ICT-2009-248730

Florence

Multi Purpose Mobile Robot for Ambient Assisted Living

STREP
Contract Nr: 248730

Deliverable: D3.1 State of the Art in Monitoring and Decision Making

Due date of deliverable: (31-07-2010)
Actual submission date: (30-07-2010)

Start date of Project: 01 February 2010 Duration: 36 months

Responsible WP: Novay

Revision: final

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)		
Dissemination level		
PU	Public	X
PP	Restricted to other programme participants (including the Commission Service	
RE	Restricted to a group specified by the consortium (including the Commission Services)	
CO	Confidential, only for members of the consortium (excluding the Commission Services)	

0 DOCUMENT INFO

0.1 Author

Author	Company	E-mail
Elena Cruz	TID	mecm@tid.es
Dirk-Jan van Dijk	Novay	Dirk-Jan.vanDijk@novay.nl
Stefan Gessler	NEC	Stefan.Gessler@neclab.eu
Leszek Holenderski	Philips	leszek.holenderski@philips.com
Benjamin Hebgen	NEC	Benjamin.Hebgen@neclab.eu
Melvin Isken	OFFIS	Melvin.Isken@offis.de
Mortaza S. Bargh	Novay	Mortaza.Bargh@novay.nl
Niels Snoeck	Novay	Niels.Snoeck@novay.nl

0.2 Documents history

Document version #	Date	Change
V0.1	04/03/2010	Starting version, template
V0.2	06/04/2010	Definition of ToC
V0.3	17/05/2010	First almost complete draft
V0.4		Integrated version (send to WP members)
V0.5		Updated version (send PCP)
V0.6	29/06/2010	Updated version (send to project internal reviewers)
Sign off	09/07/2010	Signed off version (for approval to PMT members)
V1.0	28/07/2010	Approved Version to be submitted to EU

0.3 Document data

Keywords	actuation triggering, actuators, context enhancement, context management platforms, decision making, sensor fusion, state of the art
Editor Address data	Name: Mortaza S. Bargh Partner: Novay Address: Brouwerijstraat 1 7523XC Enschede The Netherlands Phone: +31 (0)53 48 50 443 Fax: +31 (0)53 45 50 400 E-mail: Mortaza.Bargh@novay.nl
Delivery date	31/07/2010

0.4 Distribution list

Date	Issue	E-mailer
30/07/2010	Consortium members	al_florence_all@natlab.research.philips.com
	Project Officer	Luiz.Santos@ec.europa.eu
	EC Archive	INFSO-ICT-248730@ec.europa.eu

Table of Contents

1 Introduction

Work Package 3 (WP3) of the Florence project, entitled “Monitoring and Decision Making”, addresses topics such as collecting sensory information, enhancing sensory information, and making intelligent decisions for the Florence services and system. This deliverable marks the start of WP3 work and its objectives, scope and outline are described in the following subsections.

1.1 Objective

This deliverable provides an overview of the state of the art in the topics relevant to WP3. The objectives of WP3 are to devise an infrastructure for collecting sensory information, for deriving high level context information, and for context based reasoning and decision making. This requires integration of robots into a (intelligent) home environment so that the Florence services offered to users become adaptive to contextual circumstances and user preferences. The work package is going to adopt a pragmatic approach by being application driven to meet the needs of the Florence services and system.

To achieve its objectives the work package will use the existing state of the art solutions and extend/enhance them with the features specific to Florence requirements and needs. To this end, we started our work with studying and analyzing those existing solutions, methods and tools that can potentially be relevant to WP3 objectives. This deliverable aims at providing an overview of these approaches and reporting on our preliminary analysis results. Hereby our ultimate goal is to lay down a solid ground for our upcoming activities in WP3, namely design and implementation of context sensing, context based reasoning and decision making, and actuation.

1.2 Scope

The scope of the document matches that of WP3. There are four focus areas in WP3 corresponding to tasks T3.2-T3.5, namely: context management platforms, context enhancement and monitoring, decision making and coordination, and robot action and actuation triggering.

