to the changing situation in its environment.

The ability to adapt becomes more relevant to people's environments without expecting technical expertise and sensitivity to configuration changes. The automation can remain functional after changes in the background without expert intervention. For instance, Chen et al. [44] applied a type of ai-planner called Hierarchical Task Network

Planner (HTN) and introduced the novelty of user preferences as a solver constraint to find the target solution.

2.1.9 *AI Planning for IoT Composition*

The service composition of IoT devices has some differences introduced by the physical nature of the participants, such as the time to complete a request. While software services may react almost instantaneously, physical objects may take an arbitrary time to complete an action. This also means that, in the ai-planning world, the desired "effect" on the universe also takes time. Several approaches have been created to fit this temporal property of physical actions better [15, 87, 208, 226] while introducing elements such as change monitoring and auto-replanning to counter unexpected changes in the environment to avoid completely invalidating the current execution.

Error support and recovery are among the weakest aspects of the approaches mentioned above by merely communicating to the user that an error has occurred. Novel approaches aim to tackle this weakness [87] by aiming to solve conflicts of configuration, unexpected events, and device malfunctions to resume and complete the intended task. Another promising research area is the application of a weighted CSP planner [208] to allow partial goal fulfilment in a bid for resolving conflicts in a multi-user smart environment setting.

2.2 SMART ENVIRONMENTS

In this section, we will evaluate several Smart Environment research projects.

2.2.1 *Properties of Open Smart Environments*

USER ASSISTANCE Ideally, when people encounter a new product, they can understand its purpose and how the product designer intended its use[203]. However, the increasing amount of technology in a given product today enables exciting new functionality and turns technology-enriched artefacts increasingly opaque to human understanding. Therefore, assisting people in operating a product, recognising its constituent parts, and understanding the functionality basics and purpose without extensive training is a key feature of Smart Environments.

COMPLEX TASK SUPPORT when people encounter technology, it requires a cognitive effort to recognise a given product's essential operation and purpose. In other situations, people are also under mental pressure, particularly with multi-step procedures and activities that cannot be simplified to the touch of a simple but-

ton. For instance, complex setup and maintenance procedures involve activities involving multiple products where people are also active participants. These situations are more complex and may extend over long periods, making it particularly difficult for people and an important area where Smart Environments can offer support.

CONNECTED PRODUCTS We refer to smart environments that build their assistance by taking advantage of products augmented with sensors and communications. Products that can communicate about themselves and with each other.

COOPERATING PRODUCTS Products that can recognise each other collaborate to complete an ongoing activity and organise themselves to achieve a user goal or complete an ongoing operation.

OPEN EXTENSIBILITY Abstract knowledge representations and reasoning techniques are applied to achieve higher levels of adaptation and anticipation of its inhabitant needs.

PEOPLE-AWARE We focus on modelling characteristics, skills and activities of smart environment inhabitants.

SITUATION-AWARE Learn behavioural patterns, preferences, and anticipation of need. Through sensed information and the recording and analysis of user events, smart environments can "learn" preferred user settings and assess situations.

SPACE-AWARE Interactions between people and things take place across space. Here is the indoor model and CoINS.

2.2.1.1 *Ambients of Intelligent Things*

The concept of Ambients of Intelligent Thingsunderstands Smart Environments *"as networked devices communicating with each other and following strategies in order to react intelligently to the user's situation, interaction and goals (context and situation awareness)"*[35] . To achieve this, it is necessary to have distributed software infrastructures, which enable the self-organisation of devices and their software components.

A different view on building Ambient intelligent infrastructures should be able to configure themselves and grow from the available, purposeful objects (be it software services or consumer appliances) to become effective in the real world. We refer to these purposeful objects as *"Smart Products"*, which are real-world objects, devices or software services bundled with knowledge about themselves and their capabilities. These properties make Smart Products intelligible to users and smart to interpret users' actions and adapt accordingly. By naming these objects Smart Products, we convey the notion of technology available "off-the-shelves" and the notion of software services and hardware

Criteria	Evaluation
User Assistance	● User can receive assistance anywhere ◐ User can receive assistance if special infrastructure is available ○ No support.
Complex Task Support	● Assistance available involving one or more products ◐ Assistance available on single products/task. ○ No support.
Connected Products	● Products available are augmented with communication and procesing ◐ Products available are RFID enabled ○ No support.
Open Extensibility	● Independent vendors can integrate their software and hardware products ◐ Integration possible but requires specific software libraries ○ No support.
People Aware	● People are modeled with their capabilties and skills ◐ Some user information profile is kept by the system ○ No supports.
Situation Aware	● Supports Situational Context Assessment ◐ Support different Context States ○ No support.
Space-Aware	● Supports planning, wayfinding, searching and monitoring ◐ Sopports world model, location information ○ No support.

Table 2.1: Summary of the Evaluation Criteria for Smart Environments Projects

objects required to assemble new, innovative end-user components. Therefore, Smart Products share some fundamental properties: having multiple uses, being deployed independently, and networking with other objects to augment their individual and collective capabilities.

The notorious absence of large scale smart environments is strongly influenced by the complexity and effort required to build them. Existing examples such as the office [14, 58] or home [39, 115, 207] have been carefully designed top-down by hand at drawing boards, but this approach is fairly limited.

Therefore, future ambient intelligent infrastructures must be able to configure themselves from the available, purposeful objects [74] creating small worlds, *"where all kinds of smart devices are continuously working to make inhabitants' lives more comfortable"* [57].

However, despite the current availability of technology, there are two notorious trends: the scenarios are usually single-user oriented, and the absence of large scale settings. These two characteristics raise questions about the complexity and effort required with current approaches to building such intelligent living and working spaces.

A few projects such as The Adaptive House [154], the Aware Home [115], the Changing Places /House-n [105] , the MIT's Kitchensense [123] and the MavHome [227] represent some examples of a vision

	Communicating Products	Complex Task Support	User Assistance	Open Extensibility	People Aware	Situation Aware	Space-Aware
Amb. Agoras	●	○	◐	◐	◐	◐	◑
AMIGO [1]	●	◐	◐	◐	◐	◐	◐
Aura [83]	●	○	◐	○	◐	◐	◐
Aware Home [115]	◐	◐	◐	◐	◐	◐	◐
DynAMITE [168]	●	◐	◐	◐	◐	◐	◐
EasyLiving [39]	○	○	◐	○	◐	◐	○
Gaia [177]	◐	○	◐	◐	◐	◐	◐
Intel. Classroom [80]	◐	◐	◐	◐	◐	◐	◐
Intel. Workspaces [109]	◐	○	◐	○	◐	◐	○
MavHome [227]	◐	◐	◐	◐	◐	◐	◐
Nexus [227]	●	◐	◐	◐	◐	◐	◐
One.World [88]	◐	◐	◐	◐	◐	◐	◐
Oxygen [144]	◐	◐	◐	◐	◐	◐	◐
SmartKom [217]	◐	◐	◐	◐	◐	◐	◐
Open Smart Environments	●	●	●	●	●	◐	●

Table 2.2: Evaluation of related Smart Environments projects

where the environment acts as an agent, able to infer the activity of their occupants and anticipate their intentions to better adapt to their needs. Around this conception of the Smart Environments as an application, several middleware technologies (Gaia [177], Aura [83], one.world [88], Nexus [219] and iWork [109]) and programming languages [62, 72] have been developed to support the construction of such environments, and have been applied on the instrumentation of consumer-oriented locations, e.g., office [14, 58], home [39, 115, 207], bathrooms [45].

Well known projects that built up smart environments are the *EMBASSI* project [168] that realises smart environment within a living room scenario, within automobiles and terminal systems, the *MAP* project [143] for office scenarios, and the *Smartkom* project [217] for mobile and consumer scenarios. Other well-established examples are the *Easy Living* project from Microsoft [173], the *Interactive Workspaces* project [109, 110] from Stanford University, the *Intelligent Classroom* [80] from Northwestern University, or the *Oxygen* project [144] of the Massachusetts Institute of Technology. Those smart environments are examples of handcrafted integrations of devices and components

whose functionality is known to the developers, and the data flow from device to device is manually determined for every use case [102].

3

A MODEL FOR OPEN SMART ENVIRONMENTS

"A House is a machine for living in."
Le Corbusier

Smart Environments offer a vision of unobtrusive interaction, with our surroundings interpreting and anticipating our needs and offering a greater degree of control to its inhabitants. While big advances have been achieved in interaction mechanisms and interface metaphors, building and extending such environments has been difficult. The complexity and effort required to build such environments raise how to make them scalable and sustainable. Current approaches are too work-intensive and scenario dependent, and, in the real world, there are too many scenarios to be modelled one by one. Furthermore, purpose-oriented Smart Environments do not address the fact that living spaces are never "complete": a location may keep a given configuration for a while, but like any human space, it is subject to changes and mutations of its purposes and composition, as people shape their places according to their needs.

3.1 PUBLICATION AND CONTRIBUTION STATEMENT

3.1.0.1 *Publication*

This chapter is partially based on the following publications:

Fernando Lyardet, Aristotelis Hadjakos, and Diego Wong Szeto. "InSitu: An Approach for Dynamic Context Labeling Based on Product Usage and Sound Analysis." In: *Workshop Proceedings of the AAAI-11 Conference*. 2011

3.1.0.2 *Contribution Statement*

I led the concept generation and overall architecture. Prof. Dr Max Mühlhäuser has supported me during the idea generation and conceptual design and evaluation of all the prototypes and concepts developed during this thesis.

3.2 INTRODUCTION

This chapter makes two primary contributions. Firstly, it presents a model that allows an incremental development of smart environments based on the objects placed within them and the network services available. Secondly, it introduces the concept of Smart Environment properties to describe the level of *ambience* or functionality that an environment may possess. The Smart Environment properties suggest various scenarios, component interactions, and topologies to achieve each identified property.

The chapter is structured as follows: in section 3.3 we first describe the basic requirements for a flexible Smart Environment. Afterwards, in section 3.4, we introduce the basic concepts of the Open Smart Environments model, its components and interactions. In section 3.6 we explain the ambience properties defined in the OSE model, and we show through different scenarios of varying complexity how the different degrees of ambience are achieved. Finally, section 3.8 discusses the scientific contributions of the chapter.

3.3 REQUIREMENTS FOR A NEW SMART ENVIRONMENT MODEL

Following the analysis of previous approaches for building smart environments presented in the previous chapter, several key aspects have been identified as requirements for the OSE model:

ENSURE INDEPENDENT, SELF-SUSTAINABLE COMPONENTS: Every component should be able to work independently of the availability of other components to fulfil their purpose as components. They may be extended or achieve new functions in the presence of other products, but such presence must not be required to provide its basic functionality.

ALLOW DYNAMIC EXTENSION: the enriched environments should not require a "shutdown" to introduce new functionality.

ALLOW DIFFERENT DEGREES OF EMBEDDING FOR DIFFERENT PRODUCTS: To achieve true open environments, different manufacturers will produce many different kinds of products. This means that not every product will be able to justify a high degree of technology embedding economically. Therefore, it should be possible to deploy different capabilities according to each component.

ALLOW BUILDING SMART ENVIRONMENTS WITHOUT DEDICATED INFRASTRUCTURE: Although custom-built smart environments can achieve high degrees of integration, it is an approach that does not scale well for existing infrastructures. Therefore, the OSE architecture should achieve higher ambience qualities that do not require extensive infrastructure modifications.

3.4 THE OPEN SMART ENVIRONMENT (OSE) MODEL

This section presents a new model for building Smart Environments that gradually grow in complexity and smartness, as people define them with the objects they place. This new model, called Open Smart Environments (OSE), proposes different levels of ambience that can be achieved by gradually adding objects (called "Smart Products" in this model) to any given space.

We call this model ***Open***, because it introduces the *Space* as a first-class entity, enabling the integration of what until now had been integration islands: isolated smart environments such as *"the smart kitchen"* or the *"the smart office"* whose functionality and operation would include only entities within their boundaries. A second reason for calling this model ***open*** is that all artefacts in the environment are defined in a technology-independent way, allowing for third party manufacturers to add their products to the environment.

3.4.1 *Building Blocks of Open Smart Environments*

The OSE model defines a small set of components as the main building blocks for smart environments and ambience properties that help characterise the degree of assistive support available in a particular place. Furthermore, the OSE model proposes a reference architecture and some associated support technologies. The OSE model defines the following four components as the main building blocks for smart environments:

- ***People***
- ***Smart Products***
- ***Space***
- ***Active Knowledge***.

3.4.1.1 *People*

People represent the humans in the environment, with their *preferences*, *physical capabilities* and *personal skills*.

Preferences: describe the desire or predisposition of a person in favour of something. Such preferences have been modelled using a Multi-attribute Utility Theory (MAUT)-based model. Together with the physical capabilities information, they are used to adapt the assistance and guidance humans receive in a smart environment.

Physical Capabilities: have been modelled following the standard proposed by the World Health Organization (WHO): the International Classification of Functioning, Disability and Health (ICF) [161], that describes health and physical-condition-related states.

Personal Skills: model what a person can do, and it is expressed in the same procedural terms (*Actions, Requirements and Post-conditions*) as the functions available for other entities in the environment. Choosing such representation facilitates composing people and things and achieves a common goal together. Furthermore, *skills* also allows representing more specific and advanced tasks a person can so that may be required in a particular situation or task.

3.4.1.2 *Smart Products*

Smart Products are connected objects we encounter in our everyday lives, equipped with sensing, computation, and communication capabilities, that can perceive and interact with their environment and with other smart objects. Furthermore, they are augmented with embedded knowledge about themselves, their components, and how they are meant to be used.

The use of the term *"product"* denotes three important aspects: first, the fact that they keep their original use and aspect. At the same time, computing supports a new quality of interaction and behaviour. Second, a broader view includes any connected devices or services available on the network. And third, their nature as goods being produced from different vendors, distributed, sold, installed, used and eventually discarded. In this thesis, we also extend the information augmentation to software, and we will consider such soft-smart products henceforth as "smart" services to differentiate them from real-world objects. Smart Products feature 4 key parts: *Product Knowledge*, *Process Control*, *Lifecycle Knowledge & Preferences*, and *User Model*.

Product Knowledge: it describes the functionality in the physical world as an appliance, its constituent parts, and the functionality it offers. The smart product architecture and description presented in this thesis further develops this concept and introduce the notion of embedded Lifecycle Preferences for Products (LP4P). LP4P describes a product's physical and operational

characteristics, and they can be used to assess the product's usage and handling in a given physical context and lifecycle stage. Furthermore, through LP4Ps, it can also be established whether or not a given product could be used in a given environment, conform to safety norms, and still perform as intended by the manufacturer.

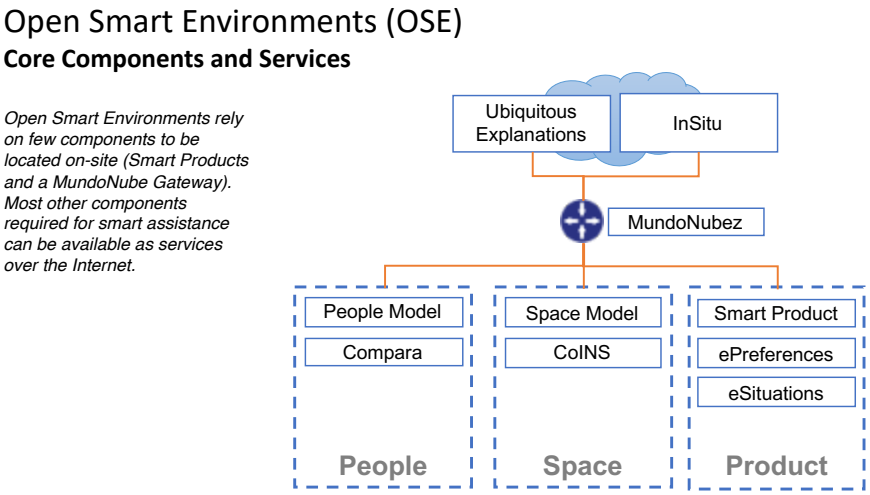


Figure 3.1: Open Smart Environments architecture.

Process Control: A Smart Product requires the ability to carry out complex, multi-step actions involving the product itself, people and other products and services as well. The current state of the art in adapting workflows only allow for a limited degree of parametric or sensed adaptations. In this thesis, we present a new kind of workflow engine that, together with plan-based reasoning, allow for adapting how a particular task can be carried out, taking advantage of the current context.

Lifecycle Knowledge & Preferences: The term *preference* is usually associated with people. The use of this term in the context of Smart Products signifies the reflective character of a product with embedded information about itself and its use, that can now inform people and other products: *"How should I (the product itself) be installed / used / transported / stored and recycled?"*

User Model: it describes what do *the product* know about its users: their preferences, usual settings, experience using *it*.

3.4.1.3 *Space*

Represents the 3D environment where products and people are situated. It models their location and can provide the necessary information to support situated activities through orientation and indoor navigation. Under the OSE model, the "Smart Environment" does

not have an explicit representation. Instead, OSEs are structured into *locations*, and each location can contain many *places*: areas within those locations that are determined by the object groupings created by people and the activities people do. These groupings can be automatically generated using the "Areas of Influence" described in chapter 6.

3.4.1.4 *Active Knowledge:*

Represents the different knowledge representation and inference mechanisms extend what an open smart environment knows. Knowledge modules are used to develop or introduce new features to products, reason, analyse and recognise the state of an environment, and decide what actions should be taken to adapt the environment to the current situation.

3.4.2 *Lifecycle Knowledge in OSE*

In OSE, smart functionality arises from the interactions between products, people, and space within open smart environments. These interactions occur within a context that encompasses the unique circumstances of each participant, such as their user's current activity or intent, but also the availability and operational status of the various products and services within the environment. Our research proposes that also the product lifecycle stage is valuable context information.

There are different standards for describing a product lifecycle process. One such standard is the OOBE (Out Of the Box Experience), which occurs when a user first interacts with a new product, typically when they first open the product's packaging. It refers to the user's initial experience with a product before they begin to use it. The OOBE process is designed to provide users with an intuitive and straightforward introduction to the product and its features. In the context of smart functionality and OSE, the OOBE process can bootstrap new opportunities for smart assistance and help create more engaging smart features.

The operational context of a product is primarily defined by its usage. Still, interactions can also occur in different contexts depending on the product's current lifecycle stage, such as transportation, buying, set up, configuration, repairing, replenishment, or disposal. Since the same components can behave differently depending on their lifecycle stage, understanding this additional context information can significantly impact the overall experience. For instance, the smart services that interact in a smart home can differ depending on the current product lifecycle stage, influencing the open smart environment's ability to provide specific smart services.

By understanding the unique contexts in which products and users interact, designers and developers can create more engaging and effective products that meet users' needs throughout the entire prod-

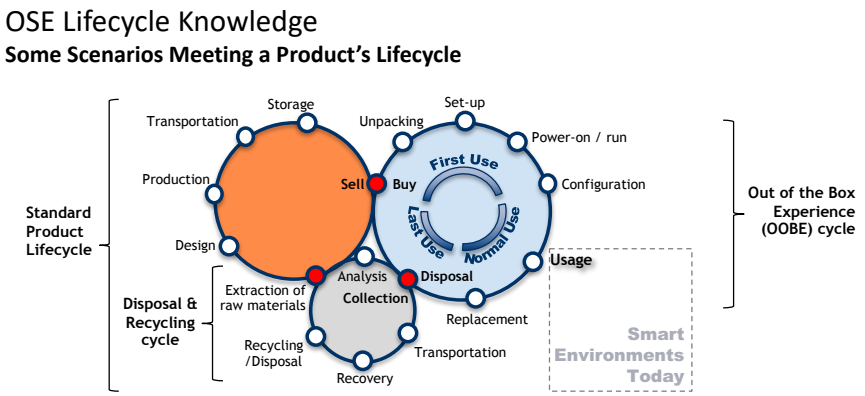


Figure 3.2: Overview of a standard product's lifecycle, and the current focus of Smart Environment research

uct lifecycle. In summary, our research highlights the importance of considering the operational context of products within open smart environments and the potential of the OOBE lifecycle phases to enable new opportunities for smart assistance. For instance, transporting products safely, buying the right appliances for the home, installing them correctly, using and combining them efficiently, and disposing of them appropriately.

Finally, other services can define different lifecycles with contact points with OOBE. For instance, in Figure 3.2, we show an example of the OOBE lifecycle and its contact points to another lifecycle process: a more general product lifecycle and more advanced disposal and recycling.

In Figure 3.3 we show a typical product's lifecycle, with further details in two of the stages: out of the box experience (OOBE) and product disposal. By mapping most of the current developments in Smart Environments presented in the related work chapter (Chapter 2), we observe the strong focus on the *use–interact–operate* stage, but poor or no support at all in most of the other stages.

In the following Figure, we propose different examples of smartness brought to different stages of a smart product.

3.5 SCENARIOS OF OSES

The following scenarios illustrate how smart environments can grow by adding smart products and gradually improving functionality and sophistication. Since the ultimate goal is to enable Smart Environments to become truly pervasive, the scenarios cannot be limited to typical already-setup situations. Instead, smart assistance or "ambience" will be present at different stages in many other situations under the OSE model, even at construction time. To illustrate this vision of a more integral approach, we present different situations involving products at different lifecycle stages and how, at each step, possibilities arise to

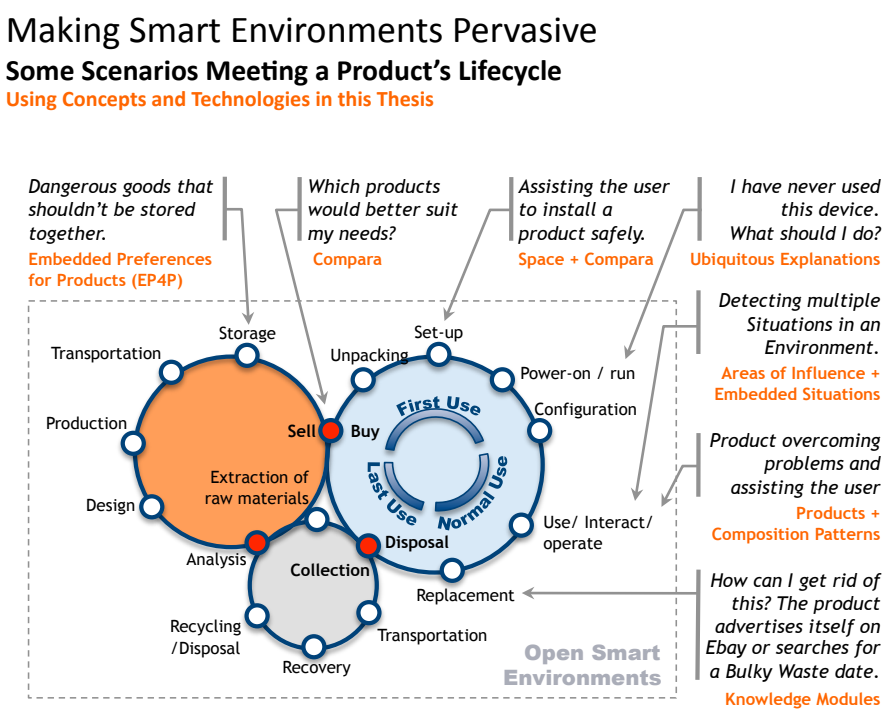


Figure 3.3: Open Smart Environments opens a wider range of scenarios for making our environments smarter

both provide smart assistance and retrofit the environment with more intelligence.

Using the different product lifecycle stages allows us to show very different scenarios where ambience can play a positive role.

We present two such scenarios, the first one a smart home scenario, and the second about the smart workplace. The smart home scenario has been extensively investigated, and several projects described in Chapter 2 have shown different visions of an everyday life augmented with technology. However, the home scenario presented here differs from previous ones. It is subdivided into more minor situations covering the initial moment a person buys a new product for the home, its installation and setup, support for its use and finally to its disposal. The intended sequence following a typical end-user OOBE (Out Of the Box Experience) product experience helps illustrate how, at each step, smart services and products can intervene and provide smart assistance.

3.5.0.1 *A Smart Home Scenario:*

John and Susan went shopping for a few things they needed at home. Susan has decided to buy an electric heater for the bathroom and that new coffee machine she saw on TV. John, instead, is looking for a lawnmower. The old machine is broken, and he will soon need a replacement with the spring about to start.

At the shop, she discovers some new attractive heater models, and she is wondering whether that new model would be the right one for her. Susan collects information about an excellent model with her mobile by scanning a

2D QRCode. The data is sent to her trusted *Smart Home Services* provider, and a list of possible locations for that heater appears on her screen. She wanted the heater for her bathroom, but it does not appear in the list of possible places at home.

-*Why not in the bathroom?* She asks. The *Ubiquitous Explanations* service answers to her screen:

```
That heater is not recommended for very humid spaces.
            How about these other models?
```

Furthermore, after tapping on the screen, a set of alternative heater models in that shop is shown. Susan selects one of them, and the Ubiquitous Explanations service tells her: *"let me guide you there"*, and the *CoINS Indoor Navigation* service guides her to where the selected model is.

In the meantime, John searches for the lawnmower he needs. Like Susan, he finds a model he likes and collects the mower's data with his mobile phone. However, that lawnmower is too small and slow for the size of John's garden. In addition to that, where John lives, a harder kind of grass usually grows that would wear the machine faster than it should. An alternative list of devices is presented on John's screen, and he selects one that fits his budget.

Susan is pleased with the heater and that coffee machine she bought back at home. While unpacking the coffee machine, she wonders where she could locate it at home. Susan taps on the kitchen's screen, displaying the home's floor plan with the possible locations for the coffee machine. She decides to stay with the traditional kitchen, but the only recommended spot is away from the window. Susan clears the place to make room for the machine. The display shows a warning message:

```
The machine is too close to the wall. Please move it forward.
```

Susan moves the machine forward a few centimetres a couple of times until the display shows a happy face. Now she can finally prepare a coffee. She plugs in the device, and after filling up, the beans holder puts her cup under the machine and automatically serves the right amount of coffee for her cup.

Some weeks later, the coffee machine tells Susan that a descaling process would soon be necessary to keep the machine in good condition. She agrees to start it right away, and together with the Ubiquitous Explanation Service, the device guides Susan through all the necessary steps to clean up the machine. The first step involves using a descaling solution, but Susan cannot remember where she has stored it. The coffee machine tells Susan that there is one on the top cupboard of the kitchen. She looked again, and back on the closet, she found the small bottle. Susan can now start the descaling process. Since some of the steps require some time to complete, Susan leaves the machine running and prepares to go out biking as she does every day.

After some 15 min riding on her bike, Susan has got a message from her Virtual Home Services, asking if she would like to reduce her energy consumption at home, as she usually will not be back for another hour. She taps on her mobile phone screen and moves s simple slider to reduce the energy consumption at home. As she moves the slider, a small warning appears: if she reduces it further, the consumables in the fridge will be lost. She takes the advice and stops.

Before the end of her usual round, Susan checks her mobile. In her mind, she wonders about those location-aware services.

"They're pretty cool, but I wonder if it's working." She thinks.

Susan pulls the mobile out of her pocket, and the screen shows that, in anticipation of her arrival, the air conditioning is already working, but the TV and radio are not.

"Interesting. Well, it makes sense after all. I am not there." Susan continues riding very relaxed, with a secret expectation she will confirm when she is back at home: finding the TV and radio also running as she left them.

In the meantime, John must get rid of the old machine, so he scans the QR code and selects "dispose of". A screen appears with information about when he can leave the machine outside his home: a request for a Bulky Waste (Sperrmüll) has been made, and an appointment is set. John's agenda will remind him again next week to take the machine outside the house for the garbage service to pick it up.

3.6 THE OSE PROPERTIES

This section presents a set of key properties that characterise Open Smart Environments and the related concepts presented in this thesis:

ASSISTED OPERATION Ubiquitous Explanations provides an approach for pervasive user assistance (see Section 4.5). Smart products can compose themselves and help solve everyday problems. Furthermore, further mechanisms have been developed to support complex multistep procedures: an Embeddable Workflow Engine (Section 4.6.1).

COMMUNICATING PRODUCTS An architecture for flexible technology embedding for Smart Products (Section 4.2.1), and a service and product discovery based on both Mundo and a custom uPNP and ZeroConf stack (Section 4.2.2).

COOPERATING PRODUCTS Under OSE, different collaboration patterns between products have been defined (Section 3.7). Also, in Chapter three, we introduce the product's lifecycle stages to define what functionality may be available at each stage 3.4.2. In Chapter 6 we further refine it with embedded Situations, a given user's set of available operations is determined in a context-aware fashion (Section 6.4).

LEARNING ENVIRONMENTS A novel approach for collaborative labelling called InSitu is presented (Section 6.5), which allows users to tag the current situation to allow the environment to learn from it. The environment also collects information regarding the pre-defined Embedded Situations in products (Section 6.4).

SPACE AWARE The Space is defined in OSE as a first-class entity (Section 5.3). It allows products to combine with nearby objects. It also provides users indoor guidance (Section 5.4).

PEOPLE AWARE People is also modelled as a first-class entity (Section 5.2). Their description includes their preferences, physical abilities and their skills. This representation can be used for other reasoners, such as AI planners, to combine people and products.

SITUATION AWARE Embedded Situations (Section 6.4) is a new approach for situation assessment based on what people do with the products.

3.7 THE OSE INTERACTION LEVELS

The aim of self-organising Smart Products is to create interconnections and generate structures that lead to positive and beneficial outcomes. In this context, a positive effect is a specific objective towards which a self-organising system should evolve.

Smart Products are typically single-purpose and have a limited set of available operations. This approach aligns with established guidelines in Human-Computer Interaction [13], which advocate for building objects with a clear purpose and function. However, this approach may limit the potential for interactions between products.

Another issue is determining where to implement the smartness and how to define and facilitate new interactions between devices. In Section 3.4.1, we describe Smart Products as the building blocks of smart environments, making them a natural location for embedding all reasoning mechanisms, similar to software agents. However, the cost implications of embedding rich programming and reasoning capabilities in single-purpose products make this scenario unlikely.

In Organized Self-Exploration (OSE) context, we introduce the concept of product interaction levels, which describes a set of behavioural and collaboration patterns as products and services become available. The interaction levels are divided into two categories: *Action-Perform* and *Assess-Anticipate-Adapt*.

This concept implies defining the global goals towards which an assembly of devices will evolve, either explicitly or implicitly. It is important to note that self-organising systems involve the interaction between two or more systems, as a machine cannot change its organisation or functional mappings by itself. Instead, it can only appear to be "self-organising" by being coupled with another machine [28].

We summarise the following aspects that must be taken into account in building self-organising systems:

1. **Goal definition:** how to define a global goal that the system emerges towards.

2. **Interaction design:** what are suitable interactions between components, and how to check that the desired result will emerge during the execution as a result of interactions.

3. **Behavior control:** how to control the behaviour once the application, running without central control, is deployed.

A general description adapted from Gershenson [84], Ashby [28] and Rocha [231] is as follows: *"A system described as **self-organizing** is one which through the interaction of its elements, allows achieving dynamically a desired or positive global function that results in the spontaneous formation of well-organised structures, patterns, or behaviours, from random initial conditions"*

The interaction levels of smart products can be classified into five categories, each with its own unique features.

SELF CENTRIC A smart product operates independently without infrastructure support. It can assist users in completing complex tasks without requiring collaboration with other products. An example is a coffee machine that provides user support for the descaling process.

INFORMATIVE Smart products can sense their environment and gather information from surrounding smart products and services. They can perform an action triggered by a user and broadcast requests for specific information to other smart products. To enable smart products to understand the nature of these requests, they are described ontologically.

COLLABORATIVE Assumes that smart products and services are reachable over a network connection. Products can collaborate with each other to achieve a goal that cannot be completed by a single product. The product controller seeks a solution by gathering information about the abilities to surround products and services and asking a ConflictSolver component to generate an alternative plan.

GOAL FINDING The user asks the environment for action, and the room controller can provide information or search for a solution. For example, a person may enter a meeting room and ask the environment to display a presentation on a particular screen. The room controller assembles a solution by combining several actions to achieve the desired outcome.

GOAL FINDING Involves the composition of non-functional properties to achieve a specific state or value. Products collaborate to achieve topic-based goals, such as energy reduction, heat, light, noise, and radiation emissions.

In summary, the interaction levels of smart products describe a range of collaborative behaviours that enable products to work together to achieve specific goals beyond the capabilities of individual products.

3.8 SUMMARY

This chapter presents the Open Smart Environment Model (OSE) as a novel approach to building smart environments. OSE defines Products, People, Space and Services as its main components. First, it is a more natural way of correlating the construction of smart environments to how people organise their living spaces with the objects they put in them. Second, the model introduces space as a first-class entity, a tool for integrating different smart locations. Finally, OSE also proposes a unique perspective for enabling pervasive smart environments by adding smart support to the various lifecycle stages of products. This approach departs from the current state of the art, primarily focusing on supporting users with smart products in existing environments.

4

BOOTSTRAPPING SMART PRODUCTS

"The real promise of connecting computers is to free people, by embedding the means to solve problems in the things around us"
Neil Gershenfeld.

4.0.0.1 *Publications*

This chapter is based on the following publications:

- Fernando Lyardet, Aristotelis Hadjakos, and Diego Wong Szeto. "In-Situ: An Approach for Dynamic Context Labeling Based on Product Usage and Sound Analysis." In: *Workshop Proceedings of the AAAI-11 Conference*. 2011
- Fernando Lyardet and Dirk Schnelle-Walka. "Ubiquitous Explanations: Anytime, Anywhere, End-User Support." de. In: *The European Jounal for the Informatics Professional, Cepis Upgrade: Special Issue Internet of Things, vol. XII, no. 1* 12.1 (Feb. 2011), pp. 52–58. URL: http://tubiblio.ulb.tu-darmstadt.de/98459/
- Erwin Aitenbichler et al. "Fine-grained Evaluation of Local Positioning Systems for Specific Target Applications." In: *Proceedings of the 6th International Conference on Ubiquitous Intelligence and Computing (UIC-09)*. Springer, July 2009
- Fernando Lyardet and Erwin Aitenbichler. *AmI-Blocks'09: Third European Workshop on Smart Products: Building Blocks of Ambient Intelligence*. Vol. AmI'09 Adjunt Proceedings (electronic format). 2009
- Fernando Lyardet, Erwin Aitenbichler, and Max Mühlhäuser. *AmI-Blocks'08: Second European Workshop on Smart Products: Building Blocks of Ambient Intelligence*. BoD, 2008. ISBN: 978-3-8391-0868-0
- Fernando Lyardet and Erwin Aitenbichler. "Smart Products: Building Blocks of Ambient Intelligence." In: *Constructing Ambient Intelligence: AmI 2007 Workshops; Darmstadt, Germany, November 2007; Revised Papers*. Vol. 11. CCIS. Springer Verlag, Heidelberg, 2007, pp. 156–157. ISBN: 978-3-540-85378-7
- Fernando Lyardet and Vicente Pelechano. "First International Workshop on Web-Enableb Objects (TouchTheWeb 2010)." In: *Current Trends in Web Engineering. 10th International Conference on Web Engineering ICWE 2010 Workshops*. Ed. by Florian Daniel and Federico Michele Facca. Vol. 6385. Lecture Notes in Computer Science (LNCS). Springer Verlag, July 2010, p. 595
- Fahim Kawsar, Fernando Lyardet, and Tatsuo Nakajima. "Three Challenges for Future Smart Object Systems." In: *AmI-Blocks'08: Second European Workshop on Smart Products: Building Blocks of Ambient Intelligence*. 2008, pp. 35–37
- Erwin Aitenbichler et al. "Fine-Grained Evaluation of Local Positioning Systems for Specific Target Applications." In: *UIC*. 2009
- Fernando Lyardet. "Ambient Learning." In: *Handbook of Research on Ubiquitous Computing Technology for Real Time Enterprises*. 2008
- Erwin Aitenbichler et al. "Engineering intuitive and self-explanatory smart products." In: *SAC '07*. 2007
- Fernando Lyardet and Dirk Schnelle. "Consumer Voice Interface Patterns." In: *EuroPLoP European Conference on Patterns and Pattern Languages.* 2007
- Dirk Schnelle and Fernando Lyardet. "Voice User Interface Design Patterns." In: *EuroPLoP European Conference on Patterns and Pattern Languages.* 2006

4.0.0.2 *Contribution Statement*

In this chapter, the Saeco Smart Coffee Machine prototype is presented, which was primarily implemented by Dr. Erwin Aitenbichler in collaboration with Andreas Hartl and Gerhard Austaller. The author of this chapter played a significant role in the conception, original idea, co-implementation of the WDPL activity model, and the evaluation of the prototype.