The focus area of context management platforms will particularly look at existing platforms that can be extended or integrated with robotic platforms for context sensing. The focus area of context enhancement and monitoring will look at sensor fusion and reasoning techniques suitable for deriving (semantically and qualitatively) high level context information. Further, the review will identify some challenges and (open) issues that the project will face. As an example, detecting user activities is considered to be a relevant topic within the Florence setting. The focus area of decision making and coordination will review the decision process in charge of mapping system inputs to desired system outputs based on the derived context information. Further, the review will identify some challenges and (open) issues that the project will face. As an example, learning user behaviour and models is considered to be a relevant topic within the Florence setting. The focus area of actuation triggering will look at software and hardware platforms that can be used for robot action triggering and for triggering of other actuators in home environments. Further, the study will identify the expected integration and interoperability challenges. Figure 1-1 illustrates the relations and dependencies between the aforementioned focus areas.

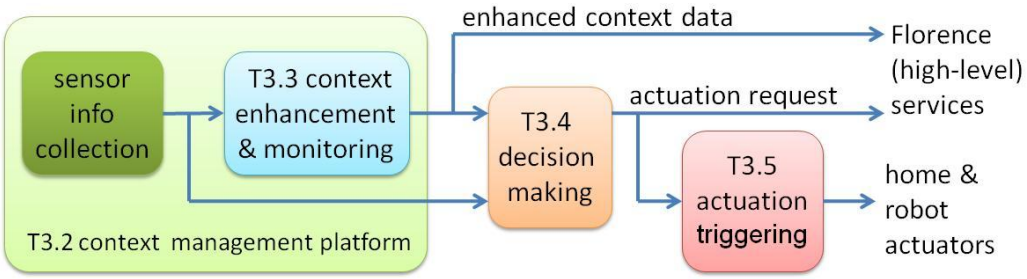

Figure 1-1: relations among WP3 tasks (and sections of this deliverable).

1.3 Outline

The deliverable starts with addressing conceptual topics and increasingly encompasses reviewing of practical and commercial products. The deliverable is structured as follows:

- Section 2 provides an overview of intelligent decision making approaches and the main (research) issues and challenges.
- Section 3 gives a summary of methods, approaches and challenges of sensor fusion and context enhancement.
- Section 4 provides an overview of context management platforms in non-robotic and robotic settings.
- Section 5 provides an overview of hardware and software platforms used for robot action triggering and (home) actuator triggering.
- Finally, Section 6 summarises our conclusions and the envisioned future work.

2 Decision making

In this section we give a high level overview of various aspects of automated decision making in complex hardware/software systems. Our scope is discrete systems, so continuous and hybrid systems are not discussed, and we only consider decision making implemented in software.

As stated in the Florence Description of Work document [DoW2009, Task 4.1 description], the control software of the Florence system should take into consideration several important aspects:

1. Distribution of intelligence in decision making,
2. Decomposition of high level goals,
3. Prioritization and coordination of executable tasks,
4. Integration of human intelligence in decision making (humans in the loop),
5. Adaptive decision making (learning of user models).

The first three aspects are discussed in Subsections 2.2, 2.3 and 2.4. The remaining ones are discussed in Subsection 2.5. We precede the detailed discussion by a general introduction to intelligent decision making, in Subsection 2.1.

2.1 Reactive intelligence

Automated decision making, implemented in software, is a vast and complex field, extensively studied in computer science for at least 40 to 50 years. As a result, many approaches were developed over time, and diverse terminology proliferated through the research community. To put some order into this chaos, we first set an abstract conceptual framework that allows discussing various aspects of decision making in some uniform language.

The framework is based on the simple concept of reactive intelligence. Its essence is to perceive all services provided by a complex hardware/software system, like the Florence robot, as just one, albeit very complex, reactive system (i.e., a set of sensors and actuators, and control that decides what actuators to trigger in response to particular stimuli from sensors). Here sensors and actuators are treated quite abstractly, as just inputs and outputs of the control. This abstraction allows us to uniformly treat both the physical sensors that measure environmental conditions (like temperature, or light intensity) and the interaction sensors that allow a user to interact with the system (say, a button on a remote controller). Similarly, we do not distinguish between physical actuators (to dim light, for example) and interaction actuators (to display some dialog on a screen, for example).