Furthermore, the middleware used in the Smart Shelf prototype, MundoCore [17], was created and implemented in its entirety by Dr. Aitenbichler, and the Smart Shelf was ideated and implemented by the author of this chapter in collaboration with Dr. Aitenbichler, who engineered the electronics and hardware interfacing over the MundoCore Middleware.

The author of this chapter also created and implemented the Me2it prototype and developed the concept of Ubiquitous Explanations. The author co-defined the architecture and implemented the workflow adapter for Ubiquitous Explanations with Dr. Dirk Schnelle and Tao Wei, a master student at Telekooperation, who implemented the prototype and necessary integration of components.

Dr. Schnelle implemented the VoiceXML interface and integrated the voice recognition interface. Additionally, the author of this chapter and Dr. Schnelle co-developed a complete pattern language for voice interactions with products.

The author of this chapter also created and implemented MundoNubez in its entirety, and Prof. Dr. Max Muhlhauser supported the idea generation and conceptual design for the Smart Products architecture.

4.1 INTRODUCTION

This chapter introduces the Smart Product component of the Open Smart Environment Model, which consists of physical devices or software services enriched with information and capable of communicating with each other. The chapter presents an architecture, description, and required interfaces for Smart Products to interoperate effectively and supporting components that enable scalable sharing and updating of information among products to support complex activities.

The chapter also explores interaction mechanisms with Smart Products, including connecting and browsing physical objects and supporting user learning through ubiquitous explanations. These insights and guidelines for effectively implementing and integrating Smart Products within the Open Smart Environment Model can facilitate seamless communication and interaction between devices to enhance overall system functionality and user experience.

4.2 SMART PRODUCTS

4.2.1 *Architecture*

Smart products encompass physical devices or software services equipped with knowledge about themselves, their surroundings, and

other products. This knowledge is organized into layers according to the level of abstraction they address: Device/Network Layer, Embodiment, Controller, Embedding, User Model, and UI. Figure 4.1 illustrates a conceptual reference architecture that demonstrates this separation of concerns, enabling the integration of different vendors and their technology. Adopting a Service Oriented Architecture (SOA) approach allows devices to implement the functionality above and user adaptation capabilities, combining the limited embedded infrastructure with services available over the network.

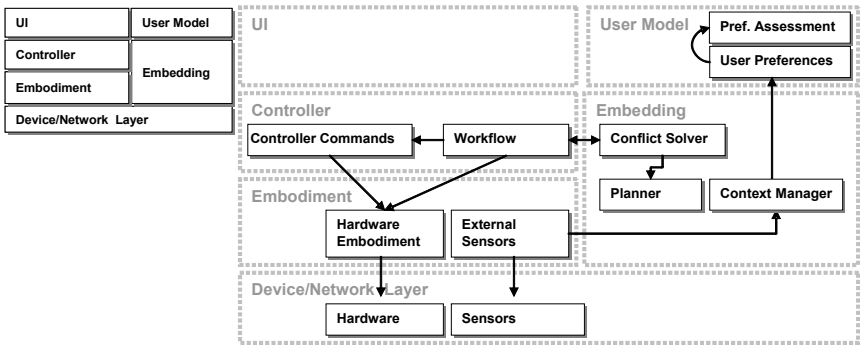


Figure 4.1: An Overview of the Smart Products Architecture

The *Device/Network Layer* constitutes the first layer, where runtime operating software or firmware resides in embedded systems. Processing power at this level operates actuators, sensors, I/O, and user interfaces (such as LCDs, status LEDs, and buttons). Knowledge embedded in this layer delineates a set of Event Condition Action (ECA) rules that govern state transitions. These rules define the device's functionality, implement smart behaviour, and ensure hardware integrity and safe operation. In the context of this architecture, the Device Layer establishes the foundation for product functionality, including the set of valid events, states, and functions that compose the product's available functionality. Additionally, it may specify communication protocols, message formats, and other service-specific details.

Smart products express their functionality in commands that users or other smart products can execute. Commands can range from simple single-step actions to complex, multi-step procedures involving users, tools, consumables, and other smart products. A *Controller* drives the execution and verifies whether any given operation can be performed according to its specifications. When a conflict arises, the Controller delegates the issue to a *Conflict Solver*, which seeks a solution (either fully automatic or involving human intervention) whenever a request cannot be fulfilled, by leveraging the knowledge available in the Embedding.

The *Embedding* enhances a product's "smart" capabilities by enabling it to gather data and process information, encompassing the knowledge a smart product possesses about itself and its operating

environment, including other smart products. This information is obtained through service discovery mechanisms. The Conflict Solver, part of the Embedding, searches for a solution. One strategy involves querying the reasoner to determine a plan for achieving the activity's goal, considering the current situation. The reasoner, based on a planning system (hence the name *Planner*), uses information from an internal facts database to resolve the query submitted by the Conflict Solver. The internal facts database is a small blackboard, updated continuously by the Context Manager. It constantly queries the environment and other smart products for new information to address specific needs and inform itself about environmental changes. The *Context Manager* is responsible for maintaining the internal information blackboard up-to-date.

The *Embodiment* refers to the physical manifestation of a smart product, i.e., how it is designed and structured to interact with its environment and users. Embodiment covers ergonomics and aesthetics in manufacturing, considering factors such as the product's size, shape, and materials. In smart products, it relates to technical characteristics, providing the APIs to access to the device's hardware functionality and sensor data.

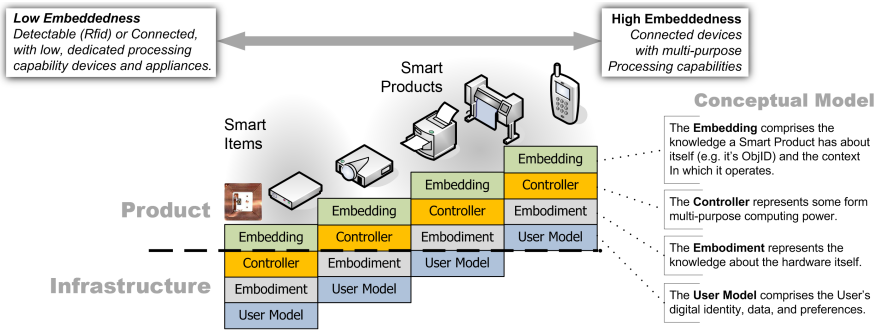


Figure 4.2: Adaptive distribution of the Runtime Environment

4.2.2 *Discovering Smart Products in the Network*

The Smart Products architecture benefits from two different service discovery mechanisms: UPnP [120], and ZeroConf [49].

In this thesis, the UPnP stack was developed to allow devices to advertise their services to control points on the network. When a control point is added to the network, it searches for devices of interest. The UPnP discovery protocol is based on the Simple Service Discovery Protocol (SSDP)[43]. However, the SSDP approach to detecting devices by broadcasting advertisement packets proved particularly sensitive to network configuration and different TCP/IP stack implementations. For scenarios with more than three devices is the support of ZeroConf compelling, with its more robust network awareness, capable of au-

tomatically obtaining a valid IP address without needing a Dynamic Host Configuration Protocol (DHCP)[70] server, using technologies like IPv4 Link-Local Addressing [46] or IPv6 Stateless Address Autoconfiguration[211], DNS-less automatic hostname resolution and more robust service discovery using protocols like mDNS[47], DNS Service Discovery (DNS-SD)[48] or Service Location Protocol (SLP)[90].

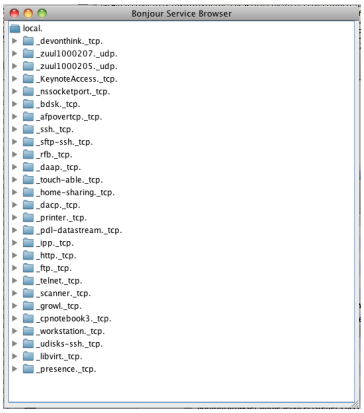


Figure 4.3: A Simple Network Service Discovery Browser combining 3 different service discovery mechanisms.

4.3 SMART PRODUCT EXAMPLES

4.3.1 *Smart Coffee Machine: Basic Smartness*

The Saeco coffee maker is a regular off-the-shelf coffee machine. When used out of the box, the user can choose between three kinds of coffee, namely espresso, small coffee, and a large coffee. A coffee bean container and a water tank are also attached to the machine. If either is empty, a small display on the machine prompts refill them. The display also prompts to empty the coffee grounds container if it is full.

We introduced some modifications to allow us to control all the buttons remotely and determine the machine's state (Figure 4.4). With this state information and additional RFID readers, we can detect user actions and automatically start processes or proceed in a workflow.

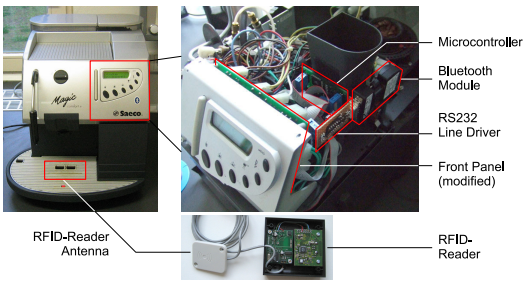


Figure 4.4: Hardware components added to the coffee machine

The hardware of the enhanced system consists of the coffee machine, two RFID readers to identify cups, one reader for reading the digital keys, and a PC running the control software. Figure 4.5 shows the hardware architecture and the individual components. The roles of the individual components are as follows.

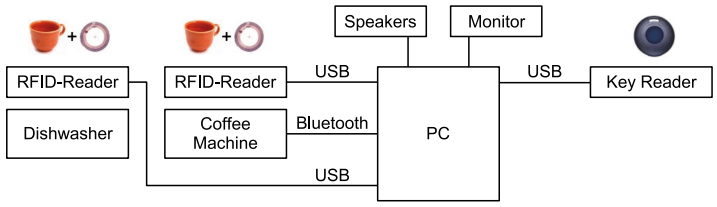


Figure 4.5: Hardware architecture

Coffee Machine:
Because the machine does not have a data interface, we modified the front panel circuit board and attached our microcontroller. The machine communicates with a Bluetooth module on a computer running the controller software. This allows us to detect key presses, simulate key presses, check if the water tank is empty and read the pump control signal. The latter indicates that the machine dispenses fluid and accurately measures how much liquid has run through the pump; one pulse on this signal corresponds to 0.4 millilitres.

RFID reader at Coffee Machine: The antenna of this reader is attached to the bottom of the coffee machine's cup holder surface. It is positioned so the reader can identify cups as soon as they are placed below the coffee dispensing unit. The RFID tags are glued to the bottom of the cups. This allows the system to start brewing coffee once a cup is put down.

RFID reader at Dishwasher: The second RFID reader is placed next to the dishwasher. Users can swipe cups over this reader before putting them into the dishwasher.

Key reader: Our reader for the digital keys consists of a SimonsVoss Smart Relais, a microcontroller, and a USB interface. Like the electronic locks in our computer science building, the relais can be triggered with the digital keys given to all employees and students. Every user owns a key, and there is a one-to-one relationship between users and keys, which makes these keys highly suitable for identification purposes. In addition, the keys can already serve as simple interaction devices. Because users are required to activate them explicitly by pressing a button, the reception of a key ID can be directly used to trigger actions.

PC: The PC hosts most services and is hidden in a cupboard. It gives users voice feedback via the speakers. Users can view and interact with their favourite web pages on the monitor. The monitor, keyboard, and mouse are optional and not required to use the system's core functions.

4.3.1.1 *Interaction Design*

In the interaction design of a smart product, we distinguish between simple and complex interactions and support them in two different ways.

Simple interactions are one-step functions generally triggered with a button. These interactions should be natural to users. This should be true even if they use a product for the first time. The product should behave as the user intuitively expects. Such interactions are implemented by monitoring the user's actions with suitable sensors and typically do not involve graphical or voice user interfaces. An example of a simple interaction is when users put their coffee mug under the dispenser and automatically get their favourite coffee.

Complex interactions refer to multi-step procedures that require the user to have product-specific knowledge. Such interactions involve graphical or voice-based user interfaces, and the system guides the user during these interactions. An example is de-scaling the coffee machine, which requires the user to perform several manual steps.

As many interactions as possible should be designed as simple interactions. To identify actions that are natural to users and to verify that designed interactions are intuitive, user studies must be performed at multiple stages in the product design. An important goal of our Smart Coffee Machine project was to evaluate how average users (i.e., non-technical people) could interact with ubiquitous applications and appliances naturally and intuitively.

4.3.1.2 *Evaluation*

The implementation has gone through several phases, and in each phase, we performed a user study to evaluate what things needed to be changed and how the users felt about using the machine. In the following, we will present our experiences to illustrate the lessons learned from our user tests. The machine was in daily use in a group of 16 people. Five people did not drink coffee and were excluded from the trial; eleven users served as the primary test subjects.

- Group 1: the lab members involved in developing the device: 5 people. This group only participated in the first user test and was excluded from the rest.
- Group 2: 11 people were used as main test subjects in the user testing. In this group, we have a mixture of researchers in computer science, technical staff, and non-technical users (e.g., secretaries).
- Group 3: 5 people who are not using the smart product enhancements of the coffee machine for several different reasons.

The number of users who use our smart product enhancements of the machine increased from the initial 4 to 16, representing 100% of the people drinking coffee. Several new ideas have emerged from the users and have been implemented in the system. All these modifications were also validated in our user tests.

4.3.1.3 *Initial Implementation*

We marked coffee cups with RFID tags in the initial design, and a user automatically gets coffee when putting their coffee cup under the dispenser. We have cups in many sizes, ranging from small espresso cups to large coffee mugs. The cup database stores the RFID tag IDs, the type of coffee (espresso, small, or large), and the cup size in millilitres.

After interviewing the users to get their subjective impressions, we found out that users preferred the smart machine because it requires no user attention to trigger the machine or control the amount of coffee they want. All users cited this "un-attended nature" of the machine as a benefit.

We programmed a notification service to email all the people using the kitchen when beans were close to running out. Since a service controls the coffee machine, the service also can log statistical data. In particular, the service knows how much coffee beans are left and how long they will last.

4.3.1.4 *Error Conditions*

In case of a problem such as running out of water, in addition to the standard error messages on the machine's display, the user receives audio instructions on resolving the problem (e.g., "Please refill the water."). After the user has fixed the problem, the machine automatically resumes the requested operation. All feedback from the system is also over voice, delivered through a Text-to-Speech engine.

With audio feedback, users changed their behaviour when handling error situations with the machine. The most common errors are a complete coffee grounds container and the machine running out of water. We observed that in case of a lack of audio feedback, the users would complain that the system was "broken" instead of looking for the problem themselves on the display as they did before. The machine behaved exactly as before (i.e., it showed the error message), but users no longer read the error messages from its display. We concluded that this was the result first of the expected consistency in the machine's behaviour. Second, although listening is slower than reading, it requires less effort, which could make it a more appealing modality for consumers.

4.3.1.5 *User Tests*

Following the modifications, two formal user tests were conducted, which involved observing users operating the coffee machine and participating in free-form interviews. The overall results were highly encouraging. Most users preferred the automated machine, despite the slightly more complex procedure and marginally longer time required to obtain coffee. It should be noted that acquiring coffee typically takes 15 to 25 seconds, depending on the type, rendering an additional delay of one or two seconds relatively insignificant.

The second set of tests concentrated on other aspects of the coffee machine's operation, such as voice feedback, association, alternate coffee selection, and subjective user perception. Input provided through voice was largely considered advantageous. However, numerous users (two-thirds) encountered difficulties comprehending the voice output in certain instances. Potential reasons for this issue include the following: When users hear a specific message for the first time, they may be caught off guard and fail to fully grasp its content, leading to the impression that the messages are difficult to understand. Additionally, voice feedback may coincide with the machine's operation, and the accompanying noise can make it challenging to discern the voice output. To address this problem, "subtitles" were introduced, ensuring that all voice output is concurrently displayed as text on the computer screen.

An additional issue emerged when users expressed curiosity about selecting a different coffee type without altering the default settings. Such a situation occurs when a user desires an alternate option, such as an espresso instead of their usual large coffee. Relying solely on a graphical user interface for this purpose was deemed insufficient. To accommodate users who might prefer varying coffee choices at different times, an "override button" was introduced. Conveniently, the coffee machine featured an extra button without a designated function, repurposed as the override button. Upon pressing this button, users can opt for their desired coffee variety rather than receiving the type programmed for their cup.

4.3.1.6 *Results*

Primarily, the experience emphasises the significance of user-centred design. Ubiquitous applications often venture into new and uncharted domains, making it difficult or impossible to predict optimal system usage. Few features from the initial design withstood the two user tests, even though the design appeared viable at the time. Specifically, employing the digital key for user identification was a positive experience. This outcome suggests that users are more inclined to accept additional actions based on familiar items from their daily lives, even if the applications differ.

Secondly, only specific modalities can be combined within a system with a multi-modal user interface. This was demonstrated by users neglecting to consult the machine's display. The issue was resolved by channelling all system output through audio. However, users did not encounter difficulties with varying input and output channel modalities. Combining "tactile" input and auditory output did not challenge the users.

4.3.2 *Smart Coffee Machine: Complex Smart Interactions*

The subsequent discussion outlines how the extensions to the machine can assist users in navigating complex procedures, such as the machine's cleaning (or de-scaling) process. Regular cleaning is essential due to the buildup of limestone within the machine. The procedure is relatively intricate and involves filling the machine with cleaning liquid, operating a specific mechanism that includes pressing the appropriate buttons and manipulating the water tube, waiting for an extended period, and ultimately flushing the machine before it is ready for use again. Most individuals in the user group are unfamiliar with this process.

The objective of implementing task support is to demonstrate how the modifications can aid users in managing complex tasks. Cleaning was selected as the task because it must be performed periodically, and although the process is detailed in the manual, most people need to consult it. Even if they read it, they often require the manual's guidance whenever they need to clean the machine. The cleaning process is well-suited for support since some steps can be automated (controlled by the system) while others necessitate manual user interaction. Moreover, there are lengthy waiting periods involved. The implemented system aids users by informing them of the next step at any given point. The system executes any steps that can be performed automatically and only requests user assistance when needed. Additionally, when a specific process step requires a long duration, the machine informs the user so they do not need to wait by the machine. A notification system has been implemented that sends an instant message to the user when a lengthy step is completed.

Internally, this process is depicted as a workflow (Figure 4.6). The XML Process Definition Language (XPDL) is utilized as the data format [50], with the JPEd graphical editor for editing workflows [181] and the OpenEmcee Microflow Engine [160] for executing workflows. A small Perl script has been developed to translate XPDL descriptions into OpenEmcee's proprietary XML format.

In the described system, all communication between services relies on MundoCore and employs channel-based publish/subscribe. A generic activity and transition class for the workflow engine were defined to interface with this publish/subscribe system.

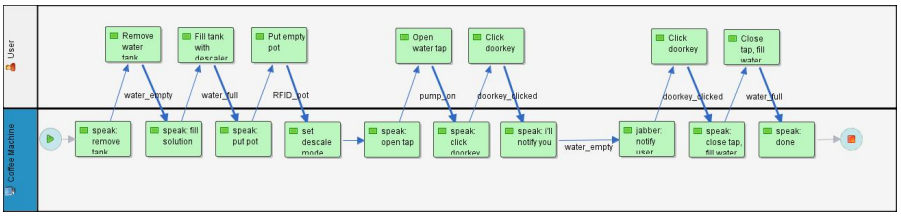


Figure 4.6: De-scaling process modelled in XPDL using XPEd

According to the reference architecture, workflow descriptions become part of the Embodiment Module and are executed by the Controller. In this example, the Embedding Module performs a basic adaptation of the process by stating the user's name, as provided by the User Model.

4.3.2.1 *Activities*

XPDL permits the assignment of an arbitrary number of *extended attributes* to *activities* and *transitions* within the workflow. These activity attributes are utilised to describe the action that should occur. An action can be a message send operation or a remote method call. Method calls rely on the dynamic invocation interface of MundoCore, which enables calling any method of any remote service. For instance, to produce output text via the Text-to-Speech engine, the subsequent attributes are employed:

```
channel   = "tts"
interface = "org.mundo.speech.synthesis.ITextToSpeech"
method    = "speak"
p0        = "Please remove water tank"
```

The channel property specifies the name of the channel to which the invocation request should be sent. The TTS service is subscribed to a channel named "tts". The message distribution is handled by MundoCore and is fully transparent to the application. (It should be noted that channels are not necessarily global - MundoCore has *zone* and *group* concepts to limit the scope of channels.)

4.3.2.2 *Transitions*

State transitions can be triggered by arbitrary MundoCore events. For example, if the water tank becomes empty, the Coffee Machine service publishes a notification of type `org.mundo.service.saeco.WaterEvent` with the content `empty = true` to the channel `saeco.event`.

MundoCore supports notification filtering based on XQuery expressions [36]. We use this mechanism to describe transition conditions in the workflow. To execute a transition as soon as the water tank is empty, the following extended attributes are specified for the transition:

Figure 4.7: Prototype of the Smart Shelf

```
channel = "saeco.event"
filter  = "for $o in $msg where
            $o[class='org.mundo.service.saeco.WaterEvent'] and
            $o/empty=true()"
```

The transition is executed when the first notification matches this filter expression.

4.3.3 *Smart Shelf: Basic Smartness*

The Smart Shelf is a weight-sensitive device designed to provide automatic in-stock control of the goods it holds. It features an intelligent assessment approach that enables the shelf to report the available stock not as a single class, as previous approaches have done, but as multi-class items without using RFID tags.

4.3.3.1 *Motivation*

Numerous enterprise processes and systems rely on the manual collection of data. As with many other manual, repetitive processes, errors can occur (e.g., mistyped or confused codes), and thus, traditional software systems often need to reflect the real world accurately.

The Internet of Things [30] is frequently referred to as a powerful tool to monitor real-world activities precisely. The combination of identification, sensor, and location technologies applied directly at the point of process execution can help companies gain deeper insights into the operation of their internal processes. Optimised shelf replenishments, such as RFID tags on products and shelves with built-in readers, are well-known examples of real-world awareness. The continuous and automated collection of information about relevant environmental conditions also serves as the foundation for real-time event-driven enterprise management and decision-making processes.

In the retail industry, as Lamarca et. al. note, *"a significant factor for economic success is high product availability at minimal operating costs. Even though considerable investments in improving their supply chain man-*

agement processes have been made, recent studies show that unsatisfactory product availability on retail shelves remains a significant problem. An ordinary retail store faces stockout rates of 5%-10%, which results in a loss of sales of up to 4%. This accounts for a reduction in revenue of hundreds of millions of dollars per year for large retailers such as Wal-Mart, Target or Metro."[150] In retailing and logistics, the main reasons shelves in stores are unexpectedly empty –or Out of Stock Situations (OOS)– have been studied. Many of them are rooted in untimely ordering and replenishing. For instance, merchandise may be in the store but was not on the shelves (38%), or erroneous forecasting was made 32%)[59].

4.3.3.2 *Current Approaches for In-Store Awareness*

The introduction of RFID technologies already at case and pallet levels allows automatic identification and tracking of goods. Furthermore, other smart applications are using RFID and computer vision to implement automated checkout for customers, theft prevention monitoring, real-time product availability and even merchandise misplacements and deadstock detection[149, 150].

RFID and computer-vision approaches are extensively used in logistics and back-store operations but are still not widely adopted on the shop floor. Costs and complexity are hindrances for such technologies to be implemented at a shelf level granularity[149].

On the other hand, other more inexpensive technologies may help bridge the gap to the shelves on the main floor. One such technology is pressure and weight sensitivity sensors, which can be embedded in the shelves, and have several applications. An in-depth discussion of the different approaches research approaches and prototypes with capacitive and weight sensors can be found at [29, 103, 150, 188, 229].

In the Smart Shelf prototype described in this chapter, we use four weight sensors for our test shelf to collect data.

4.3.4 *Smart Shelf: Smart Content Assessment*

To model the contents of a shelf, we assume $k_1, \cdots, k_n$ ***known*** different kinds of products, with corresponding $w_1, \cdots, w_n$ weights (for the time being, we don't consider the natural weight errors $\epsilon_1, \cdots, \epsilon_n$). Therefore, the contents of a shelf could be seen as fulfilling the following equation:

$$w_1 p_1 + \cdots + w_n p_n = totalweight$$

with $p_1, \cdots, p_n$ being the amount of each kind of product $k_1, \cdots, k_n$. This equation is know to belong to the family of diophantine equations. We briefly describe what linear diophantine equations are, and the assumptions we make in order to use them in the general case. The next significant step is the informed reduction of the possible solution

Smart Shelf

Weight Estimation out of Pressure Sensor Readings

The estimation of the shelf weight cannot be done directly using the pressure sensors, since their behavior/reading is not linear.

Therefore, an approximation through curve fitting has been done, using the following formula: $a.b^{c(d-x)}$

	RH	RV	LV	LH	Total
0	398,71	432,29	361,76	436,59	1629,35
1	352,17	359,33	310,08	377,58	1399,17
2	319,69	317,77	276	339,15	1252,62
3	294,33	288,17	256,5	313,42	1152,42
4	269,38	280	251,77	286,62	1087,77
5	257,53	280,8	240,13	259,27	1037,73
6	245,53	272,8	233,73	250,87	1002,93
7	239,75	258,25	227,25	244,42	969,67
8	234,73	243,27	219,93	241,47	939,4
9	230,87	248,07	209,33	228,4	916,67
10	224,64	240,21	207,07	224,64	896,57
11	220,64	230,5	205,71	222,29	879,14
12	217,64	223,21	201,29	222,29	864,43
13	216,46	220,23	197,62	212,31	846,62
14	216,31	219,85	192,69	208,31	837,15

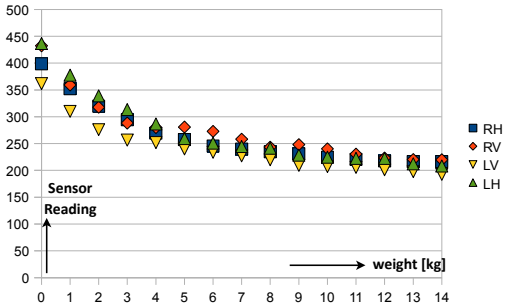


Figure 4.8: In the prototype, four weight sensors are being used to collect data. In this figure we show the data sampling done to build a weight approximation function.

space using the physical properties and limitations of the products and the shelf itself.

4.3.4.1 *Upper Bound Product Parameter Estimation*

Any given shelf can only accommodate so many units of any given product. Since we have the product information, we also know it's packaging dimensions, and using that information we can determine a range of possible numbers of product units to be between 0 and the maximum possible in a given volume.

Definition: Let K be the set of all product types. Then $\forall k \in (k_1, \cdots, k_n)$, we assign a product quantity interval $q \in (q_1, \cdots, k_n)$ such that

$$\forall q_i, q_i \in [0, \cdots, MAX(k_i, SpaceAvailable))] \tag{4.1}$$

$$sum_i q_i s_i \leqslant max$$

In Figure 4.9, the steps for the solution are outlined before introducing the pseudocode below

The following pseudocode implements the calculation of the upper bounds. It takes a product and a bounding box as input and calculates the maximum number of products that can fit in the bounding box by dividing the dimensions of the bounding box by the dimensions of the product, flooring the results, and then multiplying the maximum number of products that can fit in each dimension.

The pseudocode in Figure 4.10 takes a product and a bounding box as input and calculates the maximum number of products that can fit in the bounding box by dividing the dimensions of the bounding box by the dimensions of the product, flooring the results, and then

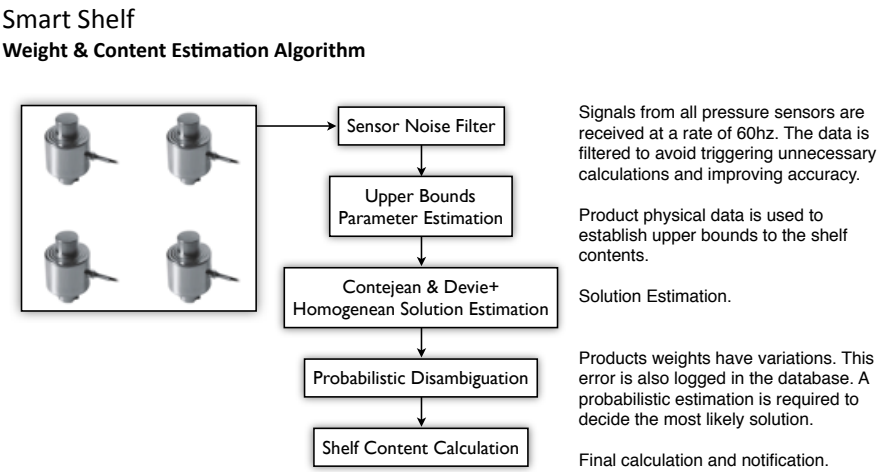


Figure 4.9: An overview of the Smart Shelf solution.

multiplying the maximum number of products that can fit in each dimension.

```
function calculateProductFit(product, boundingBox):
    boxWidth, boxHeight, boxDepth = boundingBox.dimensions
    productWidth, productHeight, productDepth = product.dimensions

    // Calculate the maximum number of products that can fit in each dimension
    maxFitWidth = floor(boxWidth / productWidth)
    maxFitHeight = floor(boxHeight / productHeight)
    maxFitDepth = floor(boxDepth / productDepth)

    // Calculate the total number of products that can fit in the bounding box
    maxFitTotal = maxFitWidth * maxFitHeight * maxFitDepth

    return maxFitTotal
```

Figure 4.10: Pseudocode implementing the calculateProductFit function

The pseudocode in Figure 4.11 iterates through all possible combinations of up to three products, calculates the maximum quantity that can fit in the bounding box for each product using the *calculateProductFit* function above, and then iterates through all possible quantity combinations within the given constraints. It finds the best combination with the smallest weight difference within the specified tolerance or margin of error.

The pseudocode in Figure 4.12 implements the function *calculateDiophantineCoefficients* calculates the Diophantine coefficients for up to 3 product weights (2 or 3) using the Contejean & Devie Homogenean Solution Estimation[55]. It tries all possible combinations of coefficients within the range of 0 to totalWeight and checks if the current weight (the sum of the products of coefficients and weights) is within the given tolerance. If so, it adds the coefficients to the list of solutions. If at least one solution is found, it returns the list of solutions; otherwise, it returns null.

```
function findBestCombination(productCalculations, totalWeight, tolerance):
    bestCombination = null
    minDifference = infinity

    for each combination of productCalculations (up to 3 products):
        productWeights = [product.weight for product in combination]
        productMaxQuantities = [product.maxQuantity for product in combination]
        coefficients = calculateDiophantineCoefficients(productWeights, totalWeight,
                       tolerance)

        if coefficients is not null:
            // Apply bounds on coefficients according to productMaxQuantities
            validCoefficients = applyBounds(coefficients, productMaxQuantities)

            if validCoefficients is not null:
                currentCombinationWeight = sum([coeff * weight for coeff, weight in
                                           zip(validCoefficients, productWeights)])
                currentDifference = abs(totalWeight - currentCombinationWeight)

                if currentDifference < minDifference:
                    minDifference = currentDifference
                    bestCombination = createProductCombination(combination,
                                      validCoefficients)

    return bestCombination
```

Figure 4.11: Pseudocode implementing the findBestCombination function

```
function calculateDiophantineCoefficients(productWeights, totalWeight, tolerance):
    solutions = []

    for a in range(0, totalWeight + 1):
        for b in range(0, totalWeight + 1):
            if len(productWeights) == 2:
                currentWeight = a * productWeights[0] + b * productWeights[1]

                if abs(currentWeight - totalWeight) <= tolerance:
                    solutions.append([a, b])

            else: # len(productWeights) == 3
                for c in range(0, totalWeight + 1):
                    currentWeight = a * productWeights[0] + b * productWeights[1] + c *
                                    productWeights[2]

                    if abs(currentWeight - totalWeight) <= tolerance:
                        solutions.append([a, b, c])

    if len(solutions) > 0:
        return solutions
    else:
        return null
```

Figure 4.12: Pseudocode implementing the calculateDiophantineCoefficients function

4.4 ME2IT: BROWSING AND CONNECTING LOCAL OBJECTS

People encounter products, devices and appliances as a regular part of their lives. In many such situations, it is common for people to wonder about further information about a given object, either to install it, use it, repair it, dispose or to know what is it, who builds it, how much it costs and where to buy it. Today, users can do very little in most such encounters other than resort to a printed manual if available.

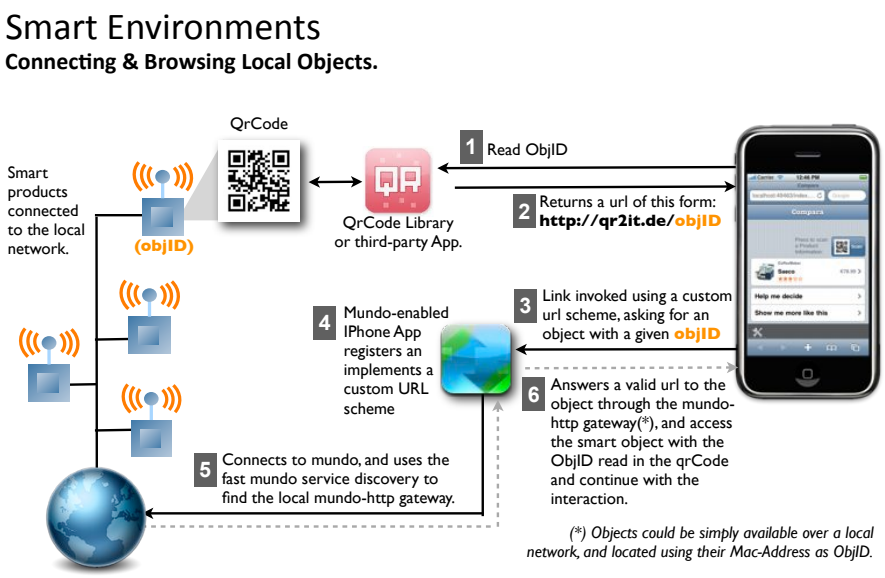


Figure 4.13: Me2IT: Browsing physical devices.

In order to overcome these limitations and facilitate the availability of information, different kinds of tagging technologies such as 2D barcodes, RFIDs and QRCodes have been developed to enable users to identify objects and retrieve information about them effortlessly. Although tagging allows identifying an object, users cannot simply connect to that device and retrieve information about its current status and functions since mapping from a physical object to an addressable network entity is a non-trivial one.

4.4.1 *Architecture*

This section presents an approach to enable physical to network mapping using standard QR codes. QR codes have become the de-facto standard for tagging objects in a way any user can capture its information simply by pointing to it with their mobile camera. The information embedded in the so-called QrCode tags is usually a URI address (although there is no restriction on what contents may be encoded in the tags). The solution shown in Figure 4.13 introduces a custom URI scheme extension, that is, the definition of a new top-level part of a URI (such as HTTP, FTP, tel, or smb) and introduce a new one that browsers can recognize and use the information encoded in the remaining parts of the URI to detect and connect to the physical object a user is pointing to. The downside of this approach is that at least a minimal URI handler must be installed for the browser to recognize it. The advantage is that how people use QR codes today and all existing software remains unchanged.