The main concept in reactive systems is the concept of the reaction itself. Formally, a reaction is a pair stimulus → response, where stimulus is a tuple of sensors (with their values), and response is a tuple of actuators (with their values). Intuitively, reaction s → r describes a single behaviour of a reactive system: if the system is stimulated by s, it should respond with r. "Stimulated by s" means that all sensors mentioned in s have appropriate values, i.e. the ones mentioned in s. "Should respond with r" means that all actuators mentioned in r should be triggered with appropriate values, i.e. the ones mentioned in r.

The externally observable behaviour of a reactive system is just a set of reactions it can perform. This set can be further structured. For example, the behaviour can be

modelled as a set of sequences of reactions (to emphasize temporal dependencies between reactions), or a tree of reactions (to emphasize temporal variants) or a graph of reactions (to express tricky causal-effect dependencies between reactions).

In this simple framework of reactive intelligence, the terms “decision making”, "intelligence" and "control" are just equated to "behaviour", and thus can be given quite precise meaning.

2.2 Distribution of intelligence

Control can be implemented as one monolithic component/service, or distributed over several components/services. The distributed variant is more flexible. For example, it allows splitting control into local control, for each service provided by Florence, and global control that controls a set of cooperating Florence services. Typically, distributed control is implemented as "distributed objects" (based on remote method invocation) or "process networks" (based on message passing).

To large extent, the two approaches are equivalent, in the sense that each of them can simulate the other one relatively easy (message passing can be implemented with two remote methods read and write, and remote method invocation is typically implemented with message passing anyway). Due to proliferation of OO (Object Oriented) programming paradigm, the distributed objects approach is more common nowadays than the historically earlier process networks approach. However, in what follows, we concentrate on process networks because this approach directly fits the reactive systems model on which the reactive intelligence is based.

2.2.1 Process networks

A process network is formed from processes, by connecting output ports of some processes to input ports of other processes. The connections are called communicating channels. To desynchronize a sender of a message from its receiver, a channel stores sent messages in a buffer, until a receiver is ready to read it. In the most popular variant of process networks, channels are organized as bounded FIFO (First-In-First-Out) buffer, although the theoretical model of such networks, called Kahn networks, assumes unbounded, i.e., potentially infinite, FIFO buffers.

Other variants of channels are possible. For example, in synchronous networks (see the next subsection), channels are just one-slot buffers. In synchronizing networks (as in Communicating Sequential Processes [Hoare1985], with its rendezvous communication model), channels are zero-slot buffers. Channels can support peer-to-peer or broadcast communication.

Process networks can be hierarchic in that a process can be itself a network.

Distributed control can be implemented as purely asynchronous, purely synchronous, or a hybrid of the two. Such hybrids can be of two kinds: GALS (Globally Asynchronous, Locally Synchronous) or LAGS (Locally Asynchronous, Globally Synchronous). In the GALS architecture, the nodes of the top level network communicate asynchronously, while the process networks in the nodes of the asynchronous network internally use synchronous communication. The LAGS architecture, which is just a dual to the GALS architecture, is rather exotic. If used at

all, it is typically used for complex streaming applications, like video/audio processing where video frames have to be synchronized with audio frames.

2.2.2 Synchronous versus asynchronous reactions

The intuitive interpretation of reaction s → r is not precise enough. It misses the time aspect: when to gather the stimulus, and when to produce the response. Essentially, there are two possible interpretations of a reaction: an asynchronous and synchronous one.

In asynchronous interpretation, the only constraint on time is that the response always happens after the stimulus, with no restriction on how long it takes to gather the stimulus and respond to it. The order of gathering the sensor values in s and triggering the actuators in r is also not restricted. Such a weak semantics causes a lot of problems with implementing a correct/reliable control, due to potential race conditions, inconsistencies in a knowledge base (if a reasoning engine is employed), deadlocks (if control contains feedback loops), and many other subtle bugs that usually manifest non-deterministically.