4.5 UBIQUITOUS EXPLANATIONS FOR SMART PRODUCTS

This section presents a new approach to supporting people in their daily encounters with technology. Be it at home, in public spaces, or on the road, people often need to solve little puzzles when facing new technology to carry out a task. As products become more sophisticated, their operation becomes less obvious or less self-explanatory, that is, the degree to which a person can reasonably explain an object's functions and operations.

Many technologies exist today to assist people facing new situations and technologies. However, providing such assistance remains a complex problem: it involves different aspects, such as how relevant and easy it is to understand the instructions, the modalities used to deliver the assistance and the limitations posed by real-world scenarios.

Traditionally, user assistance has been delivered in the form of printed manuals. However, manuals are seldom used, and when people must resort to printed materials, they do not like them. Rather, many users confronted with a new device would tinker a few minutes with it and, if unsuccessful, may return a perfectly functioning product as "broken" [13]. A similar documented phenomenon is known as *The Paradox of the Active User*[42]: many users are not interested in learning but in using the artefact immediately. It is a paradox because reading the instructions would save users time. Printed documentation does have limitations that may render it impractical in many situations:

- it is linear, where the tasks are to be carried out in only one sequence
- it does not address situations where users get stuck and need specific, short guidance on how to continue [26]
- the information available is the same regardless of the user's expertise.

However, people still need instructions. Moreover, people need more instructions today than ever before. Embedded electronics increasingly expands and introduces new functions that are either more silent or abstract (such as programming features). Many such features cannot be perceived or understood by merely watching them operate, rendering devices and machines increasingly opaque to our senses [13].

For Open Smart Environments to grow organically and become pervasive, we need an approach to support users in their interactions with the surrounding technology, wherever such encounters occur. Therefore, we introduce a new approach called *Ubiquitous Explanations* for delivering end-user assistance through voice-controlled, multimodal explanations.

Ubiquitous Explanations takes advantage of SIP (Session Initiation Protocol)-based communications on trusted [178], personal mobile

devices such as mobile phones to allow users access to explanations anywhere, regardless of the available multimedia capabilities on-site or the targeted object. We also present a modelling approach and software support for explanation adaptation to different user knowledge levels, enabling people to dynamically adjust the granularity of the instructions they receive.

4.5.1 *Requirements for Good Explanations*

Writing and designing explanations must meet several desiderata to be effective. Current approaches can be organized into two:

industrial standards such as the DIN 8418 [64], and

linguistic studies [189].

. Both approaches stress that user information should be considered part of the product, readily available in text or iconic forms. As an industrial norm, it also provides some broad recommendations on what sort of explanations there should be when they are delivered as printed manuals:

- The form and degree of detail can vary with the product itself and the expected knowledge of the users. The descriptions should be easy to understand.
- Texts and figures should be ordered as if the product were used for the first time.
- The structuring of the sections should be clear and how users think.
- They should answer predictable questions such as *why* and *what*.
- Descriptions should be short and precise while focusing on the core issues.
- Any jargon used should be explained.
- The names of the section titles should be a problem- and use-oriented.
- Images can be used to support the text and should be easily related to the corresponding text passages.
- The use of special icons or abstractions should be explained.
- It should also be easy to read, something which can be achieved by summaries.
- There may be included tables, colors, with a clear layout and in a suitable font.

Other approaches to creating good explanations can be found in linguistics. Schmidt analyses, in [189], the criteria for helpful user information: she demands that user information be understandable, complete, correct, short, precise, simple, and should have a consistent representation. An essential requirement is that users can understand the information contained. This also includes the avoidance of jargon in favour of simple language constructs that a wider variety of user groups can understand. Terminology allows people to describe something in a short and precise fashion but may not be understood by everybody. User information should aim at an active use of the product.

Furthermore, it should not be necessary for users to have a detailed understanding of the processes and causes of their actions. The increasing functionality of the devices leads to a less detailed knowledge of understanding. For example, nearly everybody can drive a car without understanding how the motor works. The level of understanding depends on the sender of the information but also the recipient. Users will have different knowledge of the details depending on the background knowledge. Too much information will confuse users with only little background knowledge.

User information also has to be short and precise. This implies a trade-off between being short on the one hand and not losing information in terms of being complete on the other hand. It is also necessary to repeat relevant information to handle the product safely. Images are a suitable means of helping in these cases. Images should not repeat the information given in the verbal description but enhance it or give an overview.

4.5.2 *A Model for Adaptive Explanations*

The previous section showed that the user's experience and expertise are key to designing good explanations. Today, manufacturers must define which users should be supported and how much explicit information is given (externalised) or left to understand (internalise). In both cases, the data is mainly fixed once defined without allowing it to adapt to the user and their preferences. In this section, we describe a model for adaptive explanations. We borrow from concurrent task trees [163] the representation of goals using a tree. The depth of the tree represents the knowledge of the user. The deeper the tree is traversed, the less experienced the user is.

4.5.3 *Prototype*

For our demo implementation, we limited the experience levels to expert, advanced, intermediate, and novice. Technically, there is no limitation on the number of levels, which can be easily adapted to a

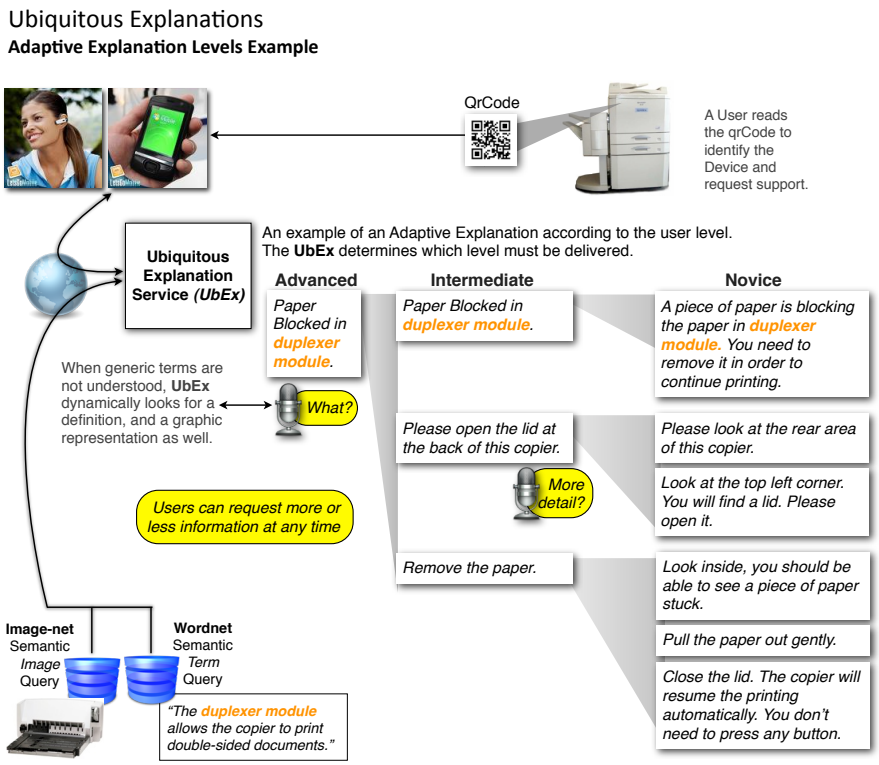


Figure 4.14: Example of an explanation for different skill levels.

custom set of experience levels. Each node represents an activity of a workflow that is being followed to achieve the user's goal. This means that, in an ideal case, a single step is sufficient to guide expert users.

4.5.4 *Profiles and Perception-Level*

According to Lyardet et al. [142], the following criteria describe an explanation: detail level, perception level, and visual information. The detail level is directly related to the requirement for short and precise explanations described above. A shorter explanation has a higher cognitive load for the user than longer and more detailed explanations. Jargon can also be used to shorten the explanation. This can be stored in a profile. Linguistic factors, as described above, are always relevant since, e.g., a setting of 90% correct information does not make much sense. All factors are rated by perception level with a degree from 0 to 100. Perception levels are related to a sense: seeing (e.g., colour, shape, volume), hearing (e.g., frequency, duration) and touching (e.g., kind of surface). The types and values of the different perception levels can be used to find representative values that can be used to determine the relevant explanation. This can be done with the help of the Multi-Attribute Utility Theory (MAUT) [185]. MAUT transforms the different factors into a multi-dimensional space and integrates them with the help of a weighted sum function.

The profile is used to determine how the system communicates with the user. In an ideal case, there is no further adaptation needed. However, the environment may influence how the user perceives the presented information. For instance, if tired, the user may be unable to concentrate on the given explanation. Hence, skill adaptation will sometimes be necessary.

4.5.5 *Skill Adaptation*

There are three options for adapting the user's skill level: direct, time-based and event-based. Direct means an explicit action by the user requiring more or less information. The system dynamically initiates the latter two. If the user does not react within a predefined period, it can be assumed that they are either confused or did not understand the given explanation. The first time this happens, the explanation may be repeated. An escalation decreases the skill level at the second occurrence. The third alternative is triggered by sensors detecting that the user performed the wrong actions. This is not always possible since not all devices have sensors. Skills are adapted event-based using the same escalation strategy as time-based adaptation. This corresponds to an established technique of voice user interface design. Cohen names *Detailed Escalation* [52]. The options are not exclusive but are based on each other and are executed in the order described.

4.5.6 *Dynamic Explanations*

The concepts introduced so far rely on a task tree that is known in advance. In this case, the required knowledge has been provided by the manufacturer. However, real-world scenarios may prove that the given information cannot be used by, e.g., novice users. In this case, a more dynamic approach is needed to create adapted user information on the fly. In a mobile setting, additional information can be acquired, e.g., via a wireless network to adopt additional knowledge. A common way to share knowledge is through the use of ontologies. In the following, we will explain their use by the example of tools needed, e.g., to correct a blockage of the copy machine. Imagine a user confronted with a term well established in the manufacturer's vocabulary and used throughout the explanations because the manufacturer can not imagine that someone does not know the correct name for the tool to use. On the other side, the user cannot act because of either understanding what is supposed to be done or how to handle that tool.

We use WordNet[1] for this purpose which features good access to information. A search, e.g., for *screwdriver*, returns both types of information we are interested in: (1) a short description of the tool,

1 http://wordnet.princeton.edu/

(2) a description of how to handle it. Other sources of information, such as ImageNet[2] can also be used to acquire images of the search. Searches are performed by means of ontologies. These information sources are integrated into the Ubiquitous Information to provide further help on demand.

4.5.7 *Ubiquitous Explanations Service (UbEx)*

In this section, we introduce *Ubiquitous Explanations*, a new approach for supporting users everywhere they interact with technology. This approach uses the current mobile platform and communications advances to deliver instructional assistance. It borrows from ubiquitous computing concepts of pervasively available computing power and communications.

However, unlike previous approaches, we focus on developing a mechanism to be as widely available as possible across different devices. Therefore we tap into the communications and computing power carried by people in the form of mobile phones, music and game consoles. A centrepiece of this approach is our use of SIP (Session Initiated Protocol [156]) -based communication between the user terminal and the Ubiquitous Explanations Service (UbEx). SIP enables UbEx multimodal voice and data communication to and from the user. UbEx interprets voice and user actions as commands or queries, and the answers (in this case, the required explanations) are delivered to the user with data (text and graphics) and voice (TTS generated).

In the following sections, we introduce the requirements and architectural approach implemented for Ubiquitous Explanations and provide a detailed description of the architecture.

4.5.7.1 *Requirements and Assumptions*

Providing support to help people with the technology they encounter is based on the following assumptions:

A ME (MINIMAL ENTITY) : *ME* devices are a representation of their users in the digital world [16]. A small wearable computer that is easy to carry around provides minimal key functionality: identity, security, association, interaction, context awareness, and networking. Our mobile phones enable a true mobile Internet experience and have become today's physical token for our *Digital-Me*.

SPEECH-BASED : to allow users a simple and natural interaction mechanism, and also be useful in situations where eyes-free, hands-free support would be desirable.

2 http://www.image-net.org

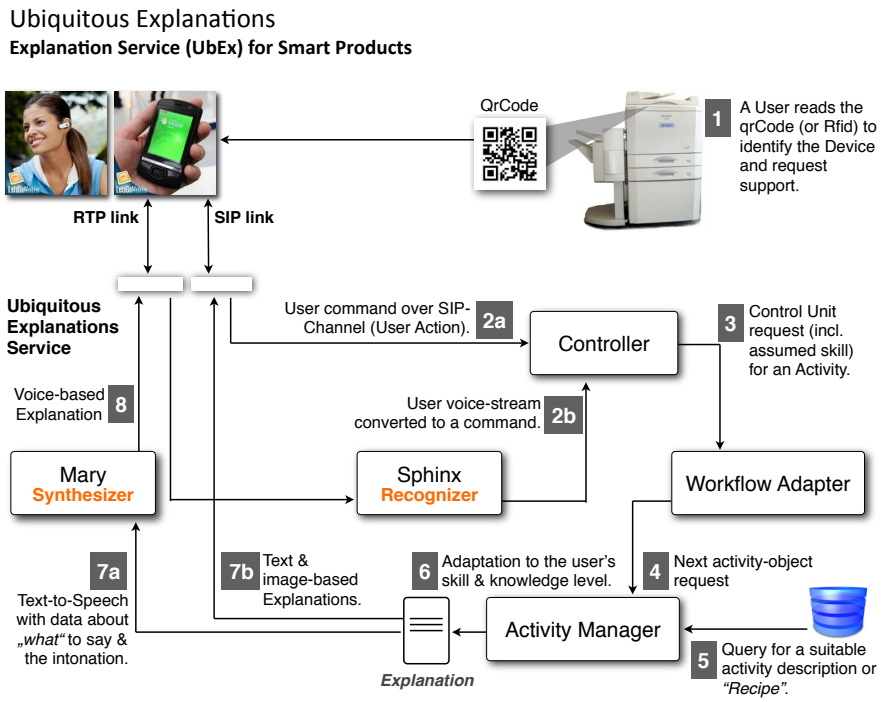


Figure 4.15: Ubiquitous Explanations Architectural Overview

ZERO DEPLOYMENT : Today, mobile devices are pervasive, but their execution capabilities and requirements differ greatly. Therefore, to provide a scalable solution, the solution must rely on the capabilities of the mobile "off-the-shelf". Therefore, we selected a SIP-based solution for voice communication and a standard web browser to deliver visual content.

SENSOR FEEDBACK : On many occasions, such as multi-step procedures, it is desirable to receive confirmation that a prescribed action has been carried out to continue. If available, such feedback allows the system to provide more natural guidance without further requests from the user.

FLEXIBLE EXPLANATIONS SUPPORT : Specific support is required to allow modelling explanations for users with different skill level.

4.5.7.2 *Architecture*

Ubiquitous Explanations Service (UbEx) can support the user in different stages of a smart product's lifecycle, and therefore, there are different ways in which assistance can be triggered. An example of a typical assistance scenario is depicted in Figure 4.15, where the user encounters a problem and he would request further assistance: the user would first associate his *ME* device (typically a mobile phone) by simply scanning the QR-code available in the product.

The Ubiquitous Explanations software establishes a SIP link that sends and receives commands from both the user terminal and sensor feedback -if available- from the target device. A second link (RTP-based) is also established to support voice communication between the user and UbEx. Whenever the user speaks or triggers a command at the user terminal or by actioning the target device, these signals are forwarded to a *Controller* module that further delivers the request to the *Workflow Adapter* controlling the explanation process. If the user has spoken a command, the signals will be delivered through the RTP link and analysed and recognised using an open-source Speech recogniser called Sphinx [213].

The *Activity Manager* plays a key role since it determines what level of information should be delivered to carry out the next task (communicated by the workflow module). It fetches the required information from different sources and assembles the explanation that must be delivered to the user. Once the explanation has been defined, which contents are to be delivered via voice, visual modalities or both is established.

The information that must be delivered over voice modality is then forwarded to a *Synthesizer* (Mary TTS [121]) together with extra information regarding the intonation in which the explanation is to be delivered. For instance, intonation information is important to help users disambiguate potentially hazardous actions from simple steps. The explanation information delivered in visual modality is pushed to the user terminal through the SIP link.

4.5.8 *Contributions of Ubiquitous Explanations*

While previous approaches have been successful in the proposed scenarios, delivering just-in-time instructions to any product has remained largely unsolved. Providing explanation assistance is challenging and includes several aspects: explanation generation, explanation adjustment to the receiver's knowledge, modalities of explanation delivery, and technology requirements to receive and control the explanation being delivered. On the one hand, conveying different degrees of information according to each user's needs requires special modelling support. On the other hand, many assistance technologies require specific hardware to deliver the assistance. This requirement limits the locations where explanations can be provided to those where such hardware is. Roadie embedded sensors in things can help identify what is going on and provide better feedback. Current limitations to such kinds of embedded functions are that the mechanism for delivering such explanations is one of a kind. We have also experimented with public/open speakers. The downside of such solutions is that they also require extra local hardware.

Several scenarios are available, most of them in similar settings from a Voice User Interface perspective: indoors, TTS delivered over speakers, and custom settings for a given scenario. In this example, we want to provide a mechanism where assistance, not only computing power, can be ubiquitous and provide opportunities for opportunistic learning. For instance, when repairing a car or changing a tire. Those are scenarios where such "common speakers" would not be available.

4.6 SUPPORT FOR COMPLEX BEHAVIOR IN SMART PRODUCTS

The first approach to modelling complex behaviour is by using State Machines. However, more complex, real-life processes can have multiple concurrent execution paths, whereas a traditional state machine can only have one. Furthermore, the typical behaviour of state machines is synchronous, far away from scenarios linking people, services, and networked products that may not be continuously available. Also, the different participants execute actions that may take an indefinite time to complete. Therefore, the process control needs to be asynchronous so that once it triggers an action, it does not need to wait until the processing ends. Complex behaviour support also allows a uniform mechanism for modelling other long-running, concurrent processes such as product lifecycle management and servicing or maintenance.

Smart Products need to support both kinds of behaviour, and they can do so by implementing commands and processes. However, the solution is not straightforward in the latter case since general-purpose programming languages such as Java or C++ have no native notion of a persistent wait state but at a threading level. Therefore, until the introduction of continuations and futures, it was hard to express the overall process in such languages. Even today, with concepts like continuations and futures that significantly simplify handling asynchronous calls to services and APIs, representing more complex interactions is still better described using a workflow language such as XPDL [200], YAWL [54], or BPML [159]. These languages help to explicitly describe complex interactions sequences among different actors and runtime support to execute.

As described in Figure 4.2, there are very different execution environments available, so for our reference architecture, a minimal execution engine based on the Micro-workflow engine [145] has been implemented and componentized into service to allow the flexible deployment necessary to accommodate the different scenarios. Furthermore, since the workflow execution is always situated contextually, it has been extended to support Context for the workflow engine becoming both a context-aware workflow engine and the execution driver for context-aware behaviour adaptation.

4.6.1 *Workflow and Context-Aware Adaptation.*

The Context-aware architecture allows the adaptation of each artefact to the current settings and to respond accordingly. In the case of the Controller, the adaptation required is not in the execution engine itself but in the processes it executes. These processes are sequences of procedures that follow a plan to achieve a specific goal. Therefore, the object adapted to the current Context (the explicit representation of the flow of control) is different from the other execution units in the system and thus requires a different support infrastructure.

The use of workflow in mixed environments is relatively new. In contrast to the original application scenarios of business processes with "digital events" such as the organization and routing of documents, mixed reality scenarios involve processes with real-world events that introduce a degree of uncertainty to which the execution of a process must adapt. This new generation of workflows is known as "Smart Workflow," and can react to world events (also described in "context"), allowing a degree of adaptation to the environment. Physical events such as Context allow the adaptee's (i.e., the workflow in process) execution flow to be influenced by the sensor data values. We call such an adaptation a "Parameterized Adaptation," in that all possible flows of execution of the workflow are known beforehand.

The integration of sensed information to control the flow of activities has been used to provide a degree of process adaptation, taking into account the Context in which the activities are taking place. However, the adaptation capabilities through sensed contextual information as parameterization for ongoing processes are limited by the combinations that can be foreseen at design time and depend greatly on the existence of predefined services that match the activities outlined in the workflow description. Several scenarios benefit from the current state of technology, such as industrial business processes [222] [206], where the behavioural components perform the actual work in each of the activities remain stable over time.

Capturing events from the physical environment is a partial view of all observable data and a component usually missing is the knowledge of the environment components. This information in the literature appears already embedded within the workflow itself [116], as known a priori. As a result of this, it is often the case that the semantic description of the required steps to accomplish a given task is expressed in terms of the concrete constituents (or participants) rather than a description of what should happen at each step.

A different approach is that taken by the Flexi workflow, which decorates existing standard workflow descriptions with an extra, separated description associated to the workflow definition, which generalises the definition of each step in a workflow regarding what is to be accomplished or goals.

Such a more generic description allows a new degree of adaptation: workflows that can execute taking into account the available means (should such a case exist and be possible). We call such workflows: "Context-Aware Workflows." The Flexi workflow comprises an embeddable workflow engine, a workflow grounding analyzer, a Conflict solver, and a Planner. Every time a new workflow is set to be executed, it is first inspected by the grounding analyzer to determine whether the contextual situation allows the workflow to be completed. The analysis is performed by inspecting each activity and checking its behavioural description with the intended, concrete participant described in the workflow description file (WDF) to act. Several conditions may prevent a participant from fulfilling its role: it may be busy in another task, disabled, or simply not there anymore. In case such a conflict exists, the grounding analyzer forwards the intended Goal to the Problem Solver to find a suitable solution by: 1) Searching for existing, predetermined solutions 2) Requesting the context server to scan the environment to find a replacement 3) Requesting the planner to come up with a solution within the current contextual constraints.

The third alternative is to invoke a reasoning module called Planner that utilises a Graphplan-based algorithm to find a solution against an updated knowledge database. When the planning successfully finds a suitable solution, it returns a workflow description of how the given goal will be accomplished in the current Context. Such a solution becomes the execution plan for the activity.

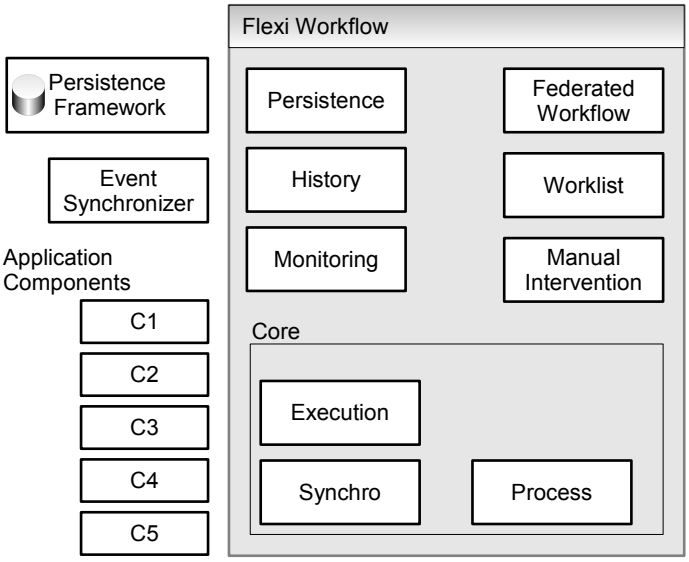


Figure 4.16: Flexi-workflow Architecture

Some scenarios that involve Smart Products call for the dynamic composition of their capabilities to reify ambience functionality. Certain processes are known to be correct for achieving a given outcome

while preserving security and integrity parameters. However, the approaches above of workflow adaptation need to be extended to provide alternatives to supporting scenarios for ad-hoc integration of Smart Products, where given the behavioural components that need to carry out the different activities may vary in their availability and in what they can do over time, introducing the need for human intervention to complete or continue with a given process. Such workflow adaptation support has to adapt how the different activities are carried out using the available means, that is, taking into account the available Smart Products.

The *Flexi-Workflow* presented in this thesis is an extension of the open source Micro-Workflow, developed to allow the use of the available devices to take advantage of them as much as possible and seek human intervention when cannot pursue the goals further.

Workflow systems implement some form of a process model. A process model defines the core concepts and entities and how they interact. Workflows, in turn, express processes instantiating the constructs defined by the process model. The Flexi-Workflow follows an activity-based model because the control flow logic is independent of the control itself, a key advantage if the logic is defined dynamically.

In an activity-based process, each activity comprises units of work called tasks. Tasks are atomic actions executed by a workflow participant (a service or a device). The activity-based workflow is a directed graph configuring an activity network, where nodes are the activities. The arcs or transitions connecting the nodes communicate the data between nodes.

Implementing such a workflow functionality on traditional object-oriented languages has the additional complexity that messages are exchanged synchronously, thus severely limiting the workflow's ability to spread the activity network into parallel activities. For this reason, Flexi-Workflow extended the open-source Micro-workflow to take advantage of the Futures implementation. In this technique, an object mediates between the caller and the receiver by immediately returning a so-called future object (also known as a promise), a placeholder for a reference to an object yet to come, allowing the caller to continue with its execution. The value placeholder will be filled transparently with reference to the result object whenever the task returns it. A good description of the implementation can be found on [145], where the implementation resorts to a technique called "trampolining". The Micro-Workflow author explains *"Each type of activity node builds and returns a different continuation to the trampoline. The activity-specific processing occurs in these continuations in response to the applyContinuation: message. Two parallel class hierarchies provide the functionality required to define and execute workflows. The first hierarchy belongs to the process realm and provides the abstractions for the activity-based process model. Developers use instances of these classes to define workflows."*[145]

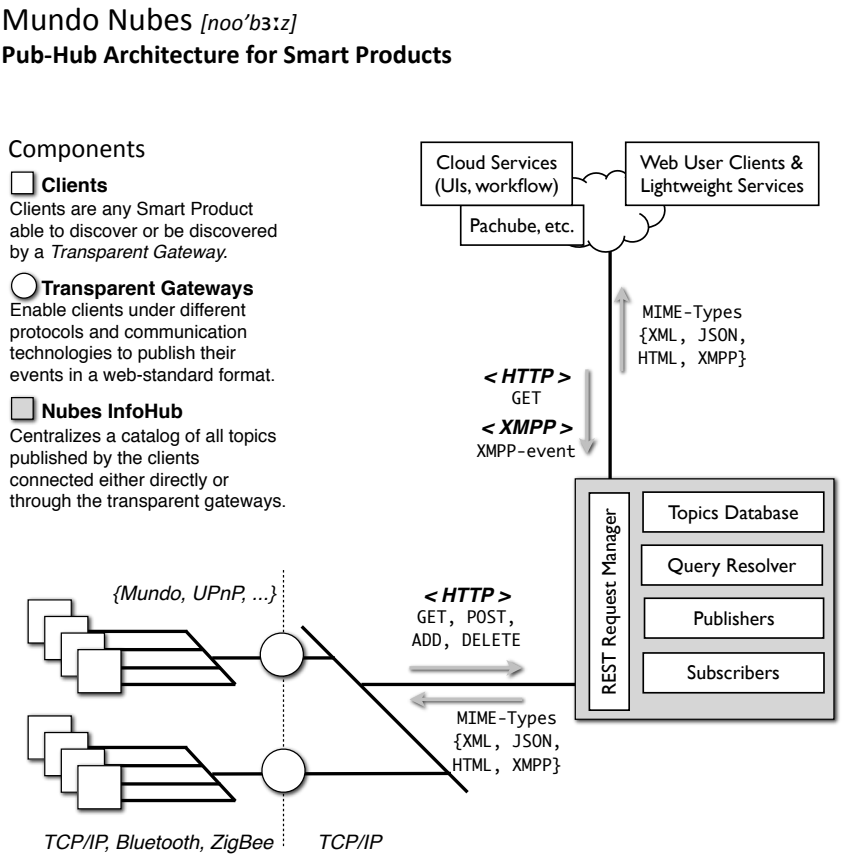


Figure 4.17: Adaptive distribution of the Runtime Environment

4.7 MUNDO NUBES: PUB-HUB FOR SMART PRODUCTS

Chapter 2 discusses that traditional communication approaches between devices rely on custom and vertical solutions. This makes it difficult to integrate products, sensors, and services while posing scalability challenges. The increasing variety of dedicated communication and protocol technologies further complicates information sharing, necessitating bespoke brokering solutions for each application.

In response to these challenges, several innovative communication technologies that use the publish-subscribe metaphor for communication and information sharing among entities have emerged. Pub-Hub architectures [76] are a specific type of publish-subscribe model in which a single component (the Hub) receives all requests from clients that want to publish a particular topic or information. All interested entities on the local network can subscribe to any topic they desire with the Hub. The Hub then notifies all interested parties whenever new information is available from one of its publishers. This configuration enables Smart Products with limited resources, such as Arduino, to share their information with other entities without overburdening their capacity. We introduce **Mundo Nubes** *[noo'bɜ:z]*, a lightweight Pub-Hub architecture for devices and services to share and access

information across different network and protocol technologies. The focus of Nubes is to simplify the acquisition and reuse of data and interoperability from shared devices using standard Web technologies. The advantages of this approach are many:

- Simplifies the integration of information sources. Devices become resources that can be addressed by web developers.
- Enables very resource-constrained Smart Products devices to share their information with all interested parties, without the risk of being overwhelmed by the multiple requests.
- Expands the integration of services that can augment the information provided by the Smart Products. Such services could be available on a cloud infrastructure, locally deployed, or lightweight and transient (e.g., Javascript-based services that live only during the lifespan of a user's webpage browse).

Disadvantages Since the central Hub maintains a connection to each subscriber, performance can degrade significantly as the number of subscribers growsgrows. One mitigation approach is to switch from transparent gateways to aggregating gateways. In this second case, the gateway would decide whether the traffic should be delivered to the Hub or not, therefore minimizing the number of concurrent connections the Hub would have to maintain and the information being delivered.

4.7.1 *Architecture*

There are three kinds of components in the Nubes architecture.

CLIENTS are any Smart Product with communication capabilities, at a minimum able to communicate themselves with a Transparent Gateway, and able to discover or be discovered on the network automatically.

NUBES INFOHUB centralizes a catalogue of all publications or "topics" being published through the Transparent Gateways by the connected Smart Products. It also takes care of subscriptions and notifications to their subscribers. Smart Products able to locate the Nubes InfoHub themselves may connect directly, without the intervention of a Transparent Gateway. The communication with Nubes InfoHub (requests, eventing and subscriptions) is over HTTP (supports the GET, POST, PUT. and DELETE commands), and resources can be retrieved using MIME types to specify the preferred format for receiving the information requested. Finally, the InfoHub advertises its presence in the network using the standard ZeroConf Bonjour under the topic `.nubes.TCP`.

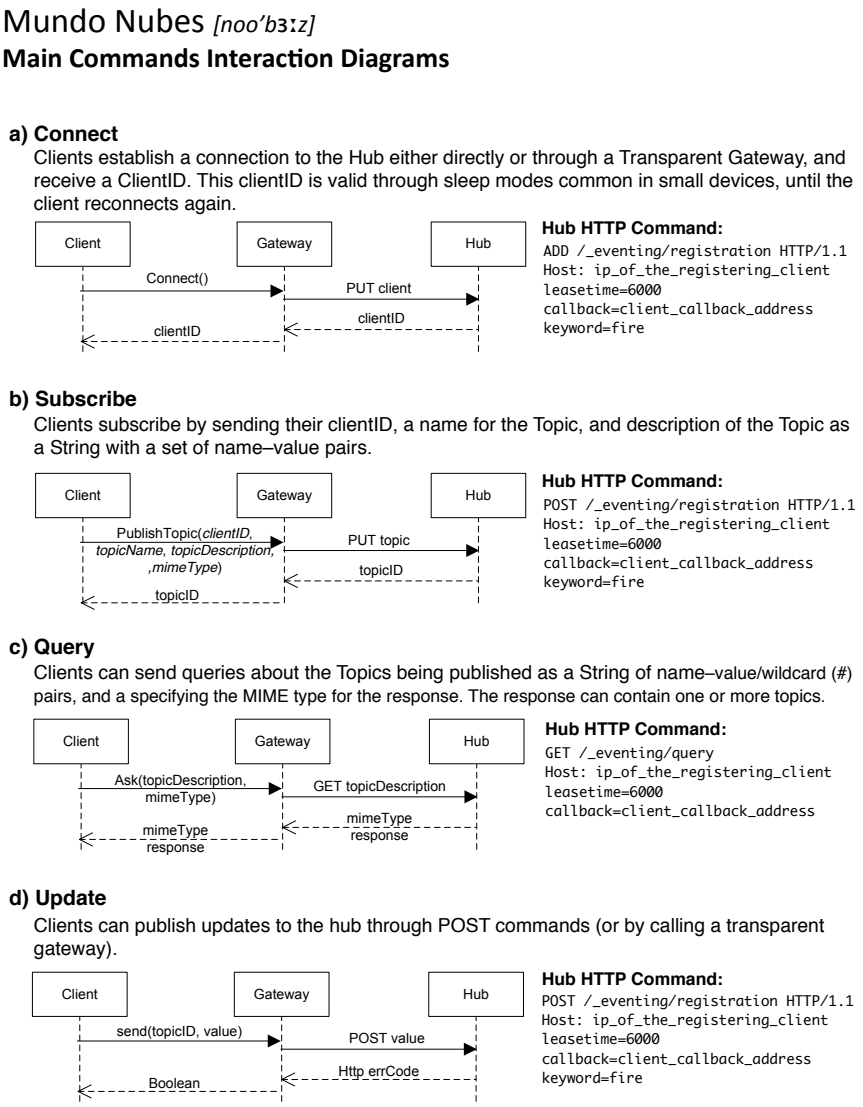


Figure 4.18: Adaptive distribution of the Runtime Environment

TRANSPARENT GATEWAYS Transparent Gateways provide a bridge between different protocols (for instance, devices that support Mundo, UPnP or ZeroConf discovery mechanisms) and communication technologies (Wlan, Zigbee, Bluetooth). Transparent Gateways forward the requests and publications from their peers to the Nubes InfoHub. There can be any number of transparent gateways available in the network. Although not required, Gateways can also be integrated within the Hub component.

4.7.2 *Publications*

Clients register and publish their information following the scheme shown in Figure XXX. First, a client acquires a client by connecting

to a Transparent Gateway or directly to the local Nubes InfoHub. Afterwards, all publications are done under this client.

Every time a client publishes a new kind of data, it provides a name for the publication, determines whether it will be as TEXT of BLOB, and provides a string describing the channel's content using an arbitrary number of name-value tags. These tags are provided to describe the publication semantically. The tags and values are arbitrary, except for a limited number of predefined tag names. Once the publication is submitted, a publication is received. The publication will be required later for eventing and to remove a specific publication from the Nubes InfoHub.

A typical topic publication description string looks like the following:

```
location=kitchen&sensor=temperature
&UUID=23430234293&spname=SmartSaeco
```

4.7.3 *Subscriptions & Queries*

Clients receive messages by creating a subscription. A subscription can be to a specific topic (e.g., current pressure sensor readouts from a smart-shelf) or may use special operators (wildcards) to retrieve all the existing publications existing related to a given topic (e.g., energy consumption). In the first case, a simple module could profile the energy needs of all devices currently connected to the network, while in the second, a client could interpret the pressure sensor readouts and establish the centre mass on the shelf. Subscriptions are specified in a similar manner as publications are done, replacing the values for those specifics to be used as filters or the wildcard # to signify "all".