Most of the issues caused by the imprecise meaning of time in the asynchronous semantics are resolved if one adopts a synchronous semantics. The synchronous semantics is similar to the computational model of silicon circuits, as explained below.

In the synchronous approach to reactive systems, time is assumed to be discrete, and consisting of a sequence of so-called instants. In every instant of time, a reactive component first gathers its inputs, then computes the output values and finally emits the outputs (this is summarized by the slogan sense → think → actuate). This whole chain of events is considered to take just one unit of time (the instant itself). Moreover, communication between reactive components is assumed to be instantaneous. To explain this, assume two connected components A and B (say, A has output "a", B has input "b", and "a" is connected to "b"). When A emits on "a", in some instant, B senses "b" in the same instant. This allows for chains of reactions to perform simultaneously, in the same, and thus consistent, global state of the whole system.

As a consequence, it is much easier to program complex control. For example, race conditions and deadlocks are automatically guaranteed to be absent (provided no instantaneous feedbacks are present, which can easily be fulfilled, for example, by delaying feedbacks by one time unit). Also, high level notations (based on logic, and thus declarative instead of imperative) can be used, because Boolean operators on events have precise meaning.

2.3 Decomposition of high level goals

A system's behaviour is just a set of concrete reactions s → r, so in theory, the system's control could be simply specified by manually listing all the reactions. However, for complex systems, this approach is infeasible, due to the size of the set. In practice, a desired reaction is generated during execution time, by some computational process that derives the desired response to an actual stimulus. This derivation process is usually specified by a variant of the divide-and-conquer method where the high level goal of deriving reaction s → r is reduced to a number of simpler sub-goals.

The decomposition of high level goals is usually done on two levels. First, one can decompose a high level goal into a process network where particular reactive components are responsible for handling sub-goals, and some other components are responsible for aggregating the sub-goals into the high level goal.

Second, a reactive component itself can be programmed declaratively (i.e., its control can employ a reasoning engine that is driven by if-then rules), and then each rule is in fact such a decomposition, as explained in Subsection 2.3.1. In Subsection 2.3.2 we describe a popular variant of if-then rules called decision trees.

2.3.1 If-then rules

A reaction s → r can be seen as an if-then rule, in some logic where sensors and actuators are modelled by predicates (here we use the word "predicate" as understood in First Order Predicate Logic). More precisely, one can model sensor S by unary relation *S* such that predicate $S(p)$, for some term *p*, is true when sensor S has a value that matches *p*. Similarly, for actuator A, $A(q)$ is true when actuator A is triggered with a value that matches *q*. Under such interpretation, the rule

if $S_1(p_1)$ and $S_2(p_2)$ and … $S_m(p_m)$ **then** $A_1(q_1)$ and $A_2(q_2)$ and … $A_n(q_n)$

encodes a set of reactions that satisfy the rule. Note that this is equivalent to satisfying the conjunction of clauses:

$A_1(q_1)$ **if** $S_1(p_1)$ and $S_2(p_2)$ and … $S_m(p_m)$
$A_2(q_2)$ **if** $S_1(p_1)$ and $S_2(p_2)$ and … $S_m(p_m)$
…
$A_n(q_n)$ **if** $S_1(p_1)$ and $S_2(p_2)$ and … $S_m(p_m)$

(for convenience, we use Prolog-like notation where "F **if** G" means "**if** G **then** F").

Instead of defining predicate $A_1(q_1)$ directly in terms of sensors $S_1(p_1)$, $S_2(p_2)$, …, $S_m(p_m)$, we can decompose the clause that defines $A_1(q_1)$ into several clauses, say

$A_1(q_1)$ **if** $P(\ldots)$ and $Q(\ldots)$
$P(\ldots)$ **if** …
$Q(\ldots)$ **if** …
…

where we use intermediary predicates $P(\ldots)$ and $Q(\ldots)$ as sub-goals. This process of decomposition can be applied as deeply as desired.

The if-then rules can then be interpreted by some reasoning engine, to derive actual response *r* to actual stimulus *s*. This is done by assuming *s* as facts and then inferring *r* as new facts, in accordance with the rules. Typical logics that can be used for this purpose are mentioned in Subsection 2.4.3.