Typical subscription queries:

`location=kitchen` (All topics related to the Kitchen location)
`location=\#\&sensor=temperature` (All temperature sensors available in all locations)

In Figure 4.18, the interactions between clients, Transparent Gateways and InfoHub for the most common operations are shown.

4.8 U.I. SUPPORT FOR SMART PRODUCTS

As does any other device or appliance, physical Smart Products present a user interface in the form of a small display with buttons and controls to operate the device. In addition, Smart Products provide the ability to define an interface for users browsing the environment with their user-terminal (such as mobiles phones and P.D.A.s).

However, user interfaces evolve over time, and they may need to support a growing number of client devices. Together with the amount

of processing implied, this level of complexity is hard to attain in devices with restricted resources. We present in this chapter a mechanism to enable very small devices to be browsed by multiple user clients concurrently and support different formats, minimizing the amount of computing power required. Furthermore, the proposed approach helps decouple the evolution of the user interface from the device itself, enabling manufacturers to provide an always up-to-date interface for their products.

The following figure present the interaction diagram describing the proposed solution:

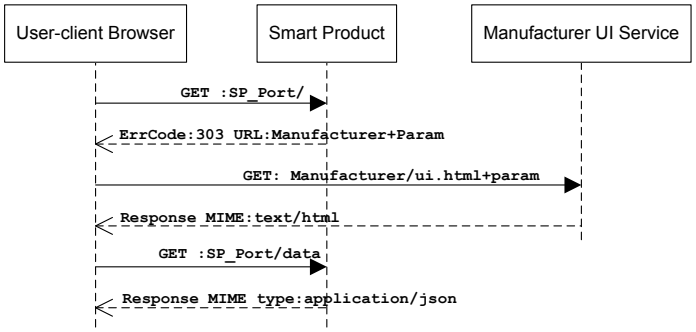


Figure 4.19: User Interface support for Smart Products

4.8.1 *Architecture*

The alternate U.I. Smart Products offer is HTML-based. Therefore, user-client browsers can access the U.I. through HTTP using the port specified by the Smart Product. Instead of serving the HTML-based itself, the Smart Product redirects the request to its manufacturer's service that will determine, according to the user-client type, the right interface to be presented to the user. During the forwarding procedure, the Smart Product also provides its local address as a parameter to the U.I. that the user-client will download. In this way, as soon as the user-client finally downloads the requested U.I., it can connect to the desired device using the network address given. The address is used to request only the required status data from the device to update the information presented by the U.I.

This basic scheme is valid as long as the browsed Smart Product is connected through TCP/IP. In other situations, access to the device's U.I. is done through the topic published by the Smart Product at the local Mundo Nubes Hub. User Clients can find the local Hub through ZeroConf Bonjour under the topic `.nubes.TCP` and browse the available local devices through the provided web interface. The overall solution scheme presented assumes user-clients browsing local physical objects have a network connection with Internet access.

4.9 SUPPORT FOR SEMANTIC DISAMBIGUATION

The concept of Open Smart Environments relies on entities to connect, find and communicate among themselves. In this chapter, we have already covered how Smart Products connect and find themselves, but to communicate and be able to combine their functionality, one last step remains: semantic disambiguation.

Several approaches have been proposed to unify and disambiguate the meaning of terms and product descriptions, mostly based on ontologies. Ontologies do offer several advantages, exposing a set of an arbitrary set of relationships among term objects. However, if O.S.E.s are to become truly open, setting fixed points such as ontologies can already set a limitation to achieve that goal.

However, such term-objects are the ones that carry the heavier semantic load. If we adopt an independent mechanism to establish the cardinal relationships among terms such as is-a, synonyms, hyponyms and hypernyms, the basic needs for intercommunication can be satisfied, leaving the use of ontologies for specific domains.

This section presents a cloud-based service for semantic disambiguation to allow smart products to better interoperate. The Semantic Disambiguation Service (http://insitu-solutions.de:8180/ InSituDictionaryService/) provides a REST interface and relies on Wordnet to provide information about terms and their relationship. The service can answer query responses in XML and JSON formats. The Disambiguation Service provides the following functionality:

All synonyms, hyponyms, and hypernyms through the following URL structure:

Base URL: http://insitu-solutions.de:8180/InSituDictionaryService

Queries: /{synonyms, closestsynonym, hyponyms, hypernyms}/{**type**}/{word}

where **type** can be any of: {n, v, a, r}, where "n" = noun, "v" = verb, "a" = adjective and "r" = adverb.

4.10 SUMMARY

In this chapter, we introduced the Smart Products building block for Open Smart Environments. One key issue identified and resolved is the heterogeneity of devices, communications means and computing resources available. In order to enable interoperability in such diverse scenarios, a new pub-hub architecture has been introduced to enable both interoperability and access to the web. The Smart products model also introduces a set of standard interfaces and document-based descriptions to enable new components from independent vendors to integrate within an environment.

5

HUMAN AND SPACE IN OPEN SMART ENVIRONMENTS

This chapter presents a model for one of the key applications of Open Smart Environments (OSEs): indoor navigation. By combining people and space models, OSEs can guide human activity across vast swaths of Space while customising the experience by sharing information.

5.0.0.1 *Publication*

This chapter is based on the following publications:

- Fernando Lyardet, Jan Grimmer, and Max Mühlhäuser. "CoINS: Context sensitive Indoor Navigation System." In: *Proceedings of IEEE International Symposium on Multimedia (ISM)*. Los Alamitos, CA, USA: IEEE Press, 2006, pp. 30–37
- Fernando Lyardet, Diego Wong Szeto, and Erwin Aitenbichler. "Context-aware indoor navigation." In: *AmI '08: Proceedings of the European Conference on Ambient Intelligence*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 290–307

- Fernando Lyardet, Diego Wong Szeto, and Erwin Aitenbichler. "Context Aware Indoor Navigation." In: *Proceedings of the European Conference on Ambient Intelligence*. LNCS. Springer Verlag, 2008
- Erwin Aitenbichler et al. "Fine-Grained Evaluation of Local Positioning Systems for Specific Target Applications." In: *UIC*. 2009
- Dirk Schnelle, Fernando Lyardet, and Tao Wei. "Audio Navigation Patterns." In: *EuroPLoP European Conference on Patterns and Pattern Languages*. 2005

5.0.0.2 *Contribution Statement*

The *CoINS Indoor Navigation prototype* conceptual design and definition of the World Model was done by me. In contrast, the low-level design was co-developed with two exceptional master students, Jan Grimmer and Diego Wong Szeto, who also implemented the prototype. Prof. Dr. Matthias Dehmer helped with the idea generation and conceptual design of applying the novel general trees for efficient indoor navigation path generation and helped verify the formal proof. Finally, the evaluation was done by me with the advice and support of Dr. Dirk Schnelle, Prof. Dr. Jussi Kangasarju, Dr. Aitenbichler and Diego Wong Szeto.

5.1 INTRODUCTION

Smart environments rely on knowledge about their inhabitants to better anticipate, adapt, and react to their needs. This knowledge is usually described by a *user model*, a representation of traits describing the humans supported by the system. This chapter introduces extensions to the state-of-the-art user models to describe the inhabitants of an environment better and add new ways products can integrate with the people they serve.

Then a model for indoor locations, called *Space*, is described together with context-aware spatial orientation and guidance services. The Space model represents physical spaces as a structured, continuous objects in which human activity occurs. The orientation and guidance services called *CoINS* (Context Indoor Navigation System) introduce functionality based on location information of physical objects and people, combined with the knowledge of the space topology.

We show in this chapter how the Space model presented can efficiently support the whereabouts of people and machines. We will also show a unique contribution of CoINS to other existing indoor orientation services: the ability to provide orientation and guidance, requiring no further location infrastructure other than sticking simple 2D barcodes called *qrCodes*, as opposed to other costly indoor sensing technologies. The system itself runs on a cloud-based platform called Google App Engine. Therefore CoINS services are always available and do not require local servers to execute. This combination of cloud-based services and minimal on-location deployment requirements

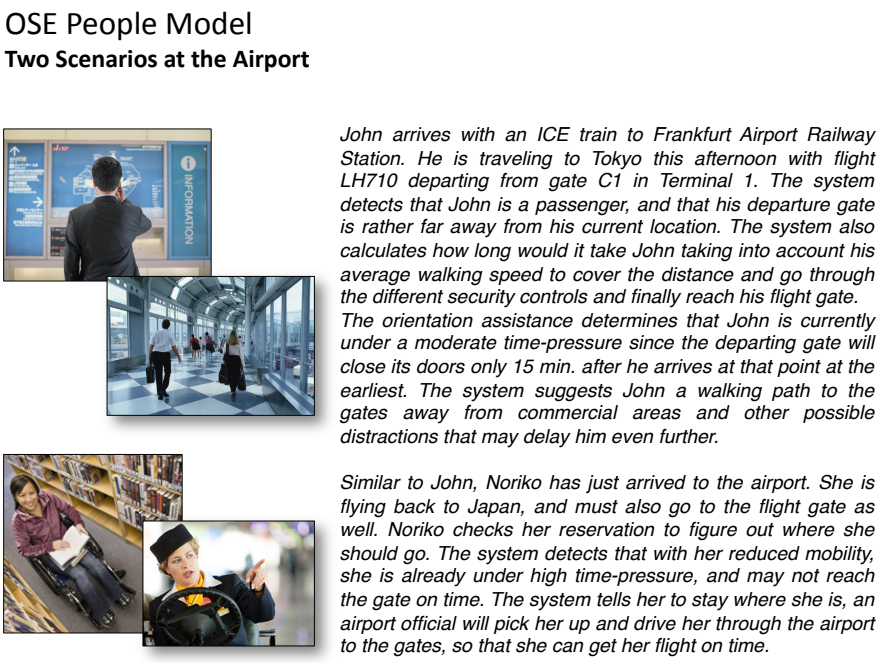


Figure 5.1: Common scenarios of smart environment assistance on public spaces. The knowledge about people is essential to solve the depicted situations.

(QrCode stickers) will foster pervasive indoor guidance assistance availability.

5.2 A MODEL FOR PEOPLE

The people model assumes that people carry a personal, trusted device like a mobile phone, that we will call *"ME"* [17], which carries the user-related data and can communicate the user information to services and products available in the surroundings. Although beyond the scope of this thesis, "ME" devices could also include agents that, as they learn from the actual user behaviour, could adjust the preferences data over time. Figure 5.1 describes two scenarios where information about people is essential. The People model presented in this section enables the realisation of such scenarios.

The OSE people model is divided into five sections, allowing anonymous data (user information) to accommodate unforeseen information and semantics and structured information with specific ontological bindings (user data, physical capabilities, preferences, skills, actions).

5.2.1 *User Information*

We take advantage of UserML [98] and the Gumo2.0 ontology [119] to model user information. Each `UserInfo` element comprises category, range, and value subelements. On a second level, the ontology defining

the categories is presented. This approach is advantageous in that different ontologies can be used with the same `UserML` tools. Thus, different user-modelling applications could use the same framework and keep their user model elements.

The UserML XML description consists of an unbounded list of UserData entries. Each one defines the category, the range and the value. A reference to the ontology stored at `http://gumo.org/2.0/` can either be placed in the metadata module, where it will serve as a default value for all `UserData` entries, or in the `UserInfo` entry itself[98] .

Figure 5.2: XML encoding structure and file organization of OSE people model structure

5.2.2 *Physical Capabilities Restrictions*

Physical capability restrictions are represented using the International Classification of Functioning, Disability and Health (ICF) of the World Health Organization (WHO) [161]. The WHO has developed the ICF to measure health and disability. The ICF provides a standardised description of health and health-related states.

The classification is structured around several domains: body functions, body structure, individual activities, and participation. Body function refers to all physiological functions of body systems such as mental, sensory, voice, cardiovascular, digestive, and so forth. Body structures include anatomical parts of the human body such as organs, muscles, bones, and tissues. Activities refer to the execution of any action by an individual and participation refers to the involvement in a life situation.

To assess the level of health or disability, the ICF proposes different qualifiers depending on the kind of impairment evaluation. An application example is the CoINS indoor orientation and guidance services that take advantage of the relevant branches of ICF classification related to human mobility (Code S, chapter 90).

5.2.3 *User Preferences*

A preference describes the inclination of a person regarding a specific topic. In the OSE model, preferences are modeled using multi-attribute utility theory (MAUT) [71]. MAUT is a powerful technique that allows decision-making using an arbitrary number of preferences while minimizing the risks of inconsistencies other rule-based approaches have.

In the model we are presenting, each person has a different set of preferences, and each preference may have a different priority. Preferences are modelled by an assessment function f that returns the "strength" of a preference. Every user has different preferences when faced in several different situations. As a result, an assessment function f holds only if certain conditions apply. Thus, the set E is defined as the set of possible events, e. An event e represents a physical situation, such as a user carrying heavy luggage or the location that is on fire.

5.2.4 *Competencies*

In scenarios where critical tasks are involved, smart environments can better adapt and assist people if a person's capabilities are known, or if there is evidence of having successfully performed a similar action before. Such kind of information is broadly known as competencies.

F	=	{f | f is an assessment function}
E	=	{e | e is an event}
X	=	{x | x can be assessed with some $f \in F$ }
where		$f : \mathcal{P}(E) \times X \rightarrow \{0, \ldots, 100\}$
and :		
E	=	(SUBJ, COND)
TIME	=	{starttime, endtime}
SUBJ	=	{user 1, user 2, room a036, ...}
COND	=	{STATE, ACTIVITY, LOCATION, TIME}
STATE	=	{ happy, busy, available, ...}
ACTIVITY	=	{ run, read, work, ...}
LOCATION	=	{room a036, kitchen, ...}

Table 5.1: Multiattribute Utility Theory definitions, and attribute examples

The concept of human competencies under the OSE people model refers to any aspect of competence, skill, attitude, ability, or learning objective that a given person holds.

The OSE people model takes advantage of the well-stablished SCORM standard called HR-XML[53] to describe a person's qualifications, bind them to task definitions and connect The qualifications to existing learning resources.

5.3 A MODEL FOR SPACE

Location information is a key parameter governing the behaviour of context-aware applications, such as indoor guidance to people in spaces unknown to them, composing the behaviour of home appliances or guiding robots and machines in large logistic settings.

Support for Space is essential for Smart Environments. It opens a breath of services and assistance possibilities to satisfy the basic human need of orientation. Orientation and wayfinding are among the most common situations in unknown premises or large public spaces such as airports, stadiums and government buildings. In this thesis, both the Space Model and the wayfinding service called CoINS (Context-aware Indoor Navigation System) 5.4 have been developed together, helping the Space model grow and improve following the requirements for wayfinding.

Being aware of the location of its inhabitants and products, smart environments can also assist even those already familiar with the particular premises. Providing the best possible route over large locations can be particularly useful, especially if several specific points must be visited. In this case, the environment can suggest the most convenient path to follow (where the minimum total distance to cover at the end of the day could be one criterion to fulfil).

Security and safety are two aspects where spatial assistance can save lives. Diverting people from dangerous premises, or providing the best alternative in case of emergency is also a significant contribution that smart environments can offer through their spatial knowledge and services.

5.3.1 *Requirements for a Location Model*

The requirements for a location model can be grouped into four categories: entity position, precision, topological relationships and scope.

Entities can be anything from an object to a person. In this location model, the position of any entity is a coordinate (geometric or symbolic) to a reference system. This is a basic requirement for indoor navigation since relevant positions like the user or target locations must be specified. Entities can be locations as well. In this work, locations are floors, corridors and rooms. A distinction between composite and single locations is made, i.e., a composite location contains at least one location. For instance, floors contain rooms and corridors.

Precision refers to the granularity of a position representation. In this case, it concerns the precision of the reference system, i.e., the tiniest area that can be represented. Although precise human navigation is not always necessary, a very precise location model allows distance and bounding box calculations. For example, these calculations are useful in scenarios with many obstacles, and the distance between them has to be estimated to determine if a subject can pass through.

Desired topological relationships in this model include spatial containment and spatial neighbourhood relationship. Spatial containment refers to the ability to determine whether a location is contained within another location. Spatial containment is helpful for indoor navigation if the user wants to navigate to a specific location without knowing its position, but only the location's name. The navigation software could find the position of the target location by doing a recursive containment query. A spatial neighbourhood relationship between locations is also helpful to calculate paths throughout locations. Two locations are neighbours if they both are directly accessible from one to the other.

The scope of this model includes a single building. The level of detail of this scope covers locations such as corridors and rooms. Nevertheless, the model should remain extensible for greater scope, like a complex of buildings.

5.3.2 *World Model*

The World Model serves two main purposes. First, it supports transformations between geometric coordinates and symbolic locations,

and vice versa. When a 3D tracking system is used that provides geometric coordinates to locate users, the model must be able to transform this coordinate into a symbolic location, such as a room number. CoiNS' pathfinding algorithm starts by using symbolic models to create a coarse plan of the route. After that, the geometric models are used for fine-tuning. Therefore, the world model must also support transforming symbolic into geometric coordinates.

Table 5.2 summarizes the advantages and disadvantages of the geometric, symbolic and hybrid models. The advantages and disadvantages of each model are summarized in table 5.2. This subsection discusses the choice of a location model for CoINS based on their characteristics.

Table 5.2: Summary of the reviewed location models

Model	Geometric coordinates	Symbolic names	Precision	Intuitive semantic
Geometric	yes	no	high	low
Symbolic	no	yes	low	high
Hybrid	yes	yes	high	high

The combined advantages of geometric and symbolic models fulfil the requirements above. On the one hand, the geometric part of the hybrid model addresses precision and geometric coordinate requirements. On the other hand, the symbolic part provides symbolic names and a base for topological relationships.

The model contains three levels of detail. On the top level, floors are modelled as a graph. Every floor is a node of the graph with a symbolic name ("Floor 1", "Floor 2", "Floor n"). Floors at the top level are connected if and only if they are connected directly with stairs or elevators. Every floor has a symbolic graph representation of corridors and rooms it contains. In this case, corridors and rooms are the nodes in the graph. Two nodes are connected if there exist entrance or exit points (e.g. doors) between them. Just like every floor, rooms and corridors have symbolic names.

Each location has a graph-based representation of its geometry. This representation can be found at the bottom level of this location model and consists of a set of rectangular nodes covering the area of the room or corridor. A quadtree is used to manage these nodes.

By using graph-based representations, the location model can take advantage from graph-traversing algorithms to perform path-searching tasks [KH06]. For instance, a path can be found by traversing the nodes of the location graph.

Single locations are convex in this model. The reason for this relies on the ease-of-distance calculation. The distance between two points in the same location can be easily computed by calculating the length of the line segment that contains both ends. Sometimes, it is desirable to decompose a very large single location, e.g. big halls or very large corridors, into smaller spaces. Even though this is not always necessary, it can be useful if:

- The geometry can be simplified in terms of distance or form. For example, a big fair hall could be divided into several areas for different expositions, each with its own geometric boundaries.
- A more meaningful symbolic representation can be found. For example, a description of a single location could be divided into several descriptions.

The world model also enables efficient pathfinding. The design of the CoINS world the model has been refined over several iterations to ensure that the search sets are as small as possible and that the basic relations needed by the pathfinding algorithm can be checked efficiently. In most cases, users move in two dimensions. Movements in the height dimension only occur when changing floors, which is modelled by using separate maps for each floor.

The CoINS world model is a hybrid model that combines symbolic graph-based models with geometric models. A symbolic model is required for indoor navigation, because *room numbers*, *corridor names*, *floor numbers*, etc. have an intuitive semantic to users. Using geometric coordinates for end-user interaction would not be suitable. The geometric model is needed to determine the shortest paths and obtain orientation information for guiding users into the correct directions.

The world model serves two main purposes. First, it supports transformations between geometric coordinates and symbolic locations and vice versa. When a 3D tracking system is used that provides geometric coordinates to locate users; the model must be able to transform this coordinate into a symbolic location, such as a room number. The pathfinding algorithm of CoINS starts with the symbolic models to create a coarse plan of the route. After that, the geometric models are used for fine-planning. At the transition between using symbolic and geometric models, the world model must also support transforming symbolic into geometric coordinates.

Second, the model enables efficient pathfinding. The design of the CoINS world model has been refined over several iterations to ensure that the search sets are as small as possible and that the basic relations needed by the pathfinding algorithm can be checked efficiently. In most cases, users will mostly move in two dimensions. Movements in the height dimension usually only occur when changing floors, which is modelled by using separate maps for each floor.

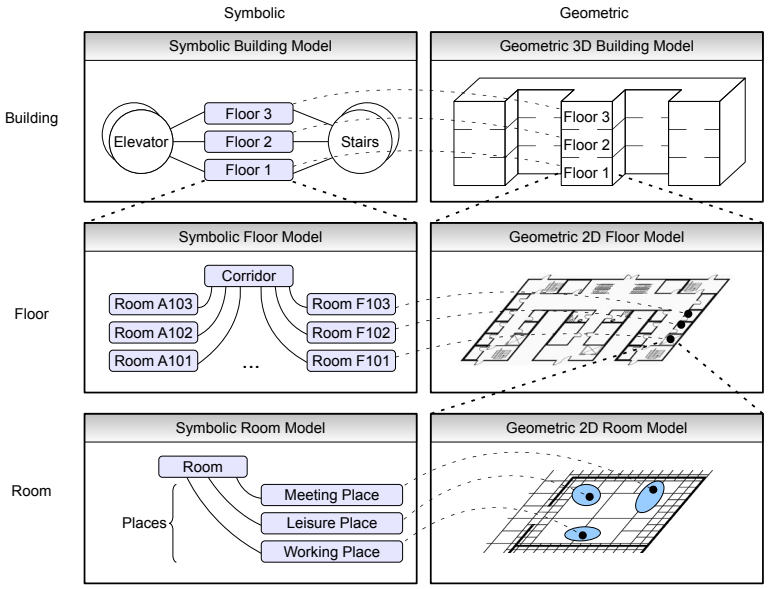


Figure 5.3: Hybrid world model of CoINS.

The world model of CoINS is shown in Figure 5.3. According to modelling level and model type, the world model consists of five parts:

Geometric Building Model: A detailed geometric 3D model of the entire building. All other model parts are created at design time from this model. At runtime, this model is not needed by the pathfinding algorithm of CoINS. However, if a 3D local positioning system is used in the building, the model must transform metric coordinates to symbolic locations.

Geometric Floor Model: A geometric 2D model of a single floor. Here, the term floor denotes parts of the building that are connected and are at the same level. It does not necessarily have to correspond to a whole story of a building. On the same floor, users will only be able to move in two dimensions. When guiding a user on a floor, a 2D representation is sufficient. The reduction to 2D Space simplifies path calculation and increases performance significantly.

Geometric Room Model: The room model is a geometric 3D model of a room or a corridor. While it is not necessary to model rooms in detail, the geometry of corridors is important for finding shortest paths and to obtain direction information for navigation.

Symbolic Building Model: This is a symbolic graph-based model of the whole building. Its nodes are *floors* and *connecting structures*, such as elevators and stairs. Each floor node holds a reference to its symbolic floor model.

Symbolic Floor Model: This is a symbolic graph-based model of a floor. Its nodes are *rooms* and *corridors*, connected by entrances and exits.

5.3.3 *Analysis and Evaluation of the Space Model*

This new model can be formalized using a graph class called universal hierarchical graphs [73]. Universal Graphs are undirected and unweighted graphs $\mathcal{G} =< \mathcal{V}, \mathcal{E}psilon >$ with vertex set $\mathcal{V}$, edge set $\mathcal{E}$ and $N = |V|$ nodes. We denote the graph class *universal hierarchical graphs* (UHG) as $\mathcal{G}_{UH}$.

In the work by Emmerich [73], undirected universal hierarchical graphs have been introduced to calculate the topological entropy of such graphs. The graph class was then extended to the directed case by the work of Dehmer[63]. In these works, it was shown that these graphs can be efficiently classified by transforming them into certain property strings capturing important structural information.

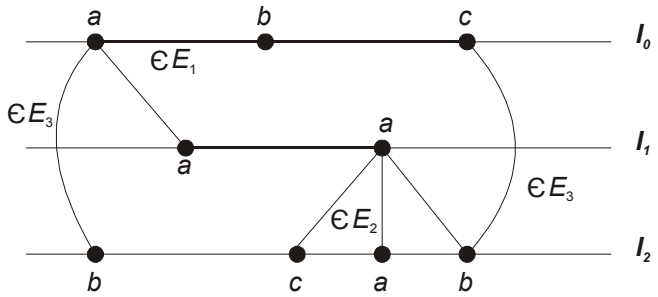


Figure 5.4: A vertex-labeled universal hierarchical graph $\mathcal{G}_{UH}$.

In the following, we extend the definition given in [73] to vertex-labelled graphs and state a special edge type specification.

Let

$$V := \{v_{0,1}, v_{0,2}, \ldots, v_{0,|V_0|}, v_{1,1}, v_{1,2}, \ldots, v_{1,|V_1|}, \\ v_{2,1}, v_{2,2}, \ldots, v_{2,|V_2|}, \ldots, v_{|L|-1,1}, v_{|L|-1,2}, \ldots, v_{|L|-1,|V_{|L|-1}|}\}, \quad (5.1)$$

be the vertex set, $|V| < \infty$ and let

$$A_V^U := \{l_1, l_2, \ldots, l_{|A_V^U|}\}, \quad (5.2)$$

be the unique vertex alphabet. $l_V : V \longrightarrow A_V^U$ represent the corresponding vertex labeling function. $L := \{l_0, l_1, \ldots, l_{|L|-1}\}$ defines the level set. $|L|$ denotes its cardinality. $\mathcal{L} : V \longrightarrow L$ is a surjective mapping that is called a multi level function if it assigns to each vertex an element of the level set L. The graph $U = (V, E)$ O MEJOR: $U = (V, E, L)$ is called a vertex-labelled universal hierarchical graph $\Leftrightarrow$ its edge set can be represented by the union $E := E_1 \cup E_2 \cup E_3$. We specify the sets E_i as follows:

- E_1 denotes the set of horizontal Across-edges. A horizontal Across-edge does not change a level i.
- E_2 denotes the set of edges which change exactly one level. .

- E_3 denotes the set of edges which overjump at least one level.

The set of undirected labelled universal hierarchical graphs is denoted by $\mathcal{G}_{LUHG}$.
As an example, Figure (5.4) shows an undirected labelled universal hierarchical graph. Here, it holds $A_V^U := \{a, b, c\}$. For example, it is $l_V(v_{0,1}) = a$ and $l_V(v_{2,4}) = b$. We see that our objects we deal with can be described by graphs $U \in \mathcal{G}_{LUHG}$. The relationship to general graphs can be understood by recalling that an arbitrary graph is not necessarily hierarchical (the hierarchies must be induced). Generally, hierarchies can be induced by selecting a distinct root node and applying algorithms based on shortest path, see, e.g., [67]. In our case, the hierarchy is given naturally because the building we want to model is hierarchical.

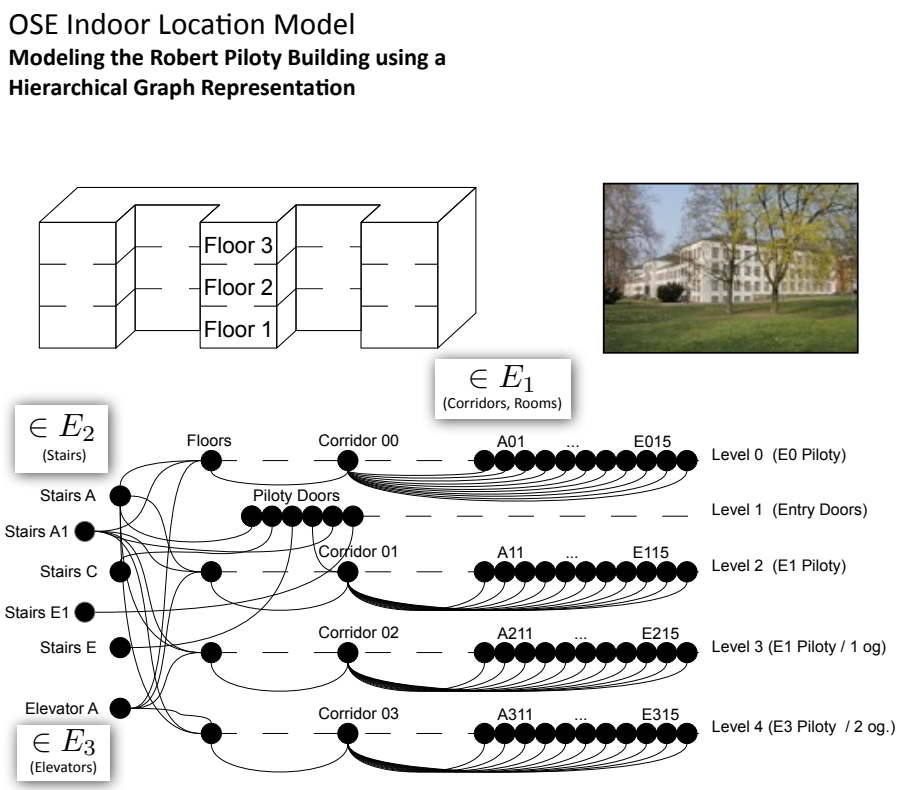


Figure 5.5: A concrete example of a building structure modeled with a universal hierarchical graph $\mathcal{G}_{UH}$.

In the following, we consider a special class $\mathcal{G}^{\star}_{LUHG}$ of undirected labelled universal hierarchical graphs such that E_2 can be written as

$$E_2 = \{\{v_{0,i_0}, v_{1,i_1}\}, \{v_{1,i_1}, v_{2,i_2}\}, \ldots, \{v_{|L|-2,i_{|L|-2}}, v_{|L|-1,i_{|L|-1}}\}\} \cup E_2^{\star}, \tag{5.3}$$

where $1 \leqslant i_0 \leqslant |V_0|$, $1 \leqslant i_1 \leqslant |V_1|, \ldots, 1 \leqslant i_{|L|-1} \leqslant |V_{|L|} - 1|$. Speaking informally, that means we assume that there always exists at least one directed path starting from at a vertex $v_{0,i_0} \in V$ and ending at $v_{|L|-1,i_{|L|-1}} \in V$ (e.g. as it is the case with stairs connecting floors).

$E_2^\star$ is assumed to contain the remaining edges $e \in E_2$. By using this assumption, we easily derive the following assertion.

Theorem 1. *Let* $U = (V, E) \in \mathcal{G}^\star_{LUHG}$. *The time complexity for finding an arbitrary vertex* $v \in V$ *is* $O(|L| - 1) + O(|V_i|)$.

Proof. Let $U = (V, E) \in \mathcal{G}^\star_{LUHG}$ and arbitrary $v \in V$. Because there exists a path $P = (v_{0,i_0}, v_{1,i_1}, \ldots, v_{|L|-1,i_{|L|-1}})$, the worst case time complexity to calculate the distance for vertices on this path is consequently $O(|L| - 1)$. The time complexity for finding a vertex on level i is $O(|V_i|)$. □

Corollary 1. *Let* $U = (V, E) \in \mathcal{G}^\star_{LUHG}$. *In case there exists a vertex on each level* i *which connects all remaining vertices on this level, the time complexity for finding an arbitrary vertex* $v \in V$ *is* $O(|L| - 1) + O(1) == O(|L|)$

Theorem 2. *Let* $U = (V, E) \in \mathcal{G}^\star_{LUHG}$. *Without the assumptions we made in Theorem (5.3.3), the time complexity for finding an arbitrary vertex* $v \in V$ *(starting from another vertex) is* $O(|V|^2)$.

Proof. Let $U = (V, E) \in \mathcal{G}^\star_{LUHG}$. We always assume that our graphs we deal with are connected. To start from an arbitrary vertex and calculating the shortest distance to another vertex, any shortest path algorithm can be used, e.g., Dijkstra-algorithm [67]. For this, the complete adjacency matrix must be parsed. But this requires time complexity $O(|V|^2)$. □

5.3.4 *World Model Description*

The CoINS System has been designed to be embeddable, extensible, and to use standard interfaces. To provide guidance between two locations, only 2D representations are required, although the software is able to convert 3D representations into 2D as well.

5.3.4.1 *Describing the World as Quake III maps*

To provide guidance between two locations, only 2D representations are required. However, the software is able to convert 3D representations into 2D as well.

Technique is called portals and sectors o Portal rendering

CoINS makes use of the Quake III format [169, 218] for the world descriptions, since it is widely known and open source tools are available to create world descriptions. Quake III maps basically consist of two kinds of components, brushes and entities. Brushes are arbitrary convex shapes. Walls, floors, and ceilings are frequently modeled using brushes. Textures can be put on brushes to describe the looks of the object in the game. For instance, textures exist to make a brush look

like a stone wall or a metal ceiling. However, these are only relevant for use in a game and are not needed for CoINS. The world thus consists of brushes (convex objects) and each side of a brush is indicated by a triangle. These points can be arbitrary points on the infinite extension of the surface area generated by the intersection of these planes. The geometry is calculated then in three steps:

1. The infinite plane described by each triangle is calculated and represented in its coordinate form which consists of a vector $\vec{n}$ which is the normal of the plane (that is, "standing" at right angle on it) and a constant d such that $\vec{n} * \vec{p} + d = 0$ for each point p on the plane.

2. The intersection of each pair of planes is calculated. If such an intersection exists, it is described by a straight line as a pair $(\vec{o}, \vec{d})$ of vectors such that $\vec{o} + r * \vec{d}$ is a point on the line for each $r \in \mathbf{R}$.

3. The intersections between these lines and the planes are calculated, resulting in eight points which are the corners of the brush.

It would have been possible to use the compiled BSP files instead. Alas, restoring the original geometry from this information would be computationally expensive and would not offer any additional features. We have used the GtkRadiant modeling kit [171] to create world models for CoINS. However, the maps created are not automatically interchangeable between the game and CoINS: entities are not solid in Quake by default, and measuring units are not the same. One unit equals one cm in CoINS but one inch in Quake. This means that the map should be scaled by a factor of $\frac{1}{2.54}$.

For CoINS, an additional entity `wm_door` was added. As the name implies, it is used to identify doors or, more generally, openings in walls through which the user can go from one room to another. This is needed for identifying separate rooms and building the search graph for path calculation. The process is explained in the following section.

5.3.4.2 *Mapping a 3D model into 2D*

The current approach implemented in CoINS is as follows:

1. A plane is created which is parallel to the ground. The z-value (that is, the height of the plane) is currently at 160 cm, which is approximately the height of the eyes of an average person.

2. The intersections between this plane and every "element" of the world (walls, doors, objects) is calculated, resulting in a set of polygons each of which is described by a list of points.

3. These polygons still have three dimensions. However, since the plane was created at constant height, all polygons, or rather all

points describing the polygons, have the same height value. By ignoring this constant z-value, these three-dimensional polygons are transformed into 2D-space.

4. The points of each 2D polygon are sorted clockwise by calculating their convex hull which is required to do any further calculations and draw the polygons in the GUI.

5.3.4.3 *Data Structures for Room Detection and Path Creation*

3.4.1. THE CONVEX HULL. A convex hull [104] for a set of coordinates or points is the minimal polygon containing all these points.

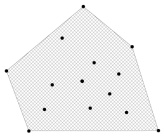


Figure 5.6: A convex hull

It is calculated with the Jarvis March algorithm [108]:

1. The algorithm starts by selecting a point p_0 which is guaranteed to be on the hull. CoINS does this by selecting the point with the minimal x coordinate. If several points with that x exist, the one with minimal y is selected from these.

2. Then, it creates straight lines between the selected point p_0 and each other point in the set. The point p_1 with minimal angle from the y-axis to the connecting line $p_0 \rightarrow p_1$ is the next point on the hull.

3. Analog to step 2, the point p_{i+1} is selected as the point with the minimal angle between the line $p_{i-1} \rightarrow p_i$ and $p_i \rightarrow p_{i+1}$.

4. Step 3 is repeated until p_0 would become selected again as a new point on the hull.

3.4.2. QUADTREE. The two-dimensional data generated as described in the previous section consists of an unordered set of polygons. Areas that are within the bounds of the map but not covered by a polygon are implicitly considered empty. However, for further calculations this representation is insufficient. To calculate paths and determine which coordinates belong are in the same room, it is necessary to create some kind of raster data. A possible solution involves the use of a two-dimensional array of data representing the respective terrain type. Each cell of the grid would be represented as one element of the array. However, the question of optimal resolution arises. Recording every single pixel (which is an area of 1×1 centimeter in our case) results in a clear oversampling of the space, resulting in enormous

calculation time when calculating areas and paths. On the other hand, an oversized grid is too imprecise and undesirable as well. Usually, in large open spaces a larger grid should be used than in smaller corridors.

Quadtrees were invented for computer graphics applications, where they are used to efficiently store image data [225]. Luckily, the terrain terrain representation problem is nearly identical. Furthermore, calculations like efficient neighbor-finding (described later) can be done on quadtrees with relatively little effort. Therefore, the unordered set of polygons which resulted from the transformation process in the previous section are now converted into a quadtree structure.

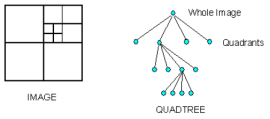


Figure 5.7: Quadtree example

Quadtrees are trees in which each node can have up to four children, and they are built in the following way:

1. The image is separated into four quadrants (called *NW*, *NE*, *SW*, and *SE* here).
2. If a quadrant is not "homogeneous", that is, all pixels have the same color, it is split further.
3. If an area is split, the sub-areas are added as children to the node representing the larger area.
4. The splitting process stops when all subregions are homogeneous or a specified resolution or tree height has been reached.

Through the process described, quadtrees offer a variable resolution down to pixel resolution which can be changed at runtime by expanding further nodes.

3.4.3. NEIGHBOR FINDING. For path calculations and room detection, one must know which areas neighbor one another. Unfortunately, adjacency in space does not relate directly to a simple relation of the node positions within the quadtree. It is possible however, to find the respective neighbors by means of tree search. One commonly used algorithm was proposed by Samet [184].

5.3.4.4 *Room Detection*

For context-sensitive navigation, it may sometimes be necessary to identify which areas (that is, grid cells) belong to one room. For instance, a command like "Go to the kitchen" might be useful. In this

case, new activity descriptions may be triggered at certain waypoints among the path. By entering a cell that belongs to the kitchen, the software knows the target area has been achieved. Determining the boundaries of an area seems simple at first glance, but it is not for a machine, especially if doors are simply modeled as notches in the walls as the context server originally did. Thus, it is not always unambiguous to which room an area belongs to. To be more precise, it is not easy to determine if two directly adjacent cells also belong to the same room. Figure 5.8 shows the situation.

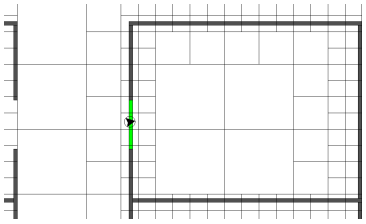


Figure 5.8: Ambiguity problem and solution approach

Two questions arise. First, one cannot always simply determine which room the user is in. Second, even if the cells that are ambiguous for a human are ignored and only the cells that could easily be assigned to a room by a human eye, this is not as easy for a computer. An additional separator is needed to determine the exact boundaries of a room.

5.3.4.5 *Generation of the World Model*

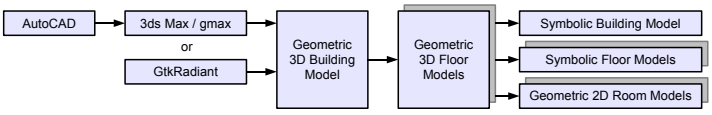


Figure 5.9: Generation of the World Model.

Figure 5.9 shows the generation of the CoINS world model and its parts.

CoINS currently uses the Quake III map format [169] as input for the world descriptions, since it is widely known and open source tools are available to create world descriptions. To create models for CoINS we have used the GtkRadiant kit [171] and Autodesk Gmax [32] which allows us to import AutoCad drawings. A detailed discussion of describing the world using Quake maps, and the further processing required to map into 2D maps and identify rooms is presented in [135].

Quake maps basically consist of two kinds of components: *brushes* and *entities*. Brushes are arbitrary convex shapes. Especially walls, floors, and ceilings are modeled that way. Entities consist of a set

of brushes and can be annotated with arbitrary metadata. CoINS introduces a number of special entities that must be modeled by the user.

- `wm_room` must be added to the floor of each room and annotated with the symbolic name of the room.
- `wm_door` identifies doors or openings in walls through which the user can go from one room to another. This is needed for obtaining the *connected-to* relationships for the symbolic graph-based location model.
- `wm_stairs` identifies stairs connecting multiple floors.
- `wm_elevator` identifies an elevator connecting multiple floors.

After the user has modeled these additional entities, they can export the model as Quake map. The transformation from the *Geometric 3D Building Model* to the *Geometric 3D Floor Models* is currently a manual step. This means that the user has to export each floor as a separate map. The rest of the model building process is automated: the *Symbolic Floor Models*, *Symbolic Building Model*, and the *Geometric 2D Floor Models* are generated from the *Geometric 3D Floor Models*.

The *Symbolic Floor Model* contains all rooms modeled in the *Geometric Floor Model*. If a door entity geometrically touches two room entities, then an edge between the two rooms is added to the symbolic model.

The *Symbolic Building Model* contains a node for each generated *Symbolic Floor Model*. If a stair or elevator entity is found in the geometric model, then an edge is added between two floors in the symbolic model. Identical stairs and elevators can be identified by using the same name attributes.

The *Geometric Room Models* are generated by reducing the 3D model to 2D and then constructing a Quadtree describing the open space.

5.3.4.6 *Path Calculation*

We realized a comparison to determine the optimal search algorithms for CoINS, selecting four Dijkstra variants. These algorithms have been tested and compared using a sample structure of 3328 nodes and 1000 arcs.

Algorithm	Average time (ms.)
Early Termination (E.T.)	968.146
Bidirectional	1 110.290
Goal-Oriented with E.T.	747.240
Bidirectional, Goal-Oriented	804.118

Table 5.3: Comparison of Optimizations.

The building housing the Computer Science Department at the University of Darmstadt was selected as a test case. At a grid size of 80 x 80 cm, 1000 random pairs of nodes (source, destination) were created (only node pairs where the destination could be reached from the source were allowed). Next, the path from source to destination was calculated using Dijkstra with early termination, bidirectional Dijkstra, goal-oriented Dijkstra with early termination, and bidirectional goal-oriented Dijkstra algorithms on a notebook PC (Intel Celeron 1300 MHz). The average calculation times are displayed in Table 5.3. The simple Dijkstra algorithm without early termination and the goal-oriented Dijkstra without early termination were not included since they would have most likely been slower than the early termination variant. As seen in Table 5.3, the bidirectional variant of the algorithm does not outperform its single-directional counterparts. Even if the complexity were theoretically lower, the overhead for double management of visited nodes, predecessor tables, and additional calculation of the abort condition (both algorithms have finally marked one node) after each step seems to outweigh the advantage of lower complexity.

5.3.4.7 *Path Simplification*

Figure 5.10 shows the result of a path calculation from a starting point (the point next to the arrow icon representing the user) and an end point. Each blue circle represents a waypoint along the path where the user would receive instructions.

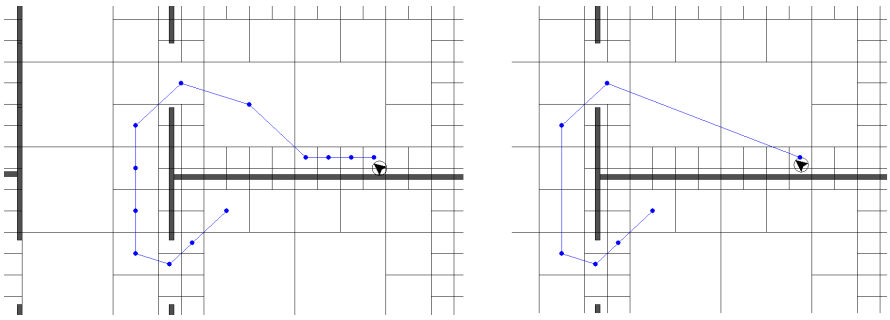


Figure 5.10: On the left, a sample path before simplification is shown. The result of the path simplification algorithm is shown on the right.

The above representation is suboptimal for three reasons:

- Even when following a straight line, the number of waypoints at which the user receives instructions is too high. This would result in a number of unnecessary "walk forward" instructions, irritating the user. However, this problem could easily be solved by only giving instructions when the direction the user should change direction.
- Since the algorithm uses the centers of the grid cells as waypoints and only allows connections between adjacent cells, the path is

not totally optimal in terms of length as it would be if each pixel of the map were a waypoint candidate.

- For the same reason, there are a number of unnecessary waypoints where the user is given the instruction to turn. In the example above, users would expect to be guided directly to the door in a straight line instead of having to perform several turns.

For the reasons stated above, a simplification of the path becomes necessary. The first idea is to remove the waypoints where no change of direction is necessary. This is a simple process which would alleviate the first problem. However, the problems of optimal paths and unnecessary turns would remain. Therefore, another method was chosen. Figure 5.11 shows the result of the simplification process. The number of relevant waypoints has been reduced, resulting in straight connections even between cells that are not directly adjacent. Turns only become necessary when needed. The algorithm uses the quadtree structure to remove unnecessary steps and thus optimize the path.

The algorithm takes advantage of the fact that, if a direct line of sight between two points, even not belonging to directly adjacent nodes, can be established, then the user can walk from one point to the other without the need of intermediate waypoints. Thus, the superfluous waypoint can be removed from the path (the algorithm for calculating if a line of sight is not interrupted by any cell is explained below). Iteratively checking the next node if a line of sight can be established is of unnecessary complexity, especially if a large number of small cells are in a straight line.

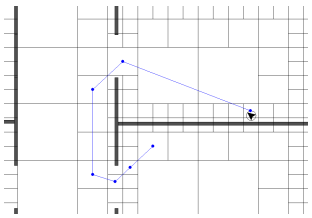


Figure 5.11: Path after simplification

Therefore, the next "relevant" waypoint is determined by using nested intervals. Two values `upper bound` and `lower bound` are used.

1. The lower bound is set to the node directly following the start node in the path (which is always directly reachable).
2. The upper bound is set to the last node of the path, the destination node.
3. If a direct line can be created between the two points (that is, all `Terrain` objects of the intersected nodes return `true` for `isPassable()`), the calculation is finished.

4. If a direct line can not be created, a `candidate` in the middle of the path at $\frac{\texttt{path.length}}{2}$ is tried.

5. If a direct connection to the `candidate` can be established, the `lower bound` is set to the index of the `candidate`, otherwise the `upper bound` is set to the index of the `candidate`.

6. If `upper bound` and `lower bound` are not equal, the algorithms continues at step 5.

7. When `upper bound` and `lower bound` are equal, the algorithm is finished. All nodes between the last relevant waypoint and the `candidate` are removed and the algorithm continues with the old `candidate` as new starting point.

5.3.4.8 *Line of sight calculation*

Line of sight calculation, that is, determination if any obstacle is on the direct connection between two points, is not totally trivial. Of course, it would be possible to check if the line connecting the points intersects with any of the polygons from the `World2D`. However, since this had to be done many times (in the worst case, the number of iterations would equal the number of nodes on the path minus one multiplied by the number of polygons), this would be far too complex. Therefore, line of sight determination is done using the quadtree's structure.

1. The node representing the smallest area containing the line's starting point and end point is found. The algorithm recursively determines which child of a node contains both points. If the node itself but none of its children contains both points, it returns itself as a result. Otherwise, the child node containing both points becomes responsible for finding the right node.

2. When the correct node has been found, the nodes that have to be checked for an intersection can be restrained to its children.

3. The intermediate nodes, the grid cells intersected by the line, are determined similarly. The node determined above checks which of its child nodes are intersected by the line.

4. If an intersected node is a leaf, it is added to the result collection, otherwise its children are checked. That way, all cells intersected by the line are collected without checking all cells in the map.

5.4 COINS: INDOOR NAVIGATION SUPPORT FOR OSE

Many people, especially business travelers, are often confronted with the situation that they are visiting new places and have to figure

out how to navigate new destinations. Path finding is an frequently-occurring problem in computer science. Accordingly, many past studies and applications have been performed on this subject. These studies provide several interesting insights, such as primary criteria for human path selection [86] and how people often prefer routes that are easier to describe and follow over theoretically optimal routes. An important indicator of perceived easier-to-follow routes are those with fewer instructions. To achieve this goal, different non-optimal algorithms that provide simpler paths were proposed, e.g., by Liu [127] and Richter [174]. However, these approaches were focused on street navigation, where the movement of people and vehicles follow the rules of street orientation.

An important aspect that has been overlooked by literature is the need for directions to adapt to the current state of the user and to the condition of the environment in indoor navigation applications. The variation of conditions people may encounter in environments such as airports, museums, corporate campuses, shopping malls, and factories is so diverse that they merit consideration when determining the most desirable path for a person to follow.

The context-aware indoor navigation system (CoINS) aims to provide efficient user navigation in buildings with a strong emphasis on the "human factor". When considering the human as part of the system, the term efficient no longer corresponds to the shortest path calculated by a mathematical method. To efficiently navigate users to their destinations, it is also vital that they quickly comprehend and execute the navigation instructions they receive from the navigation system. For example, a good route description would consist of a low

Figure 5.12: A user scanning a QR-Code.

number of turns and turns should be located at easily-identifiable landmarks at which the user could easily see in which direction they should walk. The user model and the path selection algorithm pre-

sented in this section allow the inclusion of such considerations in route planning.

Path calculation constitutes an important subcomponent of CoINS but, in this context, only considers the geometric aspects of route planning. In early work [136], only models and algorithms based on quadtrees [184] were used. This limited the scalability of the system to larger buildings. The present CoINS system is able to efficiently find the geometrically shortest paths because of two extensions. First, the symbolic part of the hybrid location model is now based on a hierarchical graph. Second, the search path algorithm was improved with a heuristic to further reduce the search space. These two factors allow to keep the single graphs small to reduce pathfinding complexity to $O(|L|-1)+O(|V_i|)$ in the average case, and $O(|V_i|^2)$ in the worst scenario.

5.4.1 *Customizable Guidance Process*

The CoINS system decouples the process of guiding a user by including a small, embeddable workflow engine called Micro workflow [145]. In this way, both the supporting services and the process of indoor user guidance can be further modified. Another advantage is that this approach also enforced further separations of responsibilities in order to allow a possible change of the workflow engine if required. Figure 5.13 shows the implementation of a guidance workflow.

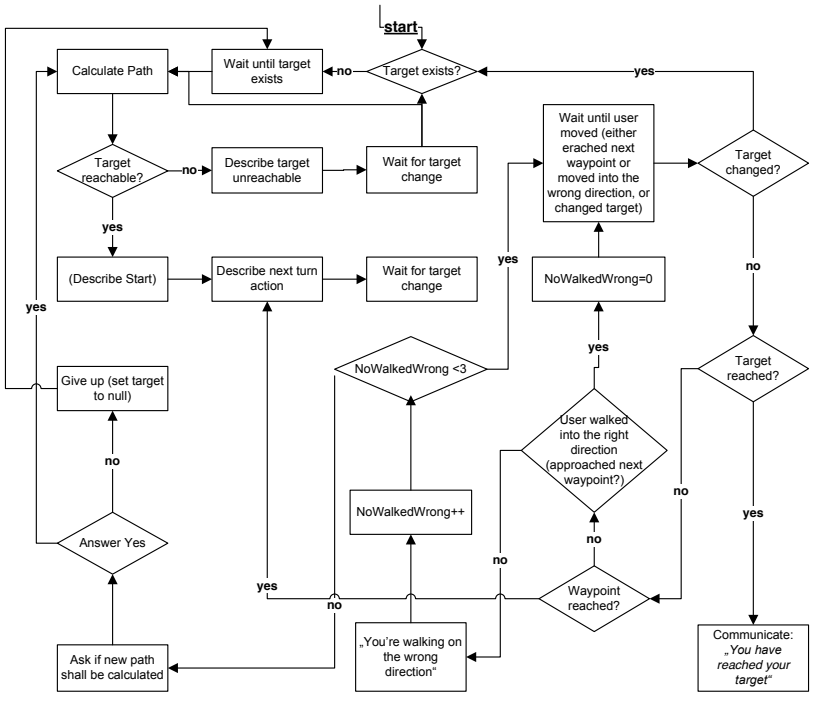


Figure 5.13: Overview of the User Guidance Process for Indoor Navigation.

5.4.2 *Path Generation and Selection*

To select the most suitable path for a specific user we use the Simple Multi-Attribute Rating Technique (SMART). Under this technique, every path can be described by individual preference attributes and through the value functions of each single attribute, the preference strength can be measured.

The process of decision making can be influenced by the user's consideration of what attributes are more important or relevant to him. For this reason, importance weights are given to attributes in order to model this kind of situation. The importance weight alters the output of an attribute's value function.

The measured preference strength of every single attribute of a path can be aggregated to return an overall path evaluation value. This resulting value will be later used to compare against other possible paths.

5.4.3 *Preference Evaluation using the SMART approach (Simple Multiattribute Rating Technique)*

Our optimal path selection problem using SMART is formalized as follows:

Let P be the set of all feasible paths, then $P = \{p | p \text{ is a path over the location model}\}$ and a_n all attributes with importance weights w_n and value functions v_n. Then, the optimal path $p \in P$ maximizes

$eval(p) = aggregate(w_i(v_i(a_i(p))))$ for i = 1,..., number of attributes

The following five steps describe how the path selection using the SMART method is conducted:

1. **Define the feasible paths and attributes for evaluation.** All feasible paths are generated. Given that locations are connected as a graph in the location model, all paths can be generated by traversing the graph using a depth first search, A* or Dijkstra. User constraints (e.g. physical disabilities) can be used at this stage to avoid traversing unnecessary paths thus filtering them out from the selection process.

2. **Assign the importance weights to the attributes.** The importance of attributes to the user are established by assigning importance weights to attributes. These weights are normalized, which means in this case that the sum of all weights equals one. This weight assignation allows an ordering of attributes according to their relevance and therefore the preference strength of an attribute with a high importance weight will have a higher impact on the path evaluation than those with lower. Importance weights can be constants or functions. The latter is necessary when no constant ordering of path attributes can be modeled. For

example, if the importance depends on the preference strength of some other path attributes.

3. **For every path, assess the path attributes separately.** At this step the iteration of value measurement over all attributes for every path generated in the first step is performed.

4. **Calculate evaluation value by aggregating all attribute assessments under consideration of importance weights.** In the fourth step, assessment values of paths attributes from step three are aggregated. Different aggregation models exist to approach this. For SMART, a very common model is the additive, which sums up all assessment values from attributes multiplied by its respective importance weight, i.e. $eval(p) = \sum_{i=1}^{n} w_i d(a_i(p))$.

5. **Select the path with the best evaluation value.** In the last step, path selection can be done by searching for the path with a target evaluation value. How the target evaluation value is defined can depend on how the value scales were created during path attribute assessment. For example, if the assessment functions map into numerical values between 0 and 100 where 0 means the least desirable and 100 the most, then path selection will concentrate on paths with highest evaluation values.

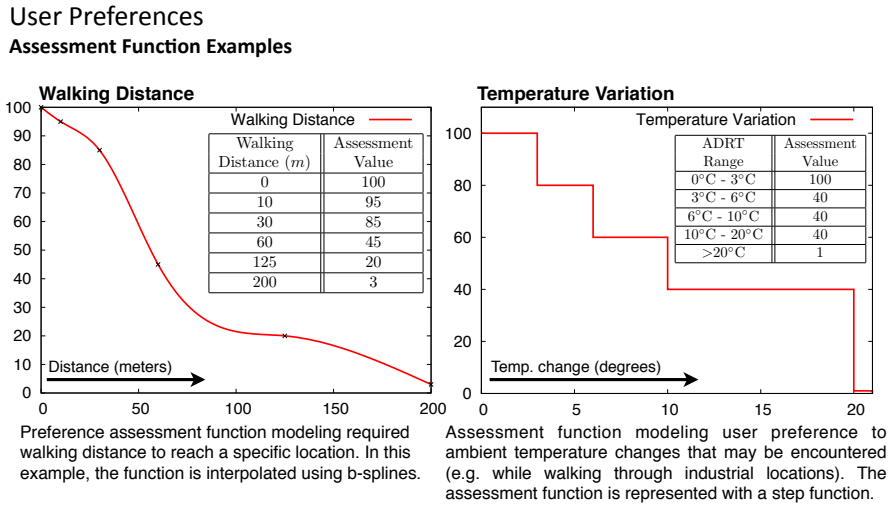


Figure 5.14: User Preferences examples from the set of preferences elicited for CoINS, later in this chapter.

5.4.4 *CoINS Architecture*

Figure 5.17 shows the architecture of the CoINS system. The *Presentation* component of CoINS can be either accessed as web interface through a web server or from rich clients. The web-based solution has the advantage that no software deployment is necessary on clients, but

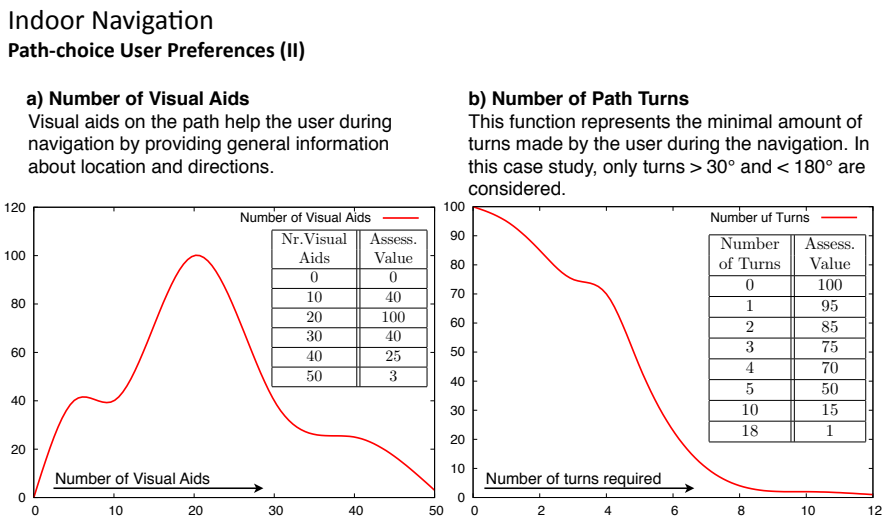


Figure 5.15: User Preferences for Indoor Navigation path selection process.

Attribute	Weight	Path 1	Path 2	Path 3
Path length	0.4	39.31 *(65m)*	51.67 *(55m)*	18.6 *(138m)*
Number of visual aids	0.2	99.29 *(19)*	96.52 *(18)*	3.09 *(50)*
ADRT	0.1	100 *(4°C)*	100 *(6C)*	0 *(29°C)*
Number of turns	0.1	94.03 *(2)*	94.03 *(2)*	80 *(5)*
Path Crowd Density	0.05	89.7 *(0.11p/m^2)*	91.66 *(0.09p/m^2)*	87.69 *(0.13p/m^2)*
Crowd Density Variance	0.15	99.22 *(0.012(p/m^2)^2)*	86.51 *(0.008(p/m^2)^2)*	98.86 (0.017(p/m^2)^2)
Evaluation value		74.35	**76.93**	35.27

Table 5.4: Assessment of path attributes. Path 2 is the selected path.

location tracking is limited to QR codes or purely infrastructure-based solutions. In contrast, rich clients can provide more customized user interfaces and support additional local positioning systems.

The *Path Calculation* component determines all possible routes from a given source location to a given destination location. In addition, it provides the geometric length of each route. This component only considers the location model.

The *Path Selection* component uses *Preference Assessment* to calculate all non-location related metrics for all candidate routes and then selects the best route. The candidate routes have been previously determined by Path Calculation. Preference Assessment uses the route, the user model, and the current context as inputs. The whole process is controlled by a small *Workflow Engine*. Using a workflow allows us to decouple the guidance process from the CoINS core system. CoINS server, the Context Server, the location tracking systems, and rich clients are based on our communication middleware MundoCore.

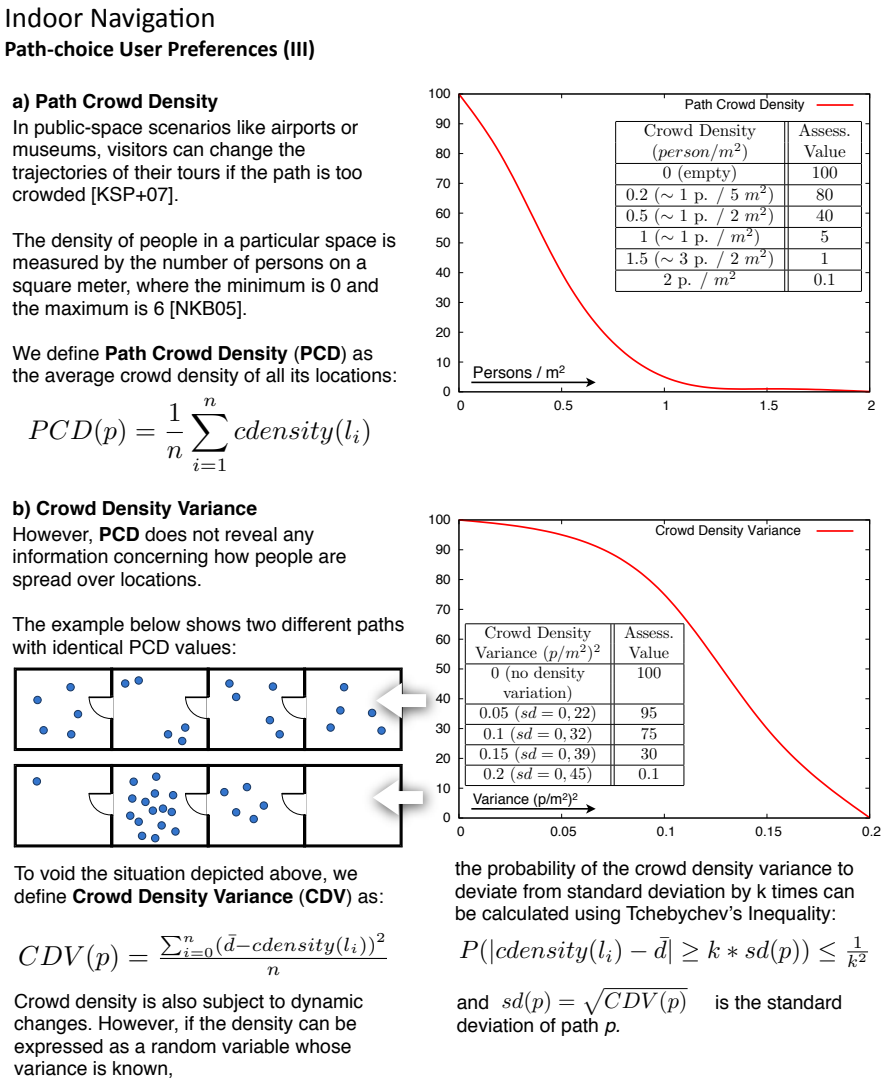


Figure 5.16: User Preferences for Indoor Navigation path selection process.

MundoCore and the Context Server are described in more detail in the following sections.

Different techniques and requirements have been studied for guiding users indoors with mobile platforms [41, 81]. Guiding techniques includes floor maps, schemas, and spatial landmarks [198] together with their spatial relationships to help build a "mental trip" on the guided user.

The user guidance in the current version of CoINS is realized through a visual modality, where instructions are displayed in the user's terminal. We have so far developed 2 different versions for the user terminal. The latest one is a "thin" client, in order to support the common situation where no software deployment can be done on the user terminal.

In a typical scenario, a user enters the building and points with the mobile's camera to any of the QR-Codes available in every door

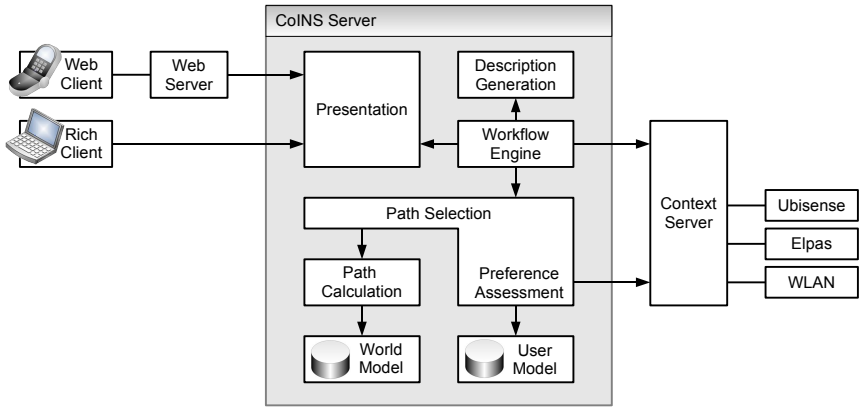


Figure 5.17: CoINS System Architecture.

(Figure 5.12). A URL is encoded that takes the user to a welcome page where information of the current location (people to contact, office hours, contact information) is displayed, and the question is the user requires directions. When the user selects this option, she can either type the name of the person or room number, and the system starts the guiding process (current location is assumed to be where the QR-Code was read).

The user continues receiving instructions every time she requires it. Along the way, the user may feel insecure about the current location and how to proceed. In such cases, just by scanning any QR-Code, CoINS resumes que navigation from the current position to the originally specified.

Indoor location systems are seldom installed in large scale settings, and can be either very expensive or impractical following the building construction characteristics. Therefore, the choice of a standard such as QR-codes appears as a natural solution. Clearly, the limitations of the thin client and QR-Codes as a location systems makes it difficult for proactive guidance. However, we are currently experimenting with new approaches to overcome these restrictions.

5.5 EVALUATION

Two different experiments have been carried out to study the paths people follow to reach a given destination in indoor settings. The purpose of the first evaluation was to gather information to find a common profile for the experiment, that is, a set of assessment functions and a ranking of the different criteria to estimate their weight in the path selection assessment procedure (shown in table 5.4).

For the first experiment, eight subjects where scheduled, all of them familiar with the Piloty Building at the University of Darmstadt, a

four-story building, that offers in every floor five different stairs and two elevators to change floors(see figure 5.5). A set of pairs of origins and destinations were carefully selected, and the subjects were asked to carry out simple tasks that implied walking between the selected destinations(e.g. *"You need to pick up a form in office XXX"*, *"Take the form to Mr. YYY in office XYZ"*, *"Go to the library"*,*"Get a Coffee at the Bistro"*, etc.).

To begin the experiment, subjects were read the appropriate directions for their assigned route at the origin. Then, the subjects began walking to the given destinations. The routes every participant followed were recorded in separate maps of the area and all possible paths were coded to record every time a particular path was followed. Subjects then completed a questionnaire on the criteria they used in selecting a path.

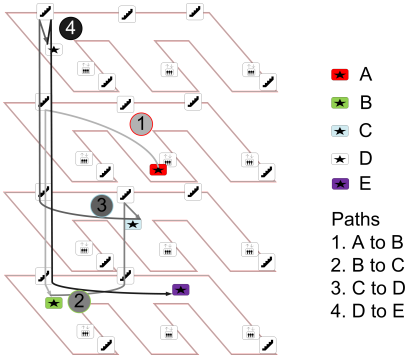
The second experiment involved nine subjects (five female, four male) that did not participate in the first experiment. The same set of destinations as in the first experiment were chosen, and the same tasks were assigned to the subjects, following the first experiment's procedure. The data and user preference profile information gathered during the first experiment was used as input for the algorithm to calculate the suggested path for every participant. Thus, the contextual characteristics at the moment when the experiment took place were taken into account. Then, the predictions were compared directly to the path actually followed by the subject. The results of this first evaluation of the system are shown in Table 5.6. These results suggest that the proposed paths by system matches those chosen by people at a particular contextual setting. In Table 5.5 a schema of the building is shown along with the proposed paths by the system and the alternatives performed by the users. Although encouraging, this study must also be performed in future work in large indoor scenarios such as airports, where users may have more options of movement according to the context.

Table 5.6 shows the reduced search space results from the heuristic implemented in CoINS. The algorithm allowed a reduction of potential paths to be analysed by more than 82% (114 found by the general Dikjstra vs 20 from CoINS spearch space reduction + Dijkstra). The results of the user behaviour are encouraging: more than 86.1% of the users followed the paths suggested by the algorithm.

5.6 CONCLUSION

This chapter introduces two central building blocks of OSEs: People and Space and system support for People to navigate that Space. The *People* model extends UserML combining the user data with information about physical capabilities, preferences, skills, and actions. Physical capabilities and preferences are required to enable smart

(a)



(b)

Subj.	M/F	A to B Path Chosen	B to C Path Chosen	C to D Path Chosen	D to E Path Chosen
1	M	Y	Y	Y	Y
2	M	Y	Y	Y	Y
3	M	Y	Y	N	Y
4	F	N	Y	Y	Y
5	F	Y	Y	Y	Y
6	F	Y	Y	N	Y
7	F	Y	Y	Y	Y
8	M	N	Y	N	Y
9	F	Y	Y	Y	Y

Table 5.5: (a) Shows the structure of the Piloty building, and the paths proposed for evaluation. (b)Table shows for each participant and path, whether the path followed by the subject matches the path proposed by the system.

	Path 1	Path 2	Path 3	Path 4	°
♯ Dijkstra's generated Paths	24	9	24	57	
♯ CoINS heuristic's generated Paths	9	9	1	1	
Users that followed path as predicted	7	9	6	9	**31**
Users that followed a different path	2	0	3	0	**5**

Table 5.6: Generated paths and evaluation results.

services such as CoINS (the indoor navigation service). At the same time, AI planners can use skills and actions to combine user actions with those that smart products can do to generate more complete instructions involving both Products and People.

The *Space* model represents the physical spaces in which human activity occurs and where Smart Products make available their functionality. The Space hybrid model was developed in several iterations,

creating a sophisticated model to represent realistic settings that can support smart services such as indoor navigation. This chapter demonstrates several aspects of its implementation, algorithms developed and data structures.

The major contribution of this chapter is the introduction of *CoINS*, a context-aware indoor navigation system, together with the required algorithms and services, and presented a detailed discussion of its implementation. The CoINS implementation takes advantage of the Space model and the techniques developed in Chapter 4 for Ubiquitous Explanations. The CoINS service addressed in its architecture and implementation several challenges in providing this functionality with in both mobile and infrastructure settings. From the architectural point of view, it also introduces a novel degree of customisation by externalising the user guidance process into a workflow description, adding a degree of flexibility that makes CoINS more adaptable for many scenarios.

6

HIGHER AMBIENCE IN OPEN SMART ENVIRONMENTS

6.1 INTRODUCTION

This chapter introduces new concepts to extend OSEs scenarios and to support higher levels of context awareness and adaptation capabilities. First, it introduces a model called *Embedded Preferences for Products (EP4P)* that enables smart interactions between Smart Products throughout their entire life cycle, as proposed in chapter 3. A sample scenario called *"Compara"* demonstrates a potential application of *Embedded Preferences* with a novel OSE scenario in a mobile shopping setting. Afterwards, the chapter presents a granular model for assessing and adapting to higher abstraction levels of context awareness called *Embedded Situations*. *Situations* describes more complex context settings taking place in an environment which are required to enable OSEs to achieve better adaptation and anticipation to their inhabitants. Finally, the chapter introduces the concepts of *Areas of Influence* and *Places* to support multiple concurrent activities in an environment.