2.3.2 Decision trees

A decision tree [DTree2010] is a decision support tool that uses a tree-like graph to model decisions and their possible consequences, including chance event outcomes, resource costs, and utility. Decision trees are commonly used to help identify a

strategy that is most likely to reach a goal. Another use of decision trees is as a descriptive means for calculating conditional probabilities. A decision tree (an example is shown in Figure 2-1 below) consists of three types of nodes:

1. Decision nodes. A decision should be taken at that point in the process (Represented by a square)
2. Chance nodes. At that point in the process happens a random event (Represented by an eclipse)
3. End nodes. Final decision.

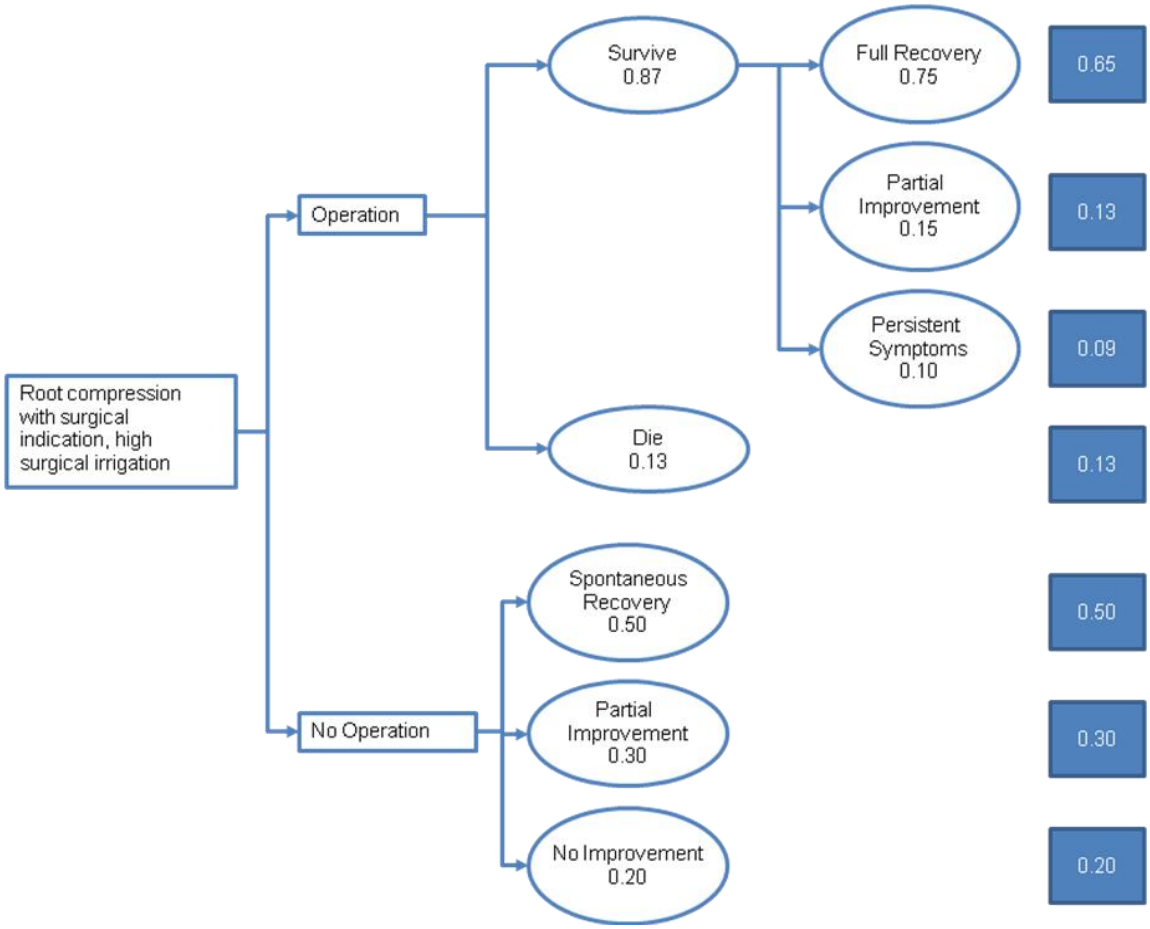

Figure 2-1: Example of a decision tree.