6.1.0.1 *Publication*

This chapter is based on the following publications:

- Fernando Lyardet, Aristotelis Hadjakos, and Diego Wong Szeto. "In-Situ: An Approach for Dynamic Context Labeling Based on Product Usage and Sound Analysis." In: *Workshop Proceedings of the AAAI-11 Conference*. 2011

6.1.0.2 *Contribution Statement*

The concept and implementation of *Embedded Preferences* was done by me, as well as the conceptional design of Situations for Smart Products, which later derived into *Embedded Situations*. *InSitu* was developed in collaboration and advice of Prof. Dr. Hadjackos and the student Diego Wong Szeto, who helped with the implementation. Prof. Hadjackos actively supported the InSitu in the areas of sound processing and machine learning models for situation labelling.

6.2 EMBEDDED PREFERENCES FOR PRODUCTS

The concept of embedded Preferences for Products (EP4P) describes a product's physical and operational characteristics that can be used to assess their application in a given physical context. Using this information, it can be established whether or not a given product could be used in a given environment, conform to safety norms, and still perform as intended by the manufacturer.

The term *preference* is usually associated with people, to refer to the desire or predisposition of a person in favor of something. The use of this term in the context of Smart Products signifies first, the reflective character of a product with embedded information about itself and its use, that can now inform to people and other products throughout its lifecycle. And second, it denotes the *soft*, non-prescriptive nature of preferences.

Some previous work has already tested the embedding of similar information, although in a more limited fashion and expressed in boolean rules. The limitation of such an approach for consumer products is that such mechanism for expressing preferences can easily run into rule consistency conflicts when the rule sets are large.

Embedded preferences are defined through a list of tuples describing the different kinds of preference information. These preference tuples can be either embedded within a smart product or available on the web through the manufacturer. They are communicated through xml using the Preferences for Products Markup Language (EP4PML).

The EP4PML language defines the set of preferences associated with a product, and can associate or redefine them for a particular stage in the product's lifecycle (PLC). The inclusion of the PLC concept allow

product preferences to be grouped according to their relevance in the different product life cycle stages. PLC information also allows for a better integration with the many other objects and environments that a given smart product encounters throughout its whole existence, as opposed to focusing only in consumer support. This a subtle change of perspective (looking at the world from an imaginary product stance), to turn Smart Products into the connecting line between multiple smart settings.

The EP4P information may take different forms, as the nature of the information described is heterogeneous as well. We present an enhanced representation based on preference assessment functions, similar in nature to those of the multi-attribute utility assessment. The extension done is later described in section 6.2.2.

The preference tuples may take one of the following forms:

{name, ontologyRef, f(x), interval, units, DmOntRef, description} : describes a preference in terms of an assessment function, and the domain of such function.

{name, ontologyRef, value, valueFormat, units, description}: describes characteristics that can be quantified with a value (e.g. expected Voltage).

{name, ontologyRef, interval, units, description}: description of a product characteristic in terms of an interval (e.g. operating temperatures).

The following chart describes the meaning of the tuples' components.

6.2.1 *Modeling Product Preferences with Multiattribute Utility Theory*

There are three key requirements that the implementation of preferences should ideally meet: a) the ability to model and evaluate an arbitrary number of preferences, b) provide a framework for deciding when conflicts appear among multiple criteria and c) the ability to model objective and subjective criteria.

There are several alternatives to modelling preferences such as Multi-Attribute Utility Theory (MAUT) [186], Analytic Hierarchy Process (AHP) [182], Multi-Criteria Decision Making (MCDM)[183], Conjoint Analysis .[172], Regression Analysis [153], and decision trees [102] are the most mature and widely established techniques.

Multi-Attribute Utility Theory (MAUT) has several advantages over the other methods for modelling preferences:

- *Flexibility*: MAUT provides a flexible framework for modelling preferences, as it can analyse a wide range of decision problems with multiple conflicting criteria.

Name	Description
prefName	preference name.
ontologyRef	ontological reference to disambiguate the meaning of the preference.
f(x)	the preference description function is a utility function $u_j : S \to \Re \in \{0, \ldots, 100\}$, where S is define in DmInterval. The function defines the preference choice so that: $u_j(s_i) \geqslant u_l(s_k) \Leftrightarrow$ the preference chosen is s_i rather that s_k.
DmUnits	preference function domain units.
DmInterval	preference function interval where it is defined.
DmOntRef	ontological reference to disambiguate f(x) domain units.
value	preference value.
valueFormat	format used to model the preference value.
interval	describes a value interval where the preference is defined.
Description	textual description used for providing explanations to a user.

Table 6.1: Component description of a product preference tuple.

- *Quantitative Approach*: MAUT is based on a quantitative approach, which allows calculating numerical scores for each option based on multiple criteria. This makes it possible to compare options objectively and consistently.
- *Incorporation of Probabilistic Information*: MAUT allows for the incorporation of probabilistic information into the decision-making process, which can be useful in situations where there is uncertainty about the outcomes of different options.
- *Integration of Subjective and Objective Criteria*: MAUT provides a framework for integrating both subjective and objective criteria into the decision-making process, allowing for a more comprehensive analysis of the options being considered.
- *Ability to Handle Complex Problems*: MAUT is well-suited for handling complex decision problems with multiple criteria and options, providing a structured approach for breaking down the problem and analysing the options systematically.

In comparison, other methods such as AHP, Conjoint Analysis, and Decision Trees may have limitations in their ability to handle complex problems with multiple criteria or their lack of a quantitative approach to modelling preferences. Regression analysis is limited because it can only model linear relationships between the dependent and independent variables and may not be appropriate for modelling complex preferences.

Multi-Attribute Utility Theory (MAUT) is a method for modelling preferences that quantify the trade-offs between conflicting criteria. It provides a systematic and structured approach for evaluating decision options based on multiple attributes. The evaluation process is presented in detail in [186]. We summarise it here for convenience and completeness:

Each item is described in terms of n attributes $a_1, \ldots, a_n$ which we can assume values from the domains $V_1, \ldots, V_n$. A concrete item e can be represented as a tuple $e = \langle x_1, \ldots, x_n \rangle$.

According to MAUT: *"the overall evaluation $v(e)$ of an object e is defined as a weighted addition of its evaluation with respect to its relevant value dimensions"* [186, 223]. The common denominator of all these dimensions is the utility for the evaluator. For example, a car could be evaluated on dimensions such as build quality, acceleration 0-100km/h, deceleration, fuel consumption, load capacity, size, weight and handling.

The overall evaluation is defined by the following overall value function:

Definition:

- the set $S = \{s_1, \ldots, s_m\}$ of preferences,
- the set $A = \{a_1, \ldots, a_n\}$ of attributes,
- the set $V = \{v_1, \ldots, v_n\}$ of utility functions $v_j : S \to \Re \in \{0, \ldots, 100\}$, that defines the preference choice so that $v_j(s_i) \geqslant v_l(s_k) \Leftrightarrow$ the preference chosen is s_i rather that s_k.

In the following table 6.2 we see some examples of such preferences and attributes:

F	= {f | f is an assessment function}
E	= {e | e is an event}
X	= {x | x can be assessed with some $f \in F$ }
where	$f : \mathcal{P}(E) \times X \to \{0, \ldots, 100\}$
and:	
E	= (SUBJ, COND)
SUBJ	= {user 1, user 2, room a036, ...}
COND	= {STATE, ACTIVITY, LOCATION, TIME}
STATE	= { happy, busy, available, ...}
ACTIVITY	= { run, read, work, ...}
LOCATION	= {room a036, kitchen, ...}
TIME	= {starttime, endtime}

Table 6.2: Multiattribute Utility Theory definitions, and attribute examples

Definition:

- $s_i \in S$ and $n = |A|$, the preference function $P : S \to \Re$ is then defined by:

$$U(x) = \sum_{i=1}^{n} w_i v_i(A_i) \tag{6.1}$$

Here, $v_i(x)$ is the evaluation of the object on the i-th value dimension and w_i the weight determining the impact or *relative importance* of the i-th value dimension on the overall consideration, n is the number of different value dimensions being considered, A_i is the set of all attributes relevant for $v_i(x)$, and

$$\sum_{i=1}^{n} w_i = 1$$

For instance, for deceleration, we should consider flat dry, downhill dry with load, and flat wet, while for load capacity there is only one attribute, the total weight the car can safely transport. In order to evaluate these disparate attributes, it is necessary to construct a scale representing the properties of the levels of an attribute (e.g: a scale from 0 (worst) to 10 (best) that can serve as a measure of the evaluation). The resulting evaluation function form will depend on whether the attributes being modelled are continuous or discrete[186].

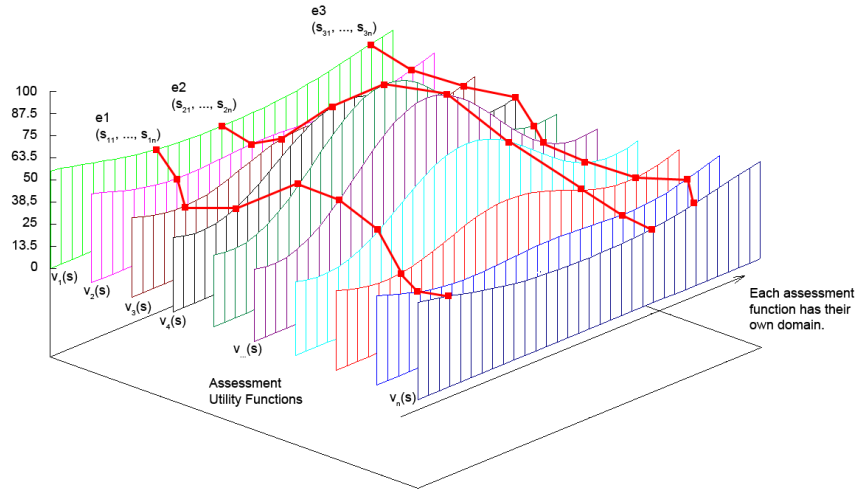


Figure 6.1: MAUT's Assessment of Utility Functions

6.2.2 *Extending the Model's Expressivity and Decision-Making Characteristics.*

The original utility theory does not correctly solve specific decision problems. As noted by [40], conflicts may arise when the worth of a set of attributes may vary in a particular context. For instance, in

Section 6.3 (Compara), a property describing the minimal distance that a Product may be from a heat source becomes invalid (or completely "worthless") if there is no known heat source. Therefore we extend our model following [40] by introducing two different decision-making strategies:

- The *conjunctive rule* determines a value for each attribute under which its utility is not allowed to be. A possible implementation is a step function representing the minimal value that must be reached. Should that not be the case, the utility is significantly reduced by a value surpassing all other possible outcomes. The conjunctive rule is often used when a failure on any attribute is considered critical and cannot tolerate compromises.
- The *disjunctive rule* states that given a strategy, the utility in at least one attribute should exceed a specific given value for this attribute. For instance, applying a large constant to the model gives preference to a strategy with an attribute above a certain value, provided no other attribute satisfies the conjunctive rule. The disjunctive rule is often used in situations where a failure on any one attribute is not considered critical, and can tolerate some compromises.

For the sake of completeness, we reproduce here the definitions of these two rules as stated in [40].

The conjunctive rule: Following the definition above, $U_j(c_{i,j})$ is:

$$V_i(x) = \begin{cases} f_i(x) & \text{if better than threshold value,} \\ -K & \text{otherwise} \end{cases}$$

With the constant K selected in the following way:

$$K > \max \left\{ v(x) = \sum_{i=1}^{n} w_i v_i(x) \right\} + 1$$

The disjunctive rule: Following the definition above, $f_j(c_{i,j})$ should be:

$$V_i(x) = \begin{cases} k & \text{if better than threshold,} \\ f_i(x) & \text{otherwise} \end{cases}$$

where:

$$k = \max \left\{ v(x) = \sum_{i=1}^{n} w_i v_i(x) \right\}$$

Therefore, the utility function in the extended model is as follows: The utility function $Vi(x) : C \to \Re$ is then defined by:

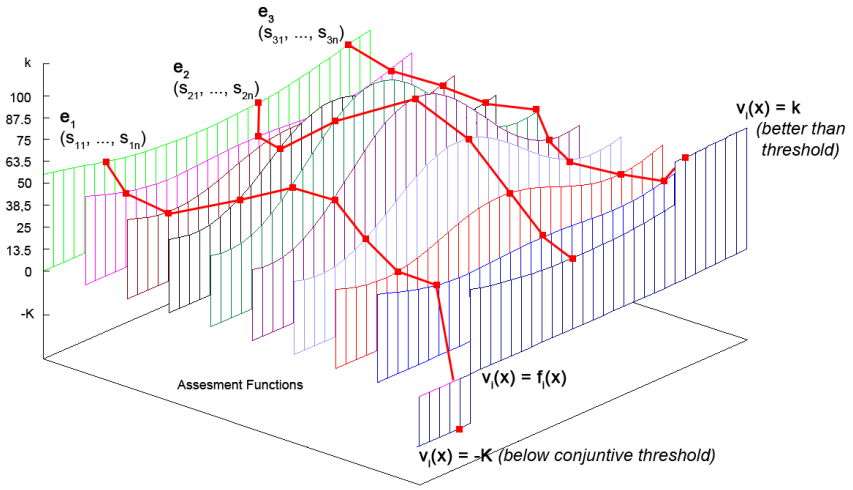


Figure 6.2: MAUT's Assessment of Utility Functions

$$V_i(x) = \begin{cases} k & \text{if better than threshold,} \\ -K & \text{if worse than threshold,} \\ f_i(x) & \text{otherwise} \end{cases}$$

and

$$k = \max \left\{ v(x) = \sum_{i=1}^{n} w_i v_i(x) \right\}$$

, and

$$K > |k| + 1$$

6.3 COMPARA: A SMART SHOPPING SCENARIO FOR EP4P

Very often consumers must select and make a choice about products without having enough information, or even understanding certain product qualities. Taking advantage of the new capabilities and information embedding that smart products bring, we present a novel approach for consumer support.

Consumers often face the challenge of selecting products that meet their needs and preferences. To address this, we propose a novel approach to consumer support that leverages the capabilities and information embedded in smart products.

Current mobile shopping systems offer various tools to support consumers, including price comparison agents, recommendation systems, and location-based services. Research has shown that people rely on recommendations as information sources, and they infer the experiences of others as "virtual experiences" [22]. In particular, virtual experiences are influential in evaluating "experience products" that require firsthand experience to be assessed, such as movies, wines, and books [21].

However, more than just virtual experiences and impressions may be required for products that require specific use and application. For example, questions such as "Will this mower be right for my lawn?" or "Could we install this photocopier in our printing room?" require factual information beyond the experiences of others. Current recommender systems need to satisfy this need for information adequately.

To overcome this limitation, we propose the concept of "Embedded Preferences for Products" (EP4P), which involves embedding information about product function, behaviour, and intended use within the product itself. EP4P provides consumers with specific information that helps them decide whether a product suits their needs.

Each product preference in EP4P describes a particular aspect and a preference profile for a specific product, and there can be many preferences. When choosing a product, consumers consider both objective and subjective factors. Objective considerations involve factual information about the product, such as whether a particular model can be safely used in the intended environment. Gathering and comparing this information can be time-consuming and prone to errors. However, embedding EP4P within the product can simplify this process for consumers.

Let's consider the following scenario shown in figure 6.3 (revisited from Chapter 3):

Compara

Sample Scenario

Susan and John went shopping for a few things they need at home. Susan has decided to buy an electric heater for the bathroom, and that new coffee-machine she saw on TV. John instead is looking for lawn mower. The old machine is broken, and with the spring about to start, he will soon need a replacement.

At the shop, she finds some interesting heater models, and she is wondering whether that new model would be the right one for her. With her mobile, Susan collects information of a nice model using her mobile by scanning a 2D QRCode. The information is downloaded to a local database in her mobile phone, and after a short comparison process, a list of possible locations for that heater appears on her screen. She wanted the heater for her bathroom, but it doesn't appear in the list possible locations at home.

-Why not in the bathroom? She asks. The answer appears in her mobile screen:

`That heater is not recommended for very humid spaces How about these other models?`

In the meantime, John searches for the lawn mower he needs. Like Susan, he finds a model he likes, and collects the mowers data with his mobile phone. However that lawn mower is too small and slow for the size of John's garden. In addition to that, where John lives is more dry and a harder kind of grass often grows in that area, that would wear the machine faster than it should. An alternative list of machines is presented in John's screen, and he selects one that fits his budget.

Figure 6.3: Sample Scenario of Smart Mobile Shopping with Embedded Preferences for Products

The ability of products to embed lifecycle preference information makes it possible to overcome situations likes those described above. Products that know about themselves and their operating capabilities and restrictions, may share that information and help a customer make a more informed choice.

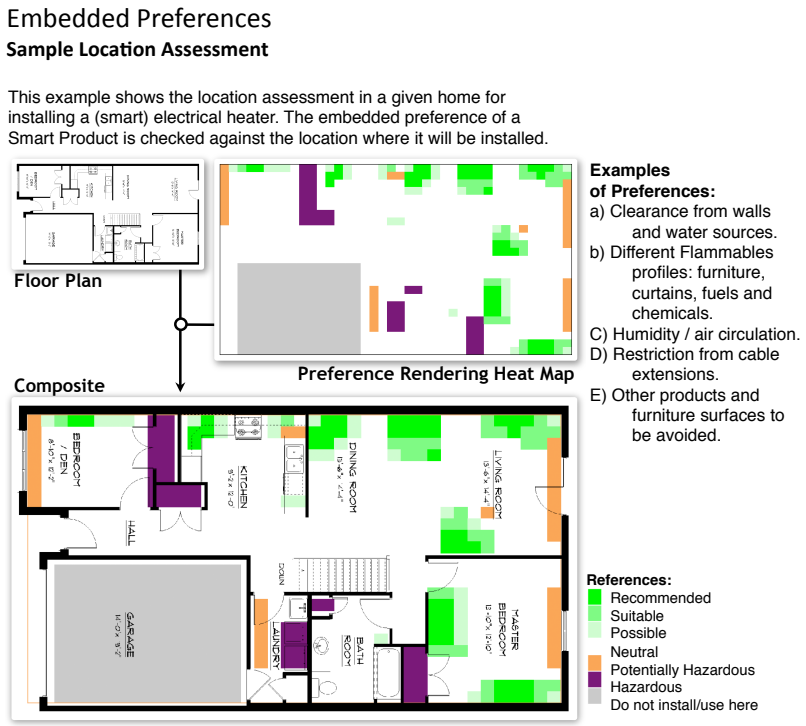


Figure 6.4: Product Location Preference Render using a "heatmap" format, to distinguish in which areas a given product may be installed.

Customers in a store may at any time collect information about the products they're interested in by scanning their barcodes or qrCodes, or by near field communication means. The information about a product's preferences is collected , and the customer can quickly checkhow that product would fit in a given place.

Such assessment functions require context information in order to determine their value. For instance, distances from the wall, power outlets, or water sources. These information need to be calculated throughout the whole space where the assessment function is to be evaluated. Such calculation is done by the location preference rendering map component.

Once a product information has been collected, the user may select different locations where the products my be used or installed. Preferences are then evaluated against the particular characteristics of the selected locations, and display the results to the user in the form of a map showing the locations and places within those locations where the product may be safely installed.

In case incompatibilities are found, an explanation is offered to describe why particular locations may not be suitable. Together with this information, a customer can decide with much better information, whether or not to buy a product.

Product preferences can be used in other stages of a product's lifecycles: at home for unpacking, location and setting up, and disposal;

for distribution, products can provide information about their storage preferences.

The location preference-rendering is an extension to the indoor location model presented in chapter 5. The extension allows collecting specific information in addition to the geometrical description of a location. The information collected records the available floorplan, storage room, and places where objects could be laid. The preference-rendering map also stores information about location volume, distances from power outlets, flammable objects, windows, doors and many more if necessary. The particular valuation of such information throughout the location is calculated by iterating over a discrete partition of the space called reference cells. These cells are discrete locations forming a grid over the whole available space where the aforementioned information is calculated, and later stored in the location preference-rendering map.

The collected information is not subject to frequent changes. Therefore, it makes sense to perform the calculations only when such changes occur, instead of every time the information is required. This allows for a much faster user assistance response and less computing power.

Finally the assessment process of the product preferences can take place either at home, at the user's mobile, or through a service running in a server cloud. One of the topologies explored in this work is using the increasingly powerful mobile phones.

6.3.0.1 *Location Preference Rendering Map*

Preference maps include much information besides the location structure used to render a product's preference functions. Sample pieces of information are distances from power outlets, flammables, windows, etc. This information requires to be evaluated throughout the whole space. The values obtained through a discrete partitioning of the space are stored in the rendering map component as either a value sequence, or as a sequence of singletons, according to the distribution of values being obtained. The advantage of using singletons is speeding up the preference rendering by reducing the number of assessment calculations needed.

The preference rendering process, that is, calculating the preferences of all known objects in a given space, is as follows: if values for the assessment are stored as enumerations in the rendering map, each one is evaluated with the corresponding assessment function. The resulting values are accumulated in each cell, so storing all those intermediate values is unnecessary. On the other hand, singletons have been stored in the rendering map. The assessment process takes place throughout the stored singleton values, and the resulting calculations are propagated to the referencing cells.

6.3.0.2 *Location Preference Rendering Map*

Let's assume we want to put a box B (the bounding box of an object we are interested in) into a scene R where we already have many other objects, each represented by its bounding box. Also, let's assume we have a set of sources Sj, both positive and negative and with different intensities (i.e., weights), that emit some signal that attracts or repels B. For this, let's first define some utility functions:

```
Function renderScene(R, ResX, ResY):
     Generate zenital orthogonal camera c.
     Set c viewport to ResX, ResY
     Render R from c, including all boxes and room walls.
                                  // Sources do not need to be considered
Return rendered image

Function projectObject(B, ResX, ResY):
    Generate zenital orthogonal camera c.
    Set c viewport to ResX, ResY
    Render B
Return rendered image

Function Dist(A,B):
Return sqrt((A.x-B.x)^2 + (A.y-B.y)^2 + (A.z-B.z)^2)
```

Now, before anything else, we need to know, given a point in the scene, how much repelled or attracted it is from the sources Sj, with a global (arbitrary) tuning constant g:

```
Functioneach s in
    Total = 0
    For each s in {Sj}:
    d = Dist(p,s)
    Total += g * s.intensity * 1/d
                  //The closer we are, the more intense it is
Return Total
```

At this code, we assume the signal's intensity is reduced with the distance as $1/d$, which may or may not be the case (sound and light propagate with $1/d^2$, so we already have counterexamples for this supposition). Now, given an object with bounding box B, we need to integrate the influence from all the sources on its volume. However, we have the projectObject function to help us with this:

```
// Here ResX, ResY will give the precision of the approximation.
// The larger these variables, the more precise but also the slower everything will be...

Function approxIntegrate(B, {Sj}, ResX, ResY):
    rasterizedObj = projectObject(B, ResX, ResY)
    total = 0
    For each pixel p in rasterizedObj:
          If p belongs to B:  // basically, if rasterizedObj is a binary image, this is p == 1
               total += eval(p, {Sj}) // Assuming linear behavior
Return total
```

Now, we can also compute the values for all the pixels in the image version of the scene:

Finally, other objects may block the effect of the sources. This would force us to modify the red code in the eval function above to:

```
Function approxRoom(R, {Sj}, ResX, ResY):
    Output = newImage(ResX, ResY)
    For each pixel p in Output:
          Output[p] = eval(p, {Sj})
Return Output


att = 1
For each object o in R:
     If o between p and s
           att = compute an attenuation value
           // if more than 1 object is attenuating, we
           // should compose their attenuations…
           // for instance, as att *= attenuation value
           // in this case, the attenuation values should go between 0 and 1
Total += g * s.intensity * 1/d * att //The closer we are, the more intense it is
```

Now, if it would be desired to add an optimiser only needs an evaluation function, which can be built using approxIntegrate above.

6.4 EMBEDDED SITUATIONS

In a smart environment, context gathering and interpretation are crucial for providing its inhabitants with a personalised and proactive experience. However, current smart environments typically rely on centralised and highly customised approaches for context gathering and interpretation. Although these approaches offer sophisticated monitoring and adaptations, they have slowly moved from prototypes into our daily lives.

In this thesis, we propose an alternative model called the OSE model, which departs from the centralized view. However, this model presents challenges when dealing with complex interactions and contextual interpretations. In Chapter 4, we introduced knowledge modules (KMs) as specialised services and reasoning capabilities for the environment. However, we learned that relying solely on KMs to achieve higher levels of adaptation would only make the services more complex, monolithic, and less flexible.

To address this challenge, we propose Embedded Situations as a solution. Embedded Situations are labelled descriptions embedded in a smart product by the manufacturer that describe possible situations a product may encounter. A Situation describes a particular setting, semantically annotates it, describes a set of participants, roles, and conditions required to take place, and a condition or predicate that asserts the situation [37].

For example, consider a meeting room like the one described in Figure 6.5, with smart products such as a beamer and presentation software. Both products can have embedded different possible Embedded Situations. The beamer may present situations such as *{"watching movies","watching TV", "showing a slideshow", "showing a presentation"}*. In contrast, the presentation software may communicate *{"editing a presentation", "giving a talk", "brainstorming"}*

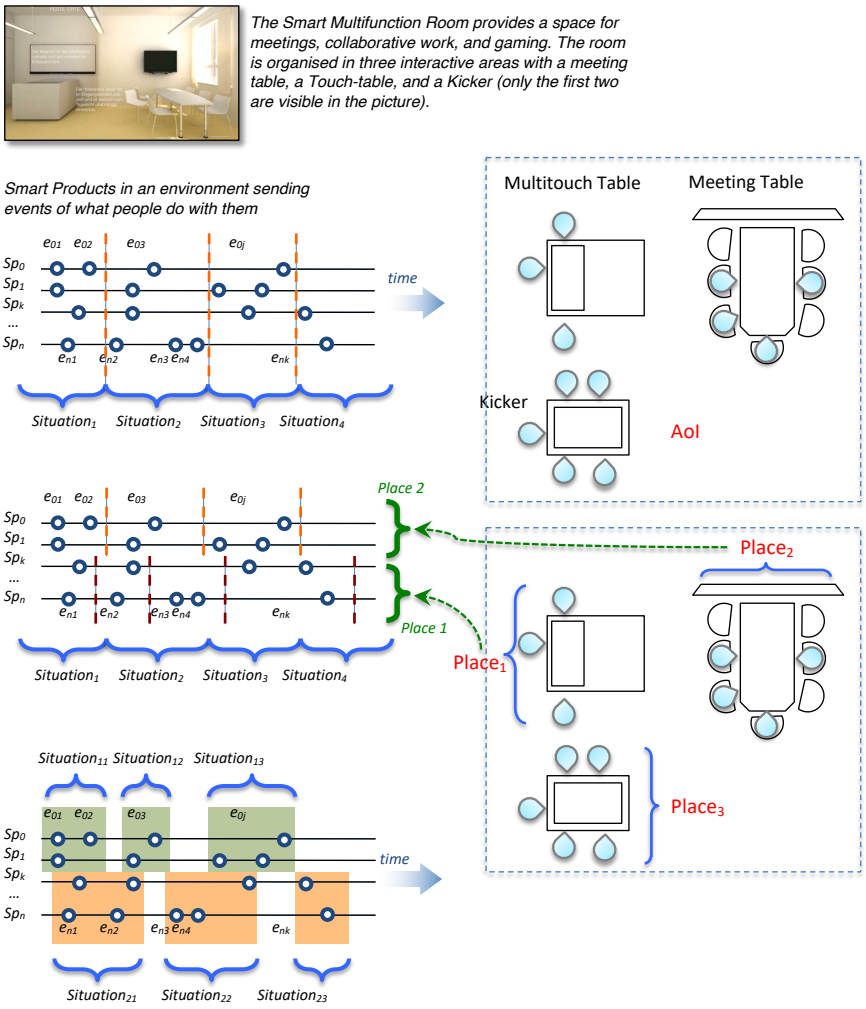


Figure 6.5: Smart Room Scenario with different activity areas.

By embedding Situations in smart products, we can improve the flexibility and adaptability of smart environments while reducing complexity. Furthermore, Embedded Situations can help smart products work together more seamlessly to provide a better overall experience for users.

Smart Product Embedded Situations formal definition:

$$
SP = \begin{cases}
(e_{00}, e_{01}, e_{02}, \ldots, e_{0k}) \text{ where } e_i \text{ are events emitted by SP,} \\
\\
\text{and } \textit{Embedded Situations} \text{ defined as:} \\
ES_{01}(S\_Name_{01}, WnetRef_{01}, Oref_{01}, e_{01}, \ldots, e_{01i}, \alpha_{01}, \ldots, \alpha_{0n}), \\
ES_{02}(S\_Name_{02}, WnetRef_{02}, Oref_{02}, e_{02}, \ldots, e_{02i}, \alpha_{02}, \ldots, \alpha_{0n}), \\
\ldots \\
ES_{0m}(S\_Name_{0m}, WnetRef_{0m}, Oref_{0m}, e_{0m}, \ldots, e_{0mi}, \alpha_{0m1}, \ldots, \alpha_{0mn}), \\
\\
\text{where:} \\
\textbf{S\_Name}_i \text{ is a label given to the situation,} \\
\textbf{WnetRef}_i \text{ is a reference to the label in Wordnet,} \\
\textbf{Oref}_i \text{ is an ontological reference describing the situation,} \\
\textbf{e}_i \text{ belongs to the known events emitted by the smart product,} \\
{}_{,i} \text{ is a context information feature}
\end{cases}
$$

As we add further products in the environment (for instance, a Kicker table), the possible set of situations defined also starts to grow. Another advantage of embedding the situation definitions within a product is that each independent vendor can dynamically add situations where any given Smart Product is used (e.g., through information gathered over social networks).

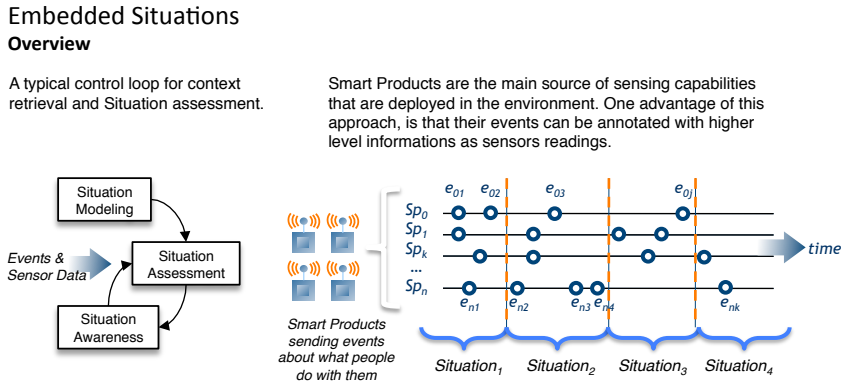


Figure 6.6: High level overview of the Embedded Situation assessment concept.

6.4.1 *Situation Assessment Architecture*

We have shown how Embedded Situations can provide descriptions and labels to what people do. The second step is to determine according to the information available what situation is taking place. The processes of analyzing and deciding the situation that is taking place a given environment is called *Situation Assessment*.

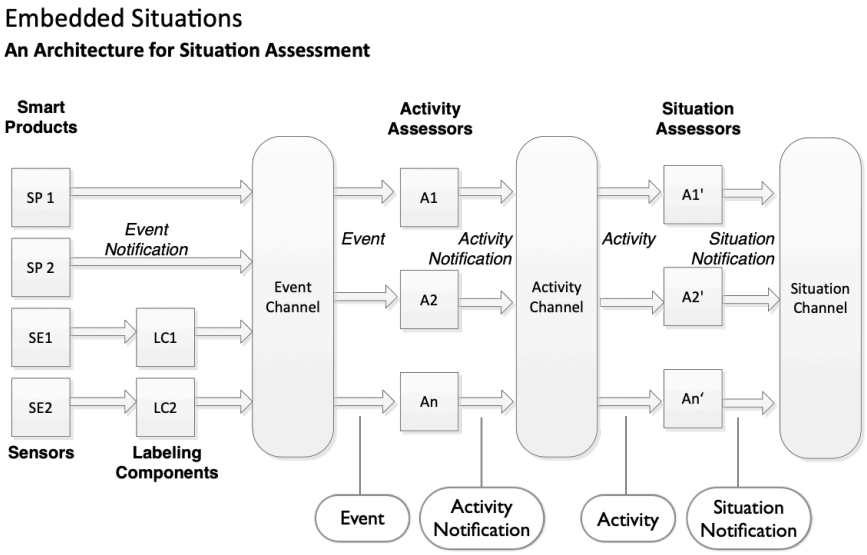


Figure 6.7: Embedded Situation assessment architecture.

6.4.2 *Requirements*

A number of requirements must be considered when designing and developing a situation model for smart products.

- Support of Event based Notifications
- Notification Persistence
- Asynchronous Assessment
- Assessment Scalability
- Support of Embedded Situations or Knowledge
- Support of Semantic Similarity

6.4.3 *Assessment Architecture*

The figure 6.7 above illustrates a layered approach to situation assessment. Utilising layers ensures consistency across various stages of evaluation. The layers are organised as follows:

1. Event Layer: This layer stores event notifications from smart products. Activity assessors from the subsequent layer query this layer for context entries.
2. Activity Layer: Assessors in this layer analyse context entries (events from smart products) and record inferred activities as context entries within the activity channel.

3. Situation Layer: This layer functions similarly to the previous layers.

These layers represent channels in a pub-sub system where notifications are preserved as context entries. Channels offer a centralised access point for storing context entries, enabling further processing by higher-level components. Under this scheme, each layer corresponds to a specific event channel:

1. *Event Channel*: This channel amasses events prompted by the environment's smart products, which publish their events here. Labelling components collect raw sensor data and publish events based on the received data. *Activity Assessors* subscribe to the event channel to access published events. Upon obtaining event data, they endeavor to identify specific activities, notifying the Activity Channel of any detections.

2. *Activity Channel*: This channel assembles context entries from activity assessors (activity evaluation) in a time series format. These context entries delineate activities inferred or detected by the activity assessors. Like the activity sensors in the preceding stage, *Situation Assessors* consume events published to the Activity Channel, determine the presence and nature of a particular situation, and notify the Situation Channel of their findings.

3. *Situation Channel*: This final channel houses context entries from situation assessors. Each context entry discloses a detected situation within the environment and notifies its subscribers accordingly.

6.5 INSITU: AN APPROACH FOR DYNAMIC CONTEXT LABELING

In this section, we present an example for recognising context situations in smart environments using the formalism for describing and sharing context states (or *situations*), and an architecture for gradually introducing contextual knowledge to an environment, where in this case, the current context is determined on sound analysis and the sensing of people's usage of devices.

One key aspect for making environments *smart* recognises the current context and how smart environments can cope with their purposes and composition changes. Furthermore, different users are often interested in different contextual situations. This makes it difficult to develop smart environments useful for many users. **InSitu** introduces the sound modality for detecting context.

One of the main challenges when constructing a user-adaptable smart environment is how to get labeled context data. Beside the obvious choice to let the users label the current situation, we propose to use context descriptions delivered within the Smart Products. These

context descriptions already provide rich information and are also used to label sound recorded with a microphone. When sufficient data has been labelled, it is possible to detect situations based on the sound alone, which helps to identify situations similar to those generated by the smart products.

6.5.1 *Natural language*

The concept of embedding situation descriptions in devices introduces some challenges. One of them is the semantic integration of the embedded information, that is often done using ontologies. *InSitu* supports ontological references in its situational descriptions. However, it becomes difficult to acquire and reason about situations in a smart environment where different smart products from different manufacturers may use different ontologies. Since each ontology represents an arbitrary set of concepts and relationships, complex mappings and processing known as ontology alignment are required, in order to match the semantics between different ontologies.