Amongst decision support tools, decision trees have several advantages. In particular, decision trees:

- Are simple to understand and interpret. People are able to understand decision tree models after a brief explanation.
- Require little data preparation. Other techniques often require data normalization, dummy variables need to be created and blank values to be removed.
- Have value even with little hard data. Important insights can be generated based on experts describing a situation (its alternatives, probabilities, and costs) and their preferences for outcomes.

- Use a white box model. If a given result is provided by a model, the explanation for the result is easily replicated by simple math.
- Can be combined with other decision techniques.
- Perform well with large amounts of data in a short time. Large amounts of data can be analyzed using personal computers in a time short enough to enable stakeholders to take decisions based on its analysis.

The use of decision trees also has some limitations:

- There are concepts that are hard to learn because decision trees do not express them easily, such as XOR, parity or multiplexer problems. In such cases, the decision tree becomes prohibitively large.
- Decision-tree learners create over-complex trees that do not generalize the data well. This is called over-fitting.

There are some techniques to avoid or reduce these limitations. It is worth saying here that the difficult part of the decision tree technique is drawing up a diagram such as the figure above from the written description of the problem. Once that has been done the solution procedure is quite straightforward.

2.4 Coordination and prioritization of executable tasks

In the framework of reactive intelligence, coordination of executable tasks can be treated as a special case of general decision making. We can treat a task as a reactive component, with some of its sensors used to pass “control” commands (like *start*, *pause*/*suspend*, or *stop*) and with some of its actuators used to emit “vital life signals” of the task (i.e., the information on the current state of the task that is needed to make coordination decisions among several tasks). To coordinate several tasks, we can add a reactive component (the coordinator) whose sensors are connected to the “vital life signals” of the coordinated tasks, and whose actuators are connected to the “command” sensors of the coordinated tasks. We can then implement the process of making coordination decisions as a suitable set of reactions of the coordinator. Prioritization is just a special case of coordination.

Typical coordination techniques can be split into imperative and declarative ones. Among imperative techniques, besides the common low-level techniques based on a detailed case analysis manually programmed in an imperative language, we mention finite state machines (Subsection 2.4.1) and process algebras (Subsection 2.4.2). Declarative techniques are based, in one way or another, on the concept of an expert system that employs some kind of a reasoning engine (Subsection 2.4.3). Finally, in Subsection 2.4.4, we present the language 3APL as a concrete example of a typical declarative technique.

2.4.1 Finite state machines

FSMs (Finite State Machines) are one of the oldest approaches to specifying control. They are still very popular, mostly due to their extreme simplicity. An FSM is just a directed graph whose nodes represent control states, and whose edges represent possible transitions between control states. Transitions are caused by so-called actions. Depending on whether actions are associated with transitions or with states and whether the actions are atomic or have some internal structure, there are several variants of FSMs. The most typical are Rabin, Moore and Mealy machines.

The main drawback of FSMs, as specification formalism, is that they do not scale up for complex control (since larger graphs tend to become chaotic). A partial solution to this problem is hierarchical FSMs, where states can contain FSMs, and this allows to structure complex graphs as a hierarchy of smaller FSMs. Also, larger FSMs can be built from smaller FSMs, using some operators from Process Algebras (see Subsection 2.4.2). Both approaches are combined together in Statecharts [Harel1987] which is currently considered to be the most advanced notation for FSMs.

2.4.2 Process algebras

The formalism of process algebras is based on the idea that a complex, control dominated, computational task can be specified using algebraic methods known from mathematics, i.e., applying operators to constants and variables. The constants represent simple processes (like sending a message, or performing some other atomic action), variables represent other (possibly unknown) processes, and operators represent the ways simple processes can be combined to form more complex processes. Typical operators are sequential composition, parallel composition, choice, iteration and recursion.

There are three leading formalisms for process algebras: CCS (Calculus of Communicating Systems), CSP (Communicating Sequential Processes) and ACP (Algebra of Communicating Processes).