In order to cope with the semantical integration challenge, *InSitu* additionally implements a similar approach to ontologies grounded on natural language such as DOLCE [82]: we rely on Wordnet [151] and natural language processing as means to describe the label's meaning and to find semantic similarities between labels. In particular, *word similarity* and *word sense disambiguation* for the measurement of semantic similarity between situation descriptions. The first measures the similarity between words and the latter differentiates them.

When smart products are introduced in an environment, they first publish their ESs descriptions, and start communicating the events resulting from people's operation of a product. Both the ESs and events carry semantic information that needs to be disambiguated. Let's consider the case of events, that are instantiated as an *event entry (*Ee*)*. We define an *event entry* as:

$$
\begin{aligned}
\text{event entry} &= \langle\text{sourceId}\rangle, \langle\text{time}\rangle, \langle\text{content}\rangle \\
\langle\text{sourceId}\rangle &= \text{a id literal} \\
\langle\text{content}\rangle &= \langle\text{subject}\rangle\ \langle\text{predicate}\rangle \\
\langle\text{subject}\rangle &= \text{a string literal} \\
\langle\text{predicate}\rangle &= \langle\text{verb}\rangle\ \langle\text{object}\rangle \\
\langle\text{verb}\rangle &= \text{a verb in the english language} \\
\langle\text{object}\rangle &= \text{a noun in the english language.}
\end{aligned}
$$

Every event entry contains a unique identification about the sender (the notifying smart product), the timestamp when the notification was issued, and the content. The source and content are mandatory because the assessment depends on them.

The content of every event entry contains a sentence in natural language. The sentence describes context information, i. e., it is used to describe events, activities or situations. A developer can describe any event, activity or situation by writing a sentence. Each sentence contains a subject and a predicate. The subject refers to a noun and the predicate contains a verb and its arguments. The number of arguments depends on the valency of the verb, i. e., the number of arguments that the verb controls. For instance, the event for a printer with id 01at 12:00 pm which notifies that

"the printer turns on"

is expressed as the event entry

$$(01, 12{:}00 \text{ pm}, (\text{printer}, (\text{turns on}, -))).$$

We use the WordNet dictionary to handle semantically similar event entry contents. This way, the event entry can also be understood if both the subject and predicate are written using synonyms. In order to quantify the similarity between words, we use the similarity measure by Wu and Palmer (1994) that expresses the semantic similarity d_{WP} between two concepts (words) c_1 and c_2 in as:

$$d_{WP}(c_1, c_2) = \frac{2C}{A + B + 2C},$$

where

$$\begin{aligned} A &= \text{length}(c_1, \text{lcs}(c_1, c_2)) \\ B &= \text{length}(c_2, \text{lcs}(c_1, c_2)) \\ C &= \text{depth}(\text{lcs}(c_1, c_2)) \\ \text{length} &: \text{the shortest path between words} \\ \text{depth} &: \text{distance to root} \end{aligned}$$

The lowest common subsumer lcs is the nearest common parent between c_1 and c_2 to which both c_1 and c_2 are a hyponymy. The length function is the shortest distance between two words and the depth function refers to the distance of a word to the root.

In a similar way to event entries, we now define *Embedded Situations (ES)*:
let $\{e_{00}, e_{01}, e_{02}, \ldots, e_{0k}\}$ where e_{0i} are events entries emitted by a Smart Product SP_0, an *Embedded Situation (ES)* defined as:

$$\begin{aligned} ES_0 &= \langle \text{Name} \rangle, \langle \text{WnetRef} \rangle, \langle \text{Oref} \rangle, \langle e_1 \ldots e_k \rangle \\ \langle \text{Name} \rangle &= \text{is a label given to the situation} \\ \langle \text{WnetRef} \rangle &= \text{is a reference to the label in Wordnet} \\ \langle \text{Oref} \rangle &= \text{ontological reference describing the situation} \\ \langle e_{01} \ldots e_{0k} \rangle &= \text{known event entries emitted by the product} \end{aligned}$$

6.5.2 *Architecture*

The InSitu applies the Embedded Situations eventing architecture (see Figure 6.7) and provides a scheme for assessing situations based on the information and events gathered by the smart products available. The overall architecture is organised in a three-stage pipeline that successively refines and interprets the simple event entries into activities first and into situations afterwards.

As described in section 6.5.1, each smart product publishes events representing data and state changes in the form of *event entries*. State changes can be triggered through user actions on the product's operations, while data changes inform the product's varying sensor readings.

The event entries broadcasted by the smart products are queried and analysed by activity assessors modules that interpret the event notifications into activities and publish activity notifications to an activity channel. Situation assessors query the published activity notifications from the activity channel and reason about them to determine the most likely situation.

At each stage of event interpretation, the architecture allows for different techniques to analyse the data and propose a possible candidate state. When two or more competing states have been found, a coordinating Assessor Manager implements a voting scheme for choosing the best candidate state before publishing it.

Once the different assessors publish their candidate situations to the situation channel, the candidate states are subject to a voting scheme to decide the most likely state. Such candidate situations are subject to a voting scheme between different context assessors to determine the most likely situation and publish it to the situation notification channel.

In this architecture, we refer with the name *channel* to a publish-subscribe networking mechanism such as [16]. With pub-sub, producers and consumers can freely publish and consume the events being generated through named channels (*Event*, *Activity* and *Situation* channels) as an alternative to point-to-point communications. The InSitu approach can also be implemented with other technologies like Apache Kafka [209] as the underlying pub-sub messaging system.

InSitu enables growing the known context state labels to an environment by sharing ES information in the products. However, since the number of contexts where a device make take part can be arbitrarily large, devices understandably cannot assume to have a comprehensive set of Embedded Situations. Therefore, users in the environments can name their current situation/context to enrich the catalogue further.

6.5.3 *Sensing Sound in smart environments*

A variety of possibilities exist to sense context the environment in order to recognize activities: wearable sensors [230], sensors in mobile phones [38], sensors in tools [165], interactions with objects [166], microphones [112], etc. The main disadvantage of wearable sensors is that they are obtrusive so that users are typically not willing to wear them in order to get implicit interaction. Users are accustomed to bring their mobile phones with them so that it is sensible to use the sensors built into these devices for activity recognition. There are two drawbacks to activity recognition with mobile phones:

- For certain activities, such as kitchen activities, the users may not always have their mobile phone with them.
- Since the mobile phone is typically placed in the pocket of the user, the movements sensed by the inertial sensors are typically not as informative as sensor signals from tracking the part of the body involved with the execution of the activity.

Using tools with embedded sensors circumvents these two drawbacks. However, it is not always realistic to equip all relevant tools with sensing due to cost constraints.

6.5.4 *Partially instrumented scenario*

In addition to Embedded Situations we propose to use sound as the an auxiliary modality. Sound is used to detect situations that are not handled by the Embedded Situations. In InSitu, there are two use cases for using sound as context source, which we call ”generalization” and "partial instrumentation”.

6.5.5 *Partial Instrumentation*

Partial instrumentation describes scenarios where only a few of the available products are smart. Using Embedded Situations, the smart product provides a label to the situation while the microphone records the sounds that are generated. Performing the training on-site has the advantage that particular properties of the room (acoustical properties, placement of the microphone, ambient sounds) are reflected in the training set, which may lead to better recognition results.

As an example, let us consider the recognition of kitchen activities, which may be used to support the user in the kitchen and to recommend recipes based on the user’s expertise. Two particular kitchen activities are the slicing (of bread) and the chopping (of vegetables). In addition to the normal knives that the user already possesses, the user gets an additional knife equipped with sensors, say accelerometers

and force transducers. Using the sensor signals, slicing and chopping is distinguished. A microphone in the kitchen records the slicing respectively chopping sounds and labels the situation according to the sensor-based recognition results. When sufficient data has been collected, chopping and slicing can be detected based on the sound only, so that the distinction does not depend on the sensor signals any more. In that way only a single knife has to be equipped with sensors, which helps to reduce the complexity and the cost of the smart kitchen environment.

6.5.6 *Generalization*

Some situations are not characterized by the use of a smart product. Consider, e.g.: a group of colleges in a meeting room discussing and taking notes with pen and paper. In such conditions Embedded Situations are obviously not useful to detect context. Based on the microphone input, however, it is possible to identify similar situations. E.g.: the smart environment may detect that the current situation is similar to a presentation situation for which a sufficiently large data set was labeled using the Embedded Situations in the beamer and the presentation application. Detecting a discussion as "a situation similar to a presentation" may, e.g., to signal that the room is occupied.

6.5.7 *Audio Processing*

Mitrovic & Eidenberger have examined a wide range of popular audio features in order to determine an optimal set of features for organizing environmental sounds [152]. Since it is not possible to know what situations are relevant to the users, we use the features as suggested by Mitrovic & Eidenberger. Novelty detection techniques [146] can then be used to separate known situations from novel sounds.

6.5.8 *Auditory Context Recognition*

Various works have recognized context using sound as the only data source:E.g., Eronen et al. detect context from the sound that are encountered by city inhabitants such as being at a restaurant, listening to music, etc. [75]; Franke et al. recognize the sounds from household devices using the microphone in a mobile phone [79]; Zhan et al. detect everyday human activities [228]. Furthermore, some works use a combination of sound and other sensing capabilities to achieve better recognition results:E.g., Lukowicz et al. use body worn accelerometers in combination with sound to detect activities in a wood workshop [128]. This is different to our approach outlined in Section 6.5.3 where we propose that a smart product with internal sensing capabilities provides a labeling of the recorded sound, which is used to train a

sound analysis component. This makes it possible to detect context in partially instrumented settings and to detect context that is similar to the context provided by the smart products.

6.5.8.1 *Ruleset Based Assessment*

The assessment is based on the verification of a set of rules. A rule comprises a body and a head. The body of a rule contains the conditions that must be fulfilled in order for the head to be valid. The assessment is completely based on the evaluation of context entries against some set of rules.
The following pseudoalgorithm describes the assessment.

1. Query a set of context entries from the layer channel
2. Evaluate all conditions for every rule in the rule set.
3. Output matched assessment results

The knowledge of the assessors using this type of assessment basically consists of the rule sets.

6.5.8.2 *Keyword Based Assessment*

This assessment method is based on text and web mining techniques that rely on the keywords found in a document. A document is a set of words that describes an idea. Keyword based assessment is a specific type of rule based assessment, where words are the conditions of a rule. The assessment takes place by reading the words in the context entries and verifying whether any keyword is present.
A pseudoalgorithm of the keyword based assessment is presented in the following snippet:

1. Query a set of context entries from the layer channel
2. Extract the words from the content of each context entry
3. Sort the words by their frequency
4. Optionally remove stop words
5. Search for keywords
6. Output matched assessment results

6.5.8.3 *Bayesian Based Assessment*

Bayesian networks are often used when dealing with uncertainty in situation assessment. A bayesian network allows a graphical definition of the probability distributions across several variables.

For a bayesian network based assessor in the activity layer, we can define a bayesian network for each activity assessment the following way:

Variables $X_1, X_2, \ldots, X_n$ that describe the state of each smart product.

A boolean variable A that represents the probability distribution of the activity (only true or false).

The activity variable A is connected to the all Xn.

Given that X_n may have a more complex probability distribution, each Xn itself can be seen as a sub-bayesian network.

The probability distribution can be calculated from a histogram from the event channel.

Querying of the form: $P(A_1 = true | X_1 = x_1, X_2 = x_2, \ldots, X_n = x_n)$, i.e. the probability of Activity 1 happening after the events $x_1, x_2, \ldots, x_n$ happened can be calculated as $P(A_1 = true) * P(x_1|A_1) * P(x_2|A_1) * \ldots * P(x_n|A_1)$

By using Bayesian Networks arises the following difficulties:

- In the above described setting, Xn may have many other activity variables Sn as parent nodes in the bayesian network. This can cause very complex probability distributions in the Xn nodes. A way to simplify this would be to create several bayesian networks, one for each activity that needs to be assessed. Then, the resulting detected activity comes from the bayesian network that came out with the highest probability.

- The creation of the bayesian network may only come from an expert. However, this "expert" can be the Smart Product itself, since it may contain embedded knowledge. In this case, the name of the Activity Node is given by the Activity label, and all otherX_n can be constructed by parsing the described events of the embedded knowledge (using any ontological or wordnet reference).

6.5.8.4 *HMM Based Assessment*

Hidden Markov Models (HMM) are very often used in activity recognition. This model is useful for detecting data or state sequences that are hidden, but their output is observable. The output is also called Observation.

Therefore, a HMM consists of hidden states, an observation set, transition probabilities between states, emission probabilities of the observation set and start probabilities.

Given a HMM as the knowledge of an assessor, the assessment takes place as follows:

1. Query a set of context entries from the layer channel
2. If not yet done, sort the context entries by timestamp

3. Perform Viberti algorithm to find most likely sequence of states
4. Output matched assessment results in accordance to the calculated sequence of states

6.5.8.5 *Assessment Voting*

As many assessors can be involved during assessment in a specific layer, conflicts can occur when assessment results differ from each other. Therefore, a voting mechanism is necessary to make a final decision.

The voting mechanism used in this framework calculates a score for assessment result. Each score is the weighted sum of the count of the same assessment result. The weight of each assessor may increase or decrease based on the reliability of its results, i.e., an assessor, whose assessment results are frequently correct, has higher weight over other assessors with less reliable results.

Voting quality directly depends on (user) feedback since weights of the assessors are proportional to the number of correct assessments.

6.6 CONCLUSION

This chapter presents Embedded Preferences, Embedded Situations and InSitu to allow the development of higher abstraction levels of smart functionality. The core novelty of these concepts and approaches is to deploy the knowledge of Preferences and Situations into the Smart Products and Smart Services, instead of becoming part of a central management software.

Embedded Preferences borrows the technique developed in Chapter 4 when developing an indoor navigation system that can adapt to the context and the user's preferences and applies an improved version to modelling information about the products and their usage. The notion of *Preference* is also significant, as it emphasises their "soft" nature instead of a more rigid rule set and more suitable for scenarios with multiple smart products taking part where conflicts among the diverse preferences of different products are very likely to emerge, providing a solid mechanism for solving such conflicts and making a decision.

InSitu is an approach to recognising context based on the usage of smart products and sound analysis. It allows a smart environment to evolve as more smart products are introduced. We propose that the manufacturer provides the smart product with Embedded Situations for that purpose. These Embedded Situations describe how the product can be used and define the rules to recognise these situations. Furthermore, the Embedded Situations are available to the smart environment via a middleware solution. When a smart product is used, the smart environment is informed of the user-device interaction via events sent by the device. The smart environment can recognise con-

text by examining the event reported by the smart products. This includes the context deduced from events that originate from multiple smart products, e.g., recognising that the users are in a home cinema context when the computer (smart product 1) shows a video in full screen on a beamer (smart product 2). Since the events from different manufacturers are used to recognise context, the "vocabularies" used in the Embedded Situations by different manufacturers must be matched. For this purpose, we propose using a simplified form of natural language to describe the Embedded Situations, which is then matched using word similarity calculations based on WordNet. Furthermore, ontologies can be used to concretise the meaning further if necessary. The context is then recognised using keyword-based and rule-based assessments.

Sound analysis is used as an additional means to assess context. A microphone placed in the smart environment records the sounds when the smart products are used. With the help of Embedded Situations, the sounds are labelled and used to train an auditory context recognition component. This allows for enlarging the scope for context recognition in two ways: First, due to cost reasons, only a subset of the products of the same kind used in a specific situation may be a smart product, equipped with sensing and communication abilities. Auditory context recognition can detect the user's context even if she uses one of the ordinary products. Second, in some situations, no smart products are used. Yet, the situations may be similar to those that are detectable with the Embedded Situations mechanism and can be labelled as such using auditory context recognition.

In conclusion, InSitu allows the smart environment to evolve as more smart products with Embedded Situations are integrated into the environment. Furthermore, enables extending the system's coverage to partially instrumented environments and detecting situations similar to those covered by the Embedded Situations but not characterised by smart product use.

7

CONCLUSION AND OUTLOOK

7.1 CONTRIBUTIONS

This thesis has presented different approaches, technologies, algorithms, data representations and architectures aiming at making smart environments happen everywhere without a centralised design, development and integration approach.

Chapter Two begins with a brief history of the evolution of the smart environments and objects that we call products generically and provides an overview of state of art on the essential technical aspects relevant to the field, a classification of the key criteria and a comparative evaluation.

In addition, interviews were carried out with respected authorities in the field, like Prof. Dr Paul Dourish from the university of California in Irvine. His work as an anthropologist explored many aspects of how people relate to technologies and how humans define the meaning of places by the objects they put in them. Another scientist interviewed was Dr Norbert Streitz, who pioneered the concept of "Ambient intelligence" with this renowned Agoras Project and the notion of the subtle automated assistive agency embedded in the background of living spaces. Finally, in a third interview, Prof. Lucy Suchman from Lancaster University studied different aspects of people's relationships and technology usage. This led to the first experience introducing assistive technologies for end-users at Xerox in the early '70s. She noticed that innovative assistive technologies are the means today's humans delegate some of the chores people have historically considered less valuable.

Interestingly, many of the needs addressed today are the same but with different technologies: before were slaves and servants, and today, smartphones and robots. The significance of intelligent environments goes beyond the inter-networking of existing infrastructures with new sensors and devices for gathering and combining data. Still, people's needs are being signified and addressed in a new way.

Chapter Three presents a novel approach for building Open Smart Environments (OSE) based on People, Smart Products, Space and Active Knowledge. On the people model, this thesis contributed to creating an enriched model that integrates a preference model that can be used for computation in different contexts using a Multi-attribute Utility Theory (MAUT) representation. It also introduced the physical Capabilities after the World Health Organization (WHO): the Interna-

tional Classification of Functioning, Disability and Health (ICF). Last, we introduced a model of actions that a person can do. It is expressed in the same procedural terms that AI-STRIPS planners can operate on (Actions, Requirements and Post-conditions) as if they were functions available for other entities in the environment. The person's abilities and things can now be integrated computationally to create a plan and guidance to achieve a goal.

Another contribution is the insight that, if we consider smartness from a product lifecycle perspective, the current vision of smart environments is limited to a single stage of the whole lifecycle: *user operation*. This observation discovers many opportunities for introducing smartness that hasn't been the focus of the main research. People do not only *use* products: people *buy* them, *install* them, *repair* and *dispose* them. Furthermore, other people further encounter them while *designing* them, *producing*, *storing*, *transporting*, *distributing* and *selling* them as well.

Chapter Four introduces Smart Products, the main building block for OSEs, together with two examples: a smart coffee machine and smart shelf, with a first application of a knowledge module to implement an estimation algorithm based on Diophantine equations. Chapter three introduced the idea of supporting smart behaviours in different lifecycle stages. In this chapter, an example is introduced with an experiment called M2IT for enriching the shopping of a product experience. Another prototype introduced is called Ubiquitous explanations exploring the introduction of adaptive explanations to People with different skill levels and supporting the configuration stage of a product lifecycle. This prototype also served to document several different voice interaction design patterns that have been published but are not part of this thesis.

Smart products integrate a MAPE (Monitor, Analyze, Plan and Execute) cycle, where a STRIPS planner combines the monitored events with the available actions and creates a plan. Such a plan needs to be carried out and is implemented in the form of a minimal workflow engine in java: the Flexi workflow engine.

The ability to compose products depends on the ability of devices to discover and interoperate among themselves. In this context, the chapter *Mundo Nubes* introduces a lightweight Pub-Hub architecture for devices and services to share and access information across different network and protocol technologies. Mundo Nubes borrows and extends some ideas of MQTT to simplify the acquisition and reuse of data and interoperability from shared devices using standard Web technologies.

Chapter Five introduces models for people, space and guiding people moving across that space through an indoor navigation proto-

type called CoiNS. The work done in Coins brought two significant contributions to indoor navigation: first, a hybrid space model that introduces a novel representation called a Hierarchical Graph to achieve a path to a destination with an order of execution significantly better than with the Dijkstra Algorithm. The second contribution is the combination of the user preferences developed in chapter three using Multi-attribute Utility Theory (MAUT) to select the most appropriate destination path.

Chapter Six introduces several concepts to extend Open Smart Environment Scenarios to different product cycle stages, as well as higher levels of context-awareness and adaptation capabilities. First, *Embedded Preferences for Products (EP4P)* are presented, where its novel contribution is the application of the MAUT-based preferences *for products* to facilitate smart interactions between Smart Products throughout their entire life cycle, as proposed in chapter 3.

A sample scenario built on EP4P called *"Compara"* demonstrates a potential application of *Embedded Preferences* with a novel OSE scenario in a mobile shopping setting. The scenario demonstrates a prototype and introduces novel algorithms to combine the MAUT-based preferences applied to the location of a product in a physical space.

Afterwards, a granular model for assessing and adapting to higher abstraction levels of context-awareness called *Embedded Situations* is presented. The notion of *Situations* describes more complex context settings in an environment and are required to enable OSEs to achieve better adaptation and anticipation to their inhabitants. Finally, the chapter introduces the concepts of *Areas of Influence* and *Places* to support multiple concurrent activities in an environment.

7.2 DIRECTIONS FOR FUTURE RESEARCH

This thesis investigated open smart environments. However, it yields interesting open challenges that researchers may address in the future.

Early IoT devices gathered and communicated data from the physically sensed world into public or private clouds. As IoT platforms continue evolving, different operating systems and hardware configurations makes it complex to keep applications maintained, updated, and secured. Some transcoding approaches such as ThingsML and custom DSL (domain languages) were the first attempts to manage the complexity through transcoding across platforms. However, the high level of automation comes at a heavy price when the underlying application model supported must be customized, relegating these technologies to more simply the sense-and-communicate generation of IoT.

Current advances in containerisation will help manage this complexity by maintaining the integrity of the underlying system without the

constraints that transcoding toolchains have. Containers will enable IoT developers to easily create complex systems while maintaining scalability, security, and portability, and as a consequence, will spin Fog Computing into reality.

New Fog Computing architectures should go beyond the current single-purpose use of data to a scenario where sensed data is reused and processed for different applications. Such capabilities will multiply the value of future edge computing infrastructures and accelerate the deployment of new smart environment scenarios.

BIBLIOGRAPHY

[1] http://www.amigo-project.org.

[2] 2021. URL: https://www.vaisala.com/.

[3] 2021. URL: https://www.cleverciti.com/.

[4] 2021. URL: https://www.jasper.com.

[5] 2021. URL: https://www.ibm.com/watson.

[6] 2021. URL: https://www.apple.com/lae/ios/ios-14/.

[7] 2021. URL: https://developers.google.com/weave.

[8] 2021. URL: https://developers.google.com/brillo.

[9] 2021. URL: https://developer.android.com/things.

[10] 2021. URL: https://azure.microsoft.com/en-us/get-started/iot/.

[11] 2021. URL: https://developer.qualcomm.com/software/alljoyn.

[12] 2021. URL: https://www.iottechtrends.com/what-happened-google-brillo-weave/.

[13] Donald A. Norman. *Emotional Design: Why We Love (or Hate) Everyday Things*. 1st ed. Basic Books, May 2005. ISBN: 0465051367. URL: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0465051367.

[14] M. Addlesee, R. Curwen, S. Hodges, J. Newman, P. Steggles, and A. Hopper A. Ward. "Implementing a Sentient Computing System." In: *IEEE Computer*. Vol. 34. IEEE, Aug. 2001, pp. 50–56.

[15] Marco Aiello and Ilche Georgievski. "Planning for Pervasive Systems." In: (2018).

[16] Erwin Aitenbichler. "System Support for Ubiquitous Computing." PhD thesis. Darmstadt University of Technology, 2006. ISBN: 3-8322-5365-3.

[17] Erwin Aitenbichler, Jussi Kangasharju, and Max Mühlhäuser. "MundoCore: A Light-weight Infrastructure for Pervasive Computing." In: *Pervasive and Mobile Computing* 3.4 (Aug. 2007). doi:10.1016/j.pmcj.2007.04.002, pp. 332–361. ISSN: 1574-1192.

[18] Erwin Aitenbichler, Fernando Lyardet, Gerhard Austaller, Jussi Kangasharju, and M. Mühlhäuser. "Engineering intuitive and self-explanatory smart products." In: *SAC '07*. 2007.

[19] Erwin Aitenbichler, Fernando Lyardet, Gerhard Austaller, Jussi Kangasharju, and Max Mühlhäuser. "Engineering Intuitive and Self-Explanatory Smart Products." In: *Proceedings of the 22nd Annual ACM Symposium on Applied Computing*. ACM Press, 2007, pp. 1632–1637. ISBN: 1-59593-480-4.

[20] Erwin Aitenbichler, Fernando Lyardet, Aristotelis Hadjakos, and M. Mühlhäuser. "Fine-Grained Evaluation of Local Positioning Systems for Specific Target Applications." In: *UIC*. 2009.

[21] Erwin Aitenbichler, Fernando Lyardet, Aristotelis Hadjakos, and Max Mühlhäuser. "Fine-grained Evaluation of Local Positioning Systems for Specific Target Applications." In: *Proceedings of the 6th International Conference on Ubiquitous Intelligence and Computing (UIC-09)*. Springer, July 2009.

[22] Erwin Aitenbichler, Fernando Lyardet, and Max Mühlhäuser. "Diseno e Implementacion de Espacios Inteligentes." In: *Cepis Upgrade* 23.188 (July 2007), pp. 29–34.

[23] Ala Al-Fuqaha, Mohsen Guizani, Mehdi Mohammadi, Mohammed Aledhari, and Moussa Ayyash. "Internet of things: A survey on enabling technologies, protocols, and applications." In: *IEEE communications surveys & tutorials* 17.4 (2015), pp. 2347–2376.

[24] Ala Al-Fuqaha, Abdallah Khreishah, Mohsen Guizani, Ammar Rayes, and Mehdi Mohammadi. "Toward better horizontal integration among IoT services." In: *IEEE Communications Magazine* 53.9 (2015), pp. 72–79.

[25] *Android Things restricted availability to partner OEMs*. 2021. URL: https://developer.android.com/things/faq.

[26] Stavros Antifakos, Florian Michahelles, and Bernt Schiele. "Proactive Instructions for Furniture Assembly." In: *Ubicomp*. 2002, pp. 351–360.

[27] Stephan Arneth. "Ubiquitous Explanations." MA thesis. TU Darmstadt, 2010.

[28] W. Ross Ashby. "Principles of the self-organizing system." In: *Principles of Self-Organization: Transactions of the University of Illinois Symposium* 1 (1962), pp. 255–278.

[29] C. M. A. Ashruf. "Thin flexible pressure sensors." In: *Sensor Review* 22 (2002), pp. 322–327.

[30] Kevin Ashton. "That 'Internet of Things' Thing." In: *RFID Journal* (July 2009). URL: https://www.rfidjournal.com/that-internet-of-things-thing.

[31] Kevin Ashton et al. "That 'internet of things' thing." In: *RFID journal* 22.7 (2009), pp. 97–114.

[32] Autodesk. *Gmax*. http://www.autodesk.com/gmax. last visited: 03.07.2008. 2008.

[33] Soma Bandyopadhyay, Munmun Sengupta, Souvik Maiti, and Subhajit Dutta. "A survey of middleware for internet of things." In: *Recent trends in wireless and mobile networks*. Springer, 2011, pp. 288–296.

[34] Sharu Bansal and Dilip Kumar. "IoT ecosystem: A survey on devices, gateways, operating systems, middleware and communication." In: *International Journal of Wireless Information Networks* (2020), pp. 1–25.

[35] Martin Becker. "Software Architecture Trends and Promising Technology for Ambient Assisted Living Systems." In: *Assisted Living Systems - Models, Architectures and Engineering Approaches*. 2007.

[36] Scott Boag, Don Chamberlin, Mary F. Fernandez, Daniela Florescu, Jonathan Robie, and Jérôme Siméon. *XQuery 1.0: An XML Query Language*. W3C Recommendation. World Wide Web Consortium (W3C), 2007. URL: https://www.w3.org/TR/xquery/.

[37] Oliver Brdiczka, James L. Crowley, and Patrick Reignier. "Learning Situation Models for Providing Context-Aware Services." In: *HCI*. 2007.

[38] T. Brezmes, J.L. Gorricho, and J. Cotrina. "Activity recognition from accelerometer data on a mobile phone." In: *Distributed Computing, Artificial Intelligence, Bioinformatics, Soft Computing, and Ambient Assisted Living* (2009), pp. 796–799.

[39] B. Brumitt, B. Meyers, J. Krumm, A. Kern, and S. Shafer. "EasyLiving: Technologies for Intelligent Environments." In: *Proc. of 2nd International Symposium on Handheld and Ubiquitous Computing (HUC 2000)*. Springer, Sept. 2000, pp. 12–27.

[40] Joel Brynielsson and Klas Wallenius. "A toolbox for multi-attribute decision-making." In: 2003.

[41] Andreas Butz, Jörg Baus, Antonio Krüger, and Marco Lohse. "A Hybrid Indoor Navigation System." In: *IUI2001: International Conference on Intelligent User Interfaces*. New York: ACM, 2001. URL: http://w5.cs.uni-sb.de/~butz/publications/papers/hybrid-iui.ps.gz.

[42] J.M. Carroll and M.B. Rosson. *The Paradox of the Active User*. MIT Press, 1987.

[43] Timothy Casey, Stuart Cheshire, William Fenner, Edwin Guttman, Mark Handley, John Hawkinson, Michael Jensen, Bartek Kolar, Sanjay Kumar, Daniel Lynch, et al. "Simple Service Discovery Protocol." In: *Internet Engineering Task Force*. 2001.

[44] Chia-Hung Chen, Alan Liu, and Pei-Chuan Zhou. "Controlling a service robot in a smart home with behavior planning and learning." In: *2014 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE. 2014, pp. 2821–2826.

[45] J. Chen, A. H. Kam, J. Zhang, N. Liu, and L. Shue. "Bathroom Activity Monitoring Based on Sound." In: *Proceedings of Pervasive Computing*. Munich, Germany, 2005.

[46] S. Cheshire, B. Aboba, and E. Guttman. *RFC 3927 - Dynamic Configuration of IPv4 Link-Local Addresses*. https://tools.ietf.org/html/rfc3927. Accessed: 2023-02-15. May 2005.

[47] S. Cheshire and M. Krochmal. *RFC 6762 - Multicast DNS*. https://tools.ietf.org/html/rfc6762. Accessed: 2023-02-15. Feb. 2013.

[48] S. Cheshire and M. Krochmal. *RFC 6763 - DNS-Based Service Discovery*. https://tools.ietf.org/html/rfc6763. Accessed: 2023-02-15. Feb. 2013.

[49] Stuart Cheshire and Daniel Steinberg. "Zero configuration networking: the Definitive Guide." In: *O'Reilly Media, Inc.* 2008. ISBN: 9780596101008. URL: https://www.oreilly.com/library/view/zero-configuration-networking/9780596101008/.

[50] Workflow Management Coalition. *XPDL 2.2 - XML Process Definition Language*. Version 2.2. 2005. URL: http://www.wfmc.org/standards/docs/TC-1025_v1.0.x.pdf.

[51] *Coexistence, Interoperability, and Other Terms*. 1999. URL: http://grouper.ieee.org/groups/802/15/pub/1999/Nov99/99114r0P802-15_Coexistance-Interoperability-and-other-Terms.doc.

[52] Michael H. Cohen, James P. Giangola, and Jennifer Balogh. *Voice User Interface Design*. Boston: Addison-Wesley, Jan. 2004.

[53] HR-XML Consortium. *Human Resources XML Vocabularies*. June 2011. URL: http://www.hr-xml.org/.

[54] YAWL Consortium. *Yet Another Workflow Language*. URL: http://www.yawlfoundation.org/.

[55] Evelyne Contejean and Hervé Devie. "A New Incremental Algorithm for Solving Systems of Linear Diophantine Equations." In: *Information and Computation* 113.1 (1994), pp. 143–172.

[56] Diane J. Cook, Juan C. Augusto, and Vikramaditya R. Jakkula. "Ambient intelligence: Technologies, applications, and opportunities." In: *Pervasive and Mobile Computing* 5.4 (2009), pp. 277–298.

[57] Diane Cook and Sajal Das. *Smart Environments: Technology, Protocols and Applications (Wiley Series on Parallel and Distributed Computing)*. Wiley-Interscience, 2004. ISBN: 0471544485.

[58] J.R. Cooperstock, S. S. Fels, W. Buxton, and K. C. Smith. "Reactive Environments: Throwing Away Your Keyboard and Mouse." In: *Comm of the ACM*. 53 7. IEEE, Sept. 1997, pp. 50–56.

[59] Daniel Corsten and Thomas Gruen. "Desperately seeking shelf availability: an examination of the extent, the causes, and the efforts to address retail out-of-stocks." In: *International Journal of Retail & Distribution Management* (2003).

[60] Daniel Cricco. "Gondor: A Sensor Network Infrastructure." MA thesis. Universidad Catolica de Asuncion, 2007.

[61] Li Da Xu, Wu He, and Shancang Li. "Internet of things in industries: A survey." In: *IEEE Transactions on industrial informatics* 10.4 (2014), pp. 2233–2243.

[62] Jessie Dedecker, Tom Van Cutsem, Stijn Mostinckx, Theo D'Hondt, and Wolfgang De Meuter. "Ambient-Oriented Programming in AmbientTalk." In: *ECOOP*. Ed. by Dave Thomas. Vol. 4067. Lecture Notes in Computer Science. Springer, 2006, pp. 230–254. ISBN: 3-540-35726-2.

[63] Matthias Dehmer and Frank Emmert-Streib. "Structural similarity of directed universal hierarchical graphs: A low computational complexity approach." In: *Applied Mathematics and Computation* 194 (2007), pp. 7–20.

[64] Deutsches Institut für Normung e.V. *Norm 8414: Benutzerinformation - Hinweise für die Erstellung*. Feb. 1988.

[65] Anind K. Dey and Gregory D. Abowd. "CybreMinder: A Context-Aware System for Supporting Reminders." In: *Proceedings of the Handheld and Ubiquitous Computing, Second International Symposium, HUC 2000*. Vol. 1927. Lecture Notes in Computer Science. Springer, Sept. 2000, pp. 172–186.

[66] Anind Kumar Dey. "Providing architectural support for building context aware applications." Director-Gregory D. Abowd. PhD thesis. Georgia Institute of Technology, 2000.

[67] E. W. Dijkstra. "A note on two problems in connection with graphs." In: *Numerische Mathematik* 9.51 (1950), pp. 161–166. ISSN: 1046-8188.

[68] Walt Disney. 1966. URL: http://disneyworld.disney.go.com/parks/epcot/.

[69] Paul Dourish and Genevieve Bell. "The infrastructure of experience and the experience of infrastructure: meaning and structure in everyday encounters with space." In: *Environment and Planning B: Planning and Design* 34.3 (2007), pp. 414–430.

[70] R. Droms. *RFC 2131 - Dynamic Host Configuration Protocol*. https://tools.ietf.org/html/rfc2131. Accessed: 2023-02-15. Mar. 1997.

[71] W. Edwards and D. von Winterfeldt. *Decision Analysis And Behavioral Research*. Cambridge University Press, 1986. ISBN: 052125308X.

[72] Nwanua Elumeze and Michael Eisenberg. "Towards Ambient Programming for Children." In: *Proceedings of the IADIS International Conference on Cognition and Exploratory Learning in Digital Age (CELDA'05)*, 2005, pp. 230–237.

[73] Frank Emmert-Streib and Matthias Dehmer. "Information theoretic measures of UHG graphs with low computational complexity." In: *Applied Mathematics and Computation* 9 (2007), pp. 1783–1794.