Practical usefulness of process algebras is rather limited. They mostly serve as input formalisms to various verification tools that help to establish the correctness of complex control, and thus are mainly used in the context of safety critical systems where errors are particularly costly (either in money, or human life).

2.4.3 Exact and approximate reasoning

A common technique to implement a complex control is to use a reasoning engine that deduces a desired response from an actual stimulus. This reasoning engine can be arranged as a reactive expert system that periodically scans the sensors, adds their current values to the knowledge base as known facts, and derives from the augmented knowledge base some conclusions for the actuators. Traditionally, such reasoning is based on exact reasoning methods that involve classical two-valued logics (like Horn Logic [Poole1998], Description Logic). We argue that it should instead employ approximate reasoning methods based on fuzzy [Klir1995], rough [Polkowski2003] or probabilistic [Pearl1988] deduction methods. Such approximate methods can cope much better with the ambiguities and uncertainties which are typical in the environments that involve human beings. The ideal would be to combine various exact and approximate reasoning engines, but there are fundamental semantic difficulties with such combined approaches.

Making robotic devices smarter by applying approximate reasoning methods faces two main challenges: knowledge acquisition and coherent integration of various techniques. Knowledge acquisition is a general problem in expert systems, no matter if classical or approximate reasoning methods are involved. First, the knowledge relevant for a particular application must be isolated (both general and domain-specific). Second, the knowledge must be somehow encoded in an underlying

formalism. In case of approximate methods, the additional challenge is to acquire various numerical values, like probabilities and degrees of truth, involved in approximate deduction algorithms.

Probabilistic reasoning addresses the problem of dealing with uncertainties while fuzzy and rough reasoning deal with ambiguities of concepts. Human intelligence can cope quite well with both, so it seems natural to demand similar abilities from the Florence robot, and thus to base its control on the combination of various reasoning techniques. However, it is not obvious whether a coherent combination of the three main techniques for approximate reasoning is feasible, or even possible, at all. The known attempts to combine fuzzy with rough [Polkowski2003, pp 479-490] is rather complex and computationally costly. Combining probabilistic with fuzzy is even more difficult due to a deep mismatch of foundations: probabilistic methods are intensional in nature while fuzzy methods are extensional [Pearl1988, pp 4-13].

2.4.4 3APL

3APL [3APL2010] is a programming language for implementing cognitive agents (the acronym 3APL stands for Artificial Autonomous Agents Programming Language). It provides programming constructs for implementing agents' beliefs, goals, basic capabilities (such as belief updates, external actions, or communication actions) and a set of practical reasoning rules through which agents' goals can be updated or revised. 3APL programs are executed by an interpreter that deliberates based on the cognitive attitudes of that agent. Figure 2-2 presents the conceptual model for a Belief, Desire and Intentions (BDI) interpreter.

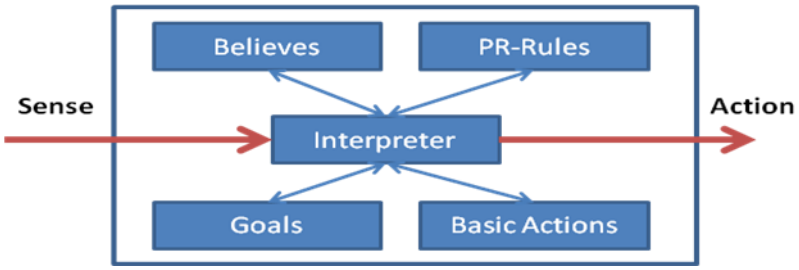

Figure 2-2 Conceptual model for a BDI Interpreter

A 3APL application is composed of four knowledge bases, which are executed by the interpreter during run-time. These knowledge bases are:

- CAPABILITIES: contains a description of the basic actions an Agent can implement.
- BELIEFBASE: beliefs describe the situation the agent is in. The beliefs are represented by a first-order domain language (Prolog).
- RULEBASE: rules are templates for building plans during execution time. A plan is a sequence of basic actions, tests on the belief base or abstract plans.
- GOALBASE: the goals of the agent denote the situation the agent wants to realize. The goals are represented by a first order domain language (Prolog).

2.5 Fixed, adaptable and adaptive control

The techniques described in Subsection 2.3 mostly deal with fixed control, where all decisions are pre-programmed in advance, during the system's development. In the parlance of "reactive intelligence" this means that a system can only perform reactions

from a fixed set, decided by its developer. Systems with fixed control have rather limited adaptation abilities since they can properly work only in the environments predicted by its developer. In order to extend the adaptation abilities of a system, the system must allow for its behaviour to be extended, or modified, during execution time.

Formally, two forms of adaptation are considered: adaptability and adaptivity. A system is adaptable if it can be adapted by a user to her needs. A system is adaptive if it can adapt itself to the user's needs or, more generally, to the environment in which it works.

In what follows, we give an overview of some typical techniques used to extend the adaptation abilities of a system. In short, the main approaches to adaptable control are configuration and programming. On the other hand, the main approaches to adaptive control are machine learning and evolutionary computing.

As stated in DoW (see the description of Task 4.1), the control software of the Florence system should take into consideration two important aspects:

- Integration of human intelligence in decision making (humans in the loop)
- Adaptive decision making (learning of user models)

Note that the first aspect is covered by adaptable control, while the second one is covered by adaptive control.

2.5.1 Adaptable control

A user can adapt a system in essentially two ways: by configuring and by programming [Saerbeck2009].

Control is configurable if it is designed as parametric. The configuration process is simply setting the parameters (say, by choosing some preferences from a list of available options). For example, the internal control of a TV set is configurable with respect to many parameters like the current TV channel we watch, volume level, brightness, and so on. We configure the internal control of a TV set by pressing various buttons on a remote controller.

Control is programmable if it is equipped with some engine that allows a user to extend/modify the pre-built control program itself (and not only its parameters). A user-programmable control allows integrating the human intelligence in decision making, by harvesting human intelligence via end-user programming. Of course, the challenge is to find both the programming metaphor and a user interface which is so easy and natural that the user does not perceive it as programming. A typical approach to end-user programming involves some sort of scripting, either directly in some textual notation, or more often, using a fancy GUI.

2.5.2 Machine learning

Machine learning is a vast subject of research devoted to finding methods and concrete algorithms that allow a system to adapt to its environment by learning some properties of the environment. Machine learning algorithms can be organized into taxonomy, based on the desired outcome of the algorithm:

- Unsupervised learning deals with the problem of clustering data. The goal is to determine some structure in a given unstructured data set, and use this

structure to partition the set into clusters. The clusters can then represent semantic concepts.
- Supervised learning deals with the problem of classifying data. This is somewhat dual to the problem of clustering data. A supervised learning algorithm is given data that has already been clustered. The goal is to determine the characteristic properties of the clusters, to be able to properly classify new data (i.e., assign it to one of the known clusters). To achieve this, the learning algorithm has to generalize from the presented data to unseen situations in a "reasonable" way.
- Semi-supervised learning combines both labelled and unlabeled examples to generate an appropriate function or classifier.
- Reinforcement learning learns how to act given an observation of the world. Every action has some impact in the environment, and the environment provides feedback in the form of rewards that guides the learning algorithm.
- Transduction tries to predict new outputs based on training inputs, training outputs, and test inputs.

Among many approaches to learning algorithms, the most widely used are: neural networks, Markov chains, belief networks, influence diagrams, and support vector machines.

2.5.3 Evolutionary computing

A reactive system can try to adapt to a new environment via a trial-and-error process that mimics biological evolution. Evolution by natural selection can be thought of as a search through the space of possible chromosome values. In that sense, an evolutionary computation is a stochastic search for an optimal solution to a given problem. The search process is arranged in the following way [Engelbrecht2002, pp. 121-182]:

First, an initial population of individuals is chosen. Then, in an iterative process, new populations are generated from the previous ones, by reproducing a selected portion of individuals. The individuals are selected on the basis of their fitness for the solution of the problem. The reproduction is governed by cross-over or mutation, or