[74] J.L. Encarnacao and T. Kirste. *Ambient Intelligence: Towards Smart Appliance En-sembles*. Vol. LNCS 3379. Springer, 2005.

[75] A.J. Eronen, V.T. Peltonen, J.T. Tuomi, A.P. Klapuri, S. Fagerlund, T. Sorsa, G. Lorho, and J. Huopaniemi. "Audio-based context recognition." In: *Audio, Speech, and Language Processing, IEEE Transactions on* 14.1 (2006), pp. 321–329. ISSN: 1558-7916.

[76] Patrick Th. Eugster, Pascal A. Felber, Rachid Guerraoui, and Anne-Marie Kermarrec. "The many faces of publish/subscribe." In: *ACM Computing Surveys*. Vol. 35. 2. 2003, pp. 114–131. DOI: 10.1145/857076.857078.

[77] Siegemund F. "A Context-Aware Communication Platform for Smart Objects." In: *Proceedings of the 2nd Int. Conference on Pervasive Computing*. 2004.

[78] Richard Fikes and Nils J. Nilsson. "STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving." In: *International Joint Conference on Artificial Intelligence*. 1971.

[79] T. Franke, P. Lukowicz, K. Kunze, and D. Bannach. "Can a Mobile Phone in a Pocket Reliably Recognize Ambient Sounds?" In: *2009 International Symposium on Wearable Computers*. IEEE. 2009, pp. 161–162.

[80] D. Franklin and K. Hammond. "The intelligent classroom: providing competent assistance." In: *Proc. fifth Int'l Conf. on Autonomous Agents 2001*. ACM Press, Apr. 2001, pp. 158 –175.

[81] Fraunhofer. *Messe Navigator*. http://www.iis.fraunhofer.de/ec/navigation/indoor/projekte/messe. Apr. 2006. URL: http://www.iis.fraunhofer.de/ec/navigation/indoor/projekte/messe.

[82] Aldo Gangemi, Nicola Guarino, Claudio Masolo, Alessandro Oltramari, and Luc Schneider. "Sweetening Ontologies with DOLCE." In: *Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web*. EKAW '02. London, UK: Springer-Verlag,

2002, pp. 166–181. ISBN: 3-540-44268-5. URL: http://portal.acm.org/citation.cfm?id=645362.650863.

[83] David Garlan, Daniel P. Siewiorek, Asim Smailagic, and Peter Steenkiste. "Project Aura: Toward Distraction-Free Pervasive Computing." In: *Pervasive* 1.2 (Apr. 2002), pp. 22–31.

[84] Gershenson. *A general methodology for designing self-organizing systems. Tech. Rep. 2005-05*. Tech. rep. ECCO, 2005.

[85] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning Theory and Practice*. Morgan Kaufmann, 2004.

[86] Reginald G. Golledge. "Path Selection and Route Preference in Human Navigation: A Progress Report." In: *COSIT*. 1995, pp. 207–222.

[87] Emre Göynügür, Sara Bernardini, Geeth de Mel, Kartik Talamadupula, and Murat Şensoy. "Policy conflict resolution in iot via planning." In: *Canadian Conference on Artificial Intelligence*. Springer. 2017, pp. 169–175.

[88] Robert Grimm. "One.world: Experiences with a Pervasive Computing Architecture." In: *Pervasive* 3.3 (July 2004), pp. 22–30.

[89] Jan Grimmer. *Context Aware Indoor Navigation*. 2006.

[90] E. Guttman, C. Perkins, J. Veizades, and M. Day. *RFC 2608 - Service Location Protocol, Version 2*. https://tools.ietf.org/html/rfc2608. Accessed: 2023-02-15. June 1999.

[91] Steve Harrison and Paul Dourish. "Re-place-ing space: the roles of place and space in collaborative systems." In: *CSCW '96: Proceedings of the 1996 ACM conference on Computer supported cooperative work*. Boston, Massachusetts, United States: ACM, 1996, pp. 67–76. ISBN: 0-89791-765-0. DOI: http://doi.acm.org/10.1145/240080.240193.

[92] George Hatzivasilis, Ioannis Askoxylakis, George Alexandris, Darko Anicic, Arne Bröring, Vivek Kulkarni, Konstantinos Fysarakis, and George Spanoudakis. "The Interoperability of Things: Interoperable solutions as an enabler for IoT and Web 3.0." In: Sept. 2018. DOI: 10.1109/CAMAD.2018.8514952.

[93] George Hatzivasilis, George Floros, Ioannis Papaefstathiou, and Harry Manifavas. "Lightweight authenticated encryption for embedded on-chip systems." In: *Information Security Journal: A Global Perspective* (Aug. 2016), pp. 1–11. DOI: 10.1080/19393555.2016.1209259.

[94] George Hatzivasilis, Konstantinos Fysarakis, Othonas Soultatos, Ioannis Askoxylakis, Ioannis Papaefstathiou, and Giorgos Demetriou. "The industrial internet of things as an enabler for a circular economy Hy-LP: a Novel IIoT protocol, evaluated on a wind park's SDN/NFV-enabled 5G industrial network." In: *Computer communications* 119 (2018), pp. 127–137.

[95] George Hatzivasilis, Ioannis Papaefstathiou, Dimitris Plexousakis, Charalampos Manifavas, and Nikos Papadakis. "AmbISPDM." In: *Applied Intelligence* 48.6 (June 2018), pp. 1623–1643. ISSN: 0924-669X. DOI: 10.1007/s10489-017-1030-0. URL: https://doi.org/10.1007/s10489-017-1030-0.

[96] Davin Heckman. *A Small World: Smart Houses and the Dream of the Perfect Day*. Duke University Press, 2008.

[97] Davin Heckman. *Small World: Smart Houses and the Dream of the Perfect Day*. Duke University Press, Feb. 2008.

[98] Dominik Heckmann and Antonio Krüger. "A User Modeling Markup Language (UserML) for Ubiquitous Computing." In: *User Modeling*. Ed. by Peter Brusilovsky, Albert T. Corbett, and Fiorella de Rosis. Vol. 2702. Lecture Notes in Computer Science. Springer, 2003, pp. 393–397. ISBN: 3-540-40381-7.

[99] T. Heider. "Goal oriented assistance for extended multimedia systems and dynamic technical infrastructures." In: *Proceedings of the Seventh IASTED International Conference on Internet and Multimedia Systems and Applications*. 2003, pp. 62–67.

[100] Andreas Heinemann, Jussi Kangasharju, Fernando Lyardet, and Max Mühlhäuser. "iClouds – Peer-to-Peer Information Sharing in Mobile Environments." In: *Euro-Par 2003. Parallel Processing, 9th International Euro-Par Conference*. Ed. by Harald Kosch, László Böszörményi, and Hermann Hellwagner. Vol. 2790. Lecture Notes in Computer Science. Klagenfurt, Austria: Springer, 2003, pp. 1038–1045.

[101] Andreas Heinemann, Jussi Kangasharju, Fernando Lyardet, and Max Mühlhäuserl. "iClouds – Peer-to-Peer Information Sharing in Mobile Environments." In: *Euro-Par 2003. Parallel Processing, 9th International Euro-Par Conference*. Ed. by Harald Kosch, László Böszörményi, and Hermann Hellwagner. Vol. 2790. Lecture Notes in Computer Science. Klagenfurt, Austria: Springer, 2003, pp. 1038–1045.

[102] Michael Hellenschmidt. "Distributed implementation of a self-organizing appliance middleware." In: *sOc-EUSAI '05*. 2005.

[103] Xingjian Hou and Chuanfei Guo. "Sensing mechanisms and applications of flexible pressure sensors." In: *Acta Physica Sinica* 69 (2020), pp. 178102–1–178102–16.

[104] Manfred Husty. *Darstellende Geometrie – Technische Mathematik*. 2005.

[105] Stephen S. Intille. "Designing a Home of the Future." In: *IEEE Pervasive Computing* 1.2 (2002), pp. 76–82. ISSN: 1536-1268. DOI: http://dx.doi.org/10.1109/MPRV.2002.1012340.

[106] Manyika J, Chui M, Bisson P, Woetzel J, Dobbs R, Bughin J, and Aharon D. *The internet of things: mapping the value beyond the hype*. Tech. rep. McKinsey global institute, 2015.

[107] *JSON-LD*. 2021. URL: https://www.w3.org/TR/json-ld/.

[108] R. A. Jarvis. "On the identification of the Convex Hull of a Finite Set of Points in the Plane." In: *Information Processing Letters* 1.2 (1973), pp. 18–22.

[109] Brad Johanson, Armando Fox, and Terry Winograd. "The Interactive Workspaces Project: Experiences with Ubiquitous Computing Rooms." In: *Pervasive* 1.2 (Apr. 2002), pp. 71–78.

[110] Bradley Earl Johanson. "Application Coordination Infrastructure for Ubiquitous Computing Rooms." PhD thesis. Stanford University, 2003.

[111] Eirini Kaldeli, Alexander Lazovik, and Marco Aiello. "Domain-independent planning for services in uncertain and dynamic environments." In: *Artificial Intelligence* 236 (2016), pp. 30–64. ISSN: 0004-3702. DOI: https://doi.org/10.1016/j.artint.2016.03.002. URL: https://www.sciencedirect.com/science/article/pii/S000437021630025X.

[112] H. Kanai, T. Nakada, Y. Hanba, and S. Kunifuji. "A Support System for Context Awareness in a Group Home Using Sound Cues." In: *Methods of Information in Medicine* 47.3 (2008), pp. 198–202. ISSN: 0026-1270.

[113] Fahim Kawsar. "Lessons Learned in Smart Environments Engineering." In: *First International Workshop on Design and Integration Principles for Smart Objects (DiPSO'07). Workshop Proceedings of Ubicomp'07*. 2007.

[114] Fahim Kawsar, Fernando Lyardet, and Tatsuo Nakajima. "Three Challenges for Future Smart Object Systems." In: *AmI-Blocks'08: Second European Workshop on Smart Products: Building Blocks of Ambient Intelligence*. 2008, pp. 35–37.

[115] Cory D. Kidd, Robert Orr, Gregory D. Abowd, Christopher G. Atkeson, Irfan A. Essa, Blair MacIntyre, Elizabeth D. Mynatt, Thad Starner, and Wendy Newstetter. "The Aware Home: A Living Laboratory for Ubiquitous Computing Research." In: *CoBuild '99: Proceedings of the Second International Workshop on Cooperative Buildings, Integrating Information, Organization, and Architecture*. London, UK: Springer-Verlag, 1999, pp. 191–198. ISBN: 3-540-66596-X.

[116] M. Wieland O. Kopp, D. Nicklas, and F. Leymann. "Towards Context-aware Workflows." In: *Proc. CAiSE'07 Workshop UMICS*. 2007.

[117] István Koren and Ralf Klamma. "The Exploitation of OpenAPI Documentation for the Generation of Web Frontends." In: *Companion Proceedings of the The Web Conference 2018*. WWW '18. Lyon, France: International World Wide Web Conferences Steering Committee, 2018, pp. 781–787. ISBN: 9781450356404. DOI: 10.1145/3184558.3188740. URL: https://doi.org/10.1145/3184558.3188740.

[118] Sebastian Kotstein and Christian Decker. "Reinforcement learning for IoT interoperability." In: *2019 IEEE International Conference on Software Architecture Companion (ICSA-C)*. IEEE. 2019, pp. 11–18.

[119] Antonio Krüger, Jörg Baus, Dominik Heckmann, Michael Kruppa, and Rainer Wasinger. "Adaptive Mobile Guides." In: *The Adaptive Web*. Ed. by Peter Brusilovsky, Alfred Kobsa, and Wolfgang Nejdl. Vol. 4321. Lecture Notes in Computer Science. Springer, 2007, pp. 521–549. ISBN: 978-3-540-72078-2.

[120] T. Kugler and M. Merz. "UPnP Service Discovery Protocol." In: *Proceedings of the 10th IEEE Workshop on Local and Metropolitan Area Networks (LANMAN)*. 2001, pp. 183–188. DOI: 10.1109/LANMAN.2001.939523.

[121] Deutsches Forschungszentrum für Künstliche Intelligenz. *The German Text-to-Speech Synthesis System MARY*. URL: http://mary.dfki.de/.

[122] Mark Lampe. "Testing and Simulation Support for Smart Products with Areas of Influence." MA thesis. TU Darmstadt, 2010.

[123] Chia-Hsun Jackie Lee, Leonardo Bonanni, Jose H. Espinosa, Henry Lieberman, and Ted Selker. "Augmenting kitchen appliances with a shared context using knowledge about daily events." In: *IUI '06: Proceedings of the 11th international conference on Intelligent user interfaces*. Sydney, Australia: ACM, 2006, pp. 348–350. ISBN: 1-59593-287-9. DOI: http://doi.acm.org/10.1145/1111449.1111533.

[124] Frédéric Lemoine, Tatiana Aubonnet, and Noëmie Simoni. "IoT composition based on self-controlled services." In: *Journal of Ambient Intelligence and Humanized Computing* 11.11 (2020), pp. 5167–5186.

[125] H. Lieberman and J. Espinosa. "A goal-oriented interface to consumer electronics using planning and commonsense reasoning." In: *Proceedings of the 11th international Conference on intelligent User interfaces In (Sydney, Australia, January 29 - February 01, 2006) IUI '06*. Ed. by John Krumm, Gregory D. Abowd,

Aruna Seneviratne, and Thomas Strang. Vol. 4717. LNCS. New York: ACM Press, 2006, pp. 226–233. ISBN: 978-3-540-74852-6.

[126] Jie Lin, Wei Yu, Nan Zhang, Xinyu Yang, Hanlin Zhang, and Wei Zhao. "A Survey on Internet of Things: Architecture, Enabling Technologies, Security and Privacy, and Applications." In: *IEEE Internet of Things Journal* PP (Mar. 2017), pp. 1–1. DOI: 10.1109/JIOT.2017.2683200.

[127] B. Liu. "Intelligent Route Finding: Combining Knowledge, Cases and an Efficient Search Algorithm." In: *Proceedings of ECAI96, 12th European Conference on Artificial Intelligence*. John Wiley & Sons, Ltd, 1996, pp. 149–155.

[128] P. Lukowicz, J.A. Ward, H. Junker, M. Stäger, G. Tröster, A. Atrash, and T. Starner. "Recognizing workshop activity using body worn microphones and accelerometers." In: *Pervasive Computing* (2004), pp. 18–32.

[129] Fernando Lyardet. "Ambient Learning." In: *Ubiquitous Computing Technology for Real Time Enterprises*. Information Science Reference, 2008. Chap. 12, pp. 530–549.

[130] Fernando Lyardet. "Ambient Learning." In: *Handbook of Research on Ubiquitous Computing Technology for Real Time Enterprises*. 2008.

[131] Fernando Lyardet and Erwin Aitenbichler. "Smart Products: Building Blocks of Ambient Intelligence." In: *Constructing Ambient Intelligence: AmI 2007 Workshops; Darmstadt, Germany, November 2007; Revised Papers*. Vol. 11. CCIS. Springer Verlag, Heidelberg, 2007, pp. 156–157. ISBN: 978-3-540-85378-7.

[132] Fernando Lyardet and Erwin Aitenbichler. "AmI-Blocks'08: Introduction and Summary of Contents." In: *AmI-Blocks'08: Second European Workshop on Smart Products: Building Blocks of Ambient Intelligence*. 2008, pp. 1–2.

[133] Fernando Lyardet and Erwin Aitenbichler. *AmI-Blocks'09: Third European Workshop on Smart Products: Building Blocks of Ambient Intelligence*. Vol. AmI'09 Adjunt Proceedings (electronic format). 2009.

[134] Fernando Lyardet, Erwin Aitenbichler, and Max Mühlhäuser. *AmI-Blocks'08: Second European Workshop on Smart Products: Building Blocks of Ambient Intelligence*. BoD, 2008. ISBN: 978-3-8391-0868-0.

[135] Fernando Lyardet, Jan Grimmer, and Max Mühlhäuser. "CoINS: Context Sensitive Indoor Navigation System." In: *Proceedings of IEEE International Symposium on Multimedia ISM 2006*. San Diego, CA: IEEE CS press, Dec. 2006, pp. 30–37.

[136] Fernando Lyardet, Jan Grimmer, and Max Mühlhäuser. "CoINS: Context sensitive Indoor Navigation System." In: *Proceedings of IEEE International Symposium on Multimedia (ISM)*. Los Alamitos, CA, USA: IEEE Press, 2006, pp. 30–37.

[137] Fernando Lyardet, Aristotelis Hadjakos, and Diego Wong Szeto. "InSitu: An Approach for Dynamic Context Labeling Based on Product Usage and Sound Analysis." In: *Workshop Proceedings of the AAAI-11 Conference*. 2011.

[138] Fernando Lyardet and Vicente Pelechano. "First International Workshop on Web-Enableb Objects (TouchTheWeb 2010)." In: *Current Trends in Web Engineering. 10th International Conference on Web Engineering ICWE 2010 Workshops*. Ed. by Florian Daniel and Federico Michele Facca. Vol. 6385. Lecture Notes in Computer Science (LNCS). Springer Verlag, July 2010, p. 595.

[139] Fernando Lyardet and Dirk Schnelle-Walka. "Ubiquitous Explanations: Anytime, Anywhere, End-User Support." de. In: *The European Jounal for the Informatics Professional, Cepis Upgrade: Special Issue Internet of Things, vol. XII, no. 1* 12.1 (Feb. 2011), pp. 52–58. URL: http://tubiblio.ulb.tu-darmstadt.de/98459/.

[140] Fernando Lyardet and Dirk Schnelle. "Consumer Voice Interface Patterns." In: *EuroPLoP European Conference on Patterns and Pattern Languages*. 2007.

[141] Fernando Lyardet, Diego Wong Szeto, and Erwin Aitenbichler. "Context Aware Indoor Navigation." In: *Proceedings of the European Conference on Ambient Intelligence*. LNCS. Springer Verlag, 2008.

[142] Fernando Lyardet, Diego Wong Szeto, and Erwin Aitenbichler. "Context-aware indoor navigation." In: *AmI '08: Proceedings of the European Conference on Ambient Intelligence*. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 290–307.

[143] *MAP*.

[144] MIT. *The Oxygen Project*. http://oxygen.lcs.mit.edu/. http://oxygen.lcs.mit.edu/.

[145] Dragos Manolescu. "Workflow Enactment with Continuation and Future Objects." In: *Proceedings of the Conference on Object Oriented Programming Systems and Laguages*. OOPSLA'02. ACM, Nov. 2002.

[146] Markos Markou and Sameer Singh. "Novelty detection: a review–part 1: statistical approaches." In: *Signal Processing* 83.12 (2003), pp. 2481 –2497.

[147] Simon Mayer, Dominic Plangger, Florian Michahelles, and Simon Rothfuss. "UberManufacturing: A goal-driven collaborative industrial manufacturing marketplace." In: *Proceedings of the 6th International Conference on the Internet of Things*. 2016, pp. 111–119.

[148] Simon Mayer, Ruben Verborgh, Matthias Kovatsch, and Friedemann Mattern. "Smart configuration of smart environments." In: *IEEE Transactions on Automation Science and Engineering* 13.3 (2016), pp. 1247–1255.

[149] C. Metzger. "High Fidelity Shelf Stock Monitoring: A Framework for Retail Replenishment Optimization." In: 2009.

[150] Christian Metzger, Jan Meyer, Elgar Fleisch, and Gerhard Tröster. "Weight-Sensitive Foam to Monitor Product Availability on Retail Shelves." In: *Pervasive*. 2007.

[151] G.A. Miller. "WordNet: a lexical database for English." In: *Communications of the ACM* 38.11 (1995), pp. 39–41. ISSN: 0001-0782.

[152] Dalibor Mitrovic and Horst Eidenberger. *Towards an Optimal Feature Set for Environmental Sound Recognition*. Tech. rep. Vienna University of Technology, 2006.

[153] Douglas C Montgomery, Elizabeth A Peck, and G Geoffrey Vining. *Introduction to linear regression analysis*. John Wiley & Sons, 2021.

[154] Michael C. Mozer. "Lessons from an adaptive house." In: *Smart environments: Technologies, protocols, and applications*. Wiley & Sons, 2005. Chap. 12, pp. 273–294.

[155] Hideyuki Nakashima, Hamid Aghajan, and Juan Carlos Augusto. *Handbook of Ambient Intelligence and Smart Environments*. 1st. Springer Publishing Company, Incorporated, 2009. ISBN: 0387938079.

[156] Network Working Group. *IETF RFC 3261. SIP: Session Initiation Protocol.* URL: http://www.ietf.org/rfc/rfc3261.txt.

[157] Mahda Noura, Mohammed Atiquzzaman, and Martin Gaedke. "Interoperability in Internet of Things: Taxonomies and Open Challenges." In: *Mobile Networks and Applications* (June 2019). DOI: 10.1007/s11036-018-1089-9.

[158] Mahda Noura and Martin Gaedke. "An Automated Cyclic Planning Framework Based on Plan-Do-Check-Act for Web of Things Composition." In: *Proceedings of the 10th ACM Conference on Web Science*. WebSci '19. Boston, Massachusetts, USA: Association for Computing Machinery, 2019, 205–214. ISBN: 9781450362023. DOI: 10.1145/3292522.3326044. URL: https://doi.org/10.1145/3292522.3326044.

[159] OMG. "M.: A Comparison of BPML and BPEL4Ws." In: *Proceedings of the 1st Conference "Berliner XML-Tage". Berlin 2003*. 2003, pp. 305–316.

[160] OpenEmcee. *OpenEmcee Microflow Engine*. Accessed: March 14, 2023. 2023. URL: https://github.com/OpenEmcee/openemcee.

[161] World Health Organization. *International Classification of Functioning, Disability and Health (ICF)*. URL: http://www.who.int/classifications/icf/en/.

[162] Javier Palanca, Elena Del Val, Ana Garcia-Fornes, Holger Billhardt, Juan Manuel Corchado, and Vicente Julián. "Designing a goal-oriented smart-home environment." In: *Information Systems Frontiers* 20.1 (2018), pp. 125–142.

[163] F. Paternò, C. Mancini, and S. Meniconi. "Concurtasktrees: A diagrammatic notation for specifying task models." In: *NTERACT '97: Proceedings of the IFIP TC13 International Conference on Human-Computer Interaction*. London, UK: Chapman & Hall, Ltd., 1997, pp. 362–369.

[164] Charith Perera, Arkady Zaslavsky, Peter Christen, and Dimitrios Georgakopoulos. "Context aware computing for the internet of things: A survey." In: *IEEE communications surveys & tutorials* 16.1 (2013), pp. 414–454.

[165] Cuong Pham, Thomas Plötz, and Patrick Olivier. "A Dynamic Time Warping Approach to Real-Time Activity Recognition for Food Preparation." In: *Ambient Intelligence*. Ed. by Boris de Ruyter, Reiner Wichert, David Keyson, Panos Markopoulos, Norbert Streitz, Monica Divitini, Nikolaos Georgantas, and Antonio Mana Gomez. Vol. 6439. Lecture Notes in Computer Science. Springer, 2010, pp. 21–30.

[166] Matthai Philipose, Kenneth P, Mike Perkowitz, Donald J. Patterson, Dieter Fox, Henry Kautz, and Dirk Hähnel. "Inferring activities from interactions with objects." In: *IEEE Pervasive Computing* 3 (2004), pp. 50–57.

[167] Philips. *Philips Research*. url http://www.research.philips.com/. 2003. URL: http://www.research.philips.com/.

[168] BMBF Project. *EMBASSI (Electronic Multimedia Operating and Service Assistance)*. http://www.igd.fhg.de/igd-a1/projects/ambientintelligence/index.html. URL: http://www.igd.fhg.de/igd-a1/projects/ambientintelligence/index.html.

[169] Kekoa Proudfoot. *Unofficial Quake 3 Map Specs*. <http://graphics.stanford.edu/~kekoa/q3/>. Mar. 2000. URL: http://graphics.stanford.edu/~kekoa/q3/.

[170] *RDF*. 2021. URL: https://www.w3.org/TR/rdf11-concepts/.

[171] *Radiant*. <http://www.qeradiant.com>. Mar. 2006. URL: http://www.qeradiant.com.

[172] Vithala R Rao. *Applied conjoint analysis*. Springer Science & Business Media, 2014.

[173] Microsoft Research. *The Easy Living Project*. http://research.microsoft.com/easyliving/. Dec. 2008.

[174] K-F. Richter and M. Duckham. "Simplest Instructions: Finding Easy-to-Describe Routes for Navigation." In: *Lecture Notes in Computer Science (GIScience 2008)*. 2008.

[175] S.A. Rijsdijk, Hultink, E.J. Jan, and A. Diamantopoulos. *Product Intelligence: Its Conceptualization, Measurement and Impact on Consumer Satisfaction*. Tech. rep. ERIM Report Series Reference No. ERS-2007-006-ORG, 2007.

[176] B Roberts. "Integration vs interoperability: what's the difference?" In: *Alpharetta, GA: Surgical Information Systems* (2017).

[177] Manuel Roman, Christopher Hess, Renato Cerqueira, Anand Ranganathan, Roy H. Campbell, and Klara Nahrstedt. "Gaia: A Middleware Infrastructure for Active Spaces." In: *Pervasive* 1.4 (Oct. 2002), pp. 74–83.

[178] J. Rosenberg, H. Schulzrinne, G. Camarillo, A. Johnston, J. Peterson, R. Sparks, M. Handley, and E. Schooler. *Session Initiation Protocol (SIP): Invite and Associated Requests for Initiating Sessions*. RFC 3261. June 2002. URL: https://tools.ietf.org/html/rfc3261.

[179] Gustavo Rossi, Silvia Gordillo, and Fernando Lyardet. "Design Patterns for Context Aware Adaptation." In: *Workshop on Context Aware Adaptation and Personalization for the Mobile Internetl*. IEEE. New York, NY: IEEE Press, 2005, pp. 170–173.

[180] Stuart J. Russell and Peter Norvig. *Artificial Intelligence: A Modern Approach*. Vol. 82. Prentice Hall, 1996, pp. 369–380.

[181] Irina Rychkova and Linh Thao Ly. "Automatic generation of business process models using JPEd graphical editor." In: *IEEE International Conference on e-Technology, e-Commerce and e-Service*. 2005, pp. 221–228. DOI: 10.1109/EEE.2005.56.

[182] Thomas L. Saaty. "How to Make a Decision: The Analytic Hierarchy Process." In: *Interfaces* 24 (1990), pp. 19–43.

[183] Davood Sabaei, John Erkoyuncu, and Rajkumar Roy. "A Review of Multi-criteria Decision Making Methods for Enhanced Maintenance Delivery." In: vol. 37. Oct. 2015. DOI: 10.1016/j.procir.2015.08.086.

[184] Hanan Samet. "The Quadtree and Related Hierarchical Data Structures." In: *ACM Comput. Surv.* 16.2 (1984), pp. 187–260. ISSN: 0360-0300. DOI: http://doi.acm.org/10.1145/356924.356930.

[185] R. Schäfer. *Rules for using multi-attribute utility theory for estimating a user's interest*. Tech. rep. DFKI, 2001.

[186] Ralph Schäfer. "Rules for Using Multi-Attribute Utility Theory for Estimating a User's Interests." In: 2001.

[187] Bernt Schiele. "Sensory Augmented Computing and its Potential for Human-Computer Interaction (Sensorunterstützte Computersysteme und deren Potenzial für die Mensch-Maschine-Interaktion)." In: *it - Information Technology* 50.1 (2008), pp. 40–44.

[188] Albrecht Schmidt, Martin Strohbach, Kristof Van Laerhoven, Adrian Friday, and Hans-Werner Gellersen. "Context Acquisition Based on Load Sensing." In: *UbiComp*. 2002.

[189] U. Schmidt. "Gebrauchsanweisungen - Form und Struktur." PhD thesis. Rheinische Friedrich-Wilhelms-Universität zu Bonn, 1996.

[190] Dirk Schnelle and Fernando Lyardet. "Voice User Interface Design Patterns." In: *Proceedings of 11th European Conference on Pattern Languages of Programs (EuroPlop 2006)*. to appear. 2006.

[191] Dirk Schnelle and Fernando Lyardet. "Voice User Interface Design Patterns." In: *EuroPLoP European Conference on Patterns and Pattern Languages.* 2006.

[192] Dirk Schnelle and Fernando Lyardet. "Consumer Voice User Interface Design Patterns." In: *Proceedings of 12th European Conference on Pattern Languages of Programs (EuroPlop 2007)*. 2007.

[193] Dirk Schnelle, Fernando Lyardet, and Tao Wei. "Audio Navigation Patterns." In: *Proceedings of EuroPLoP 2005*. July 2005, pp. 237–260.

[194] Dirk Schnelle, Fernando Lyardet, and Tao Wei. "Audio Navigation Patterns." In: *EuroPLoP European Conference on Patterns and Pattern Languages.* 2005.

[195] Dominik Schultz. "Smart Device Compositions." MA thesis. TU Darmstadt, 2007.

[196] Nigel Shadbolt. "Ambient Intelligence." In: *IEEE Intelligent Systems* 18.4 (2003), pp. 2–3.

[197] CC Sobin. "A survey on architecture, protocols and challenges in iot." In: *Wireless Personal Communications* 112.3 (2020), pp. 1383–1429.

[198] M.E. Sorrows and S.C. Hirtle. "The Nature of Landmarks for Real and Electronic Spaces." In: *COSIT '99*. Ed. by Christian Freksa and David M. Mark. Vol. 1661. Stade, Germany: Springer, 1999, pp. 37–50. ISBN: 3-540-66365-7.

[199] Dag Spicer. *A Computer in the Basement?* Apr. 1968.

[200] Steffen Staab, Wil van der Aalst, V. Richard Benjamins, Amit Sheth, John A. Miller, Christoph Bussler, Alexander Maedche, Dieter Fensel, and Dennis Gannon. "Web Services: Been There, Done That?" In: *IEEE Intelligent Systems* 18.1 (2003), pp. 72–85. ISSN: 1541-1672. DOI: http://doi.ieeecomputersociety.org/10.1109/MIS.2003.1179197.

[201] Robert J. Sternberg. "The concept of intelligence and its role in lifelong learning and success." In: *American Psychologist* 52 (1997), pp. 1030–1037.

[202] Lucy A. Suchman. "Affiliative Objects." In: *Organization* 12 (3 2005), pp. 379–399.

[203] Lucy A. Suchman. *Human-Machine Reconfigurations: Plans and Situated Actions*. New York, NY, USA: Cambridge University Press, 2006. ISBN: 052167588X.

[204] Diego Wong Szeto. "Adaptive Indoor Navigation." MA thesis. TU Darmstadt, 2008.

[205] Diego Wong Szeto. "InSitu: An Architecture for Situation Recognition and Assessment." MA thesis. TU Darmstadt, 2011.

[206] Feilong Tang, Minyi Guo, Mianxiong Dong, Minglu Li, and Hu Guan. "Towards Context-Aware Workflow Management for Ubiquitous Computing." In: *Embedded Software and Systems, 2008. ICESS '08. International Conference on* (July 2008), pp. 221–228.

[207] E. M. Tapia, S. Intille, and K. Larson. "Activity Recognition in the Home using Simple and Ubiquitous Sensors." In: *Proc. of Pervasive 2004*. Springer, Apr. 2004, pp. 158 –175. ISBN: 3-540-21835-1.

[208] Noel Nuo Wi Tay, Janos Botzheim, and Naoyuki Kubota. "Human-centric automation and optimization for smart homes." In: *IEEE Transactions on Automation Science and Engineering* 15.4 (2018), pp. 1759–1771.

[209] The Apache Software Foundation. *Apache Kafka*. https://kafka.apache.org/. [Online; accessed 13 March 2023]. 2022.

[210] *The W3C Web of Things*. 2021. URL: https://www.w3.org/WoT/.

[211] S. Thomson, T. Narten, and T. Jinmei. *RFC 4862 - IPv6 Stateless Address Autoconfiguration*. https://tools.ietf.org/html/rfc4862. Accessed: 2023-02-15. Sept. 1998.

[212] Unify-IoT. *Deliverable D03.01 Report on IoT platform activities*. Tech. rep. 2016.

[213] Carnegie Mellon University. *CMU Sphinx: The Open Source Toolkit for Speech Recognition*. URL: http://cmusphinx.sourceforge.net/.

[214] *W3C Linked Data*. 2021. URL: https://www.w3.org/standards/semanticweb/data.

[215] *W3C WoT Thing Description*. 2021. URL: https://w3c.github.io/wot-thing-description/.

[216] W3C. *W3C Semantic Integration and Interoperability Using RDF and OWL*. 2021. URL: https://www.w3.org/2001/sw/BestPractices/OEP/SemInt/.

[217] Wolfgang Wahlster. "Smartkom: Fusion and fission of speech, gestures, and facial expressions." In: *Proceedings of the 1st International Workshop on Man-Machine Symbiotic Systems*. 2002, pp. 213–225.

[218] A. Ward, A. Jones, and A. Hopper. "A new location technique for the active office." In: *Personal Communications, IEEE* 4 (5 1997), pp. 42–47.

[219] Torben Weis, Marcus Handte, Mirko Knoll, and Christian Becker. "Customizable Pervasive Applications." In: *PerCom 2006*. 2006.

[220] M. Weiser. "The Computer for the 21st Century." In: *Scientific American* 265 (1991), pp. 66–75.

[221] Mark Weiser. "The Computer for the Twenty-First Century." In: *Scientific American* 24.7 (Sept. 1991), pp. 94–100.

[222] Matthias Wieland, Peter Kaczmarczyk, and Daniela Nicklas. "Context Integration for Smart Workflows." In: *Proceedings of the Sixth Annual IEEE International Conference on Pervasive Computing and Communications PerCom*. IEEE Computer Society, 2008, pp. 239–242.

[223] Detlof von Winterfeldt and Ward Edwards. "Decision Analysis and Behavioral Research." In: 1986.

[224] Zhibiao Wu and Martha Palmer. "Verbs semantics and lexical selection." In: *Proceedings of the 32nd annual meeting on Association for Computational Linguistics*. ACL '94. Las Cruces, New Mexico: Association for Computational Linguistics, 1994, pp. 133–138.

[225] G. Z. Yang and D. F. Gillies. *Computer Vision*. Imperial College, 1996. Chap. 4.

[226] Stephen S Yau and Arun Balaji Buduru. "Intelligent planning for developing mobile IoT applications using cloud systems." In: *2014 IEEE International Conference on Mobile Services*. IEEE. 2014, pp. 55–62.

[227] G. Michael Youngblood, Lawrence B. Holder, and Diane J. Cook. "Managing Adaptive Versatile Environments." In: *PerCom*. IEEE Computer Society, Mar. 2005, pp. 351–360. ISBN: 0-7695-2299-8.

[228] Y. Zhan, S. Miura, J. Nishimura, and T. Kuroda. "Human activity recognition from environmental background sounds for wireless sensor networks." In: *Networking, Sensing and Control, 2007 IEEE International Conference on*. IEEE. 2007, pp. 307–312. ISBN: 1424410762.

[229] Bo Zhou, Monit Shah Singh, Sugandha Doda, Muhammet Yildirim, Jingyuan Cheng, and Paul Lukowicz. "The carpet knows: Identifying people in a smart environment from a single step." In: *2017 IEEE International Conference on Pervasive Computing and Communications Workshops (PerCom Workshops)* (2017), pp. 527–532.

[230] A. Zinnen and B. Schiele. "A new approach to enable gesture recognition in continuous data streams." In: *ISWC 2008*. 2008.

[231] J. Rocha Jr. et al. "X-Peer: A Middleware for Peer-to-Peer Applications." In: *Proceedings of 1st Brazilian Wksp. Peer-to-Peer*. 2005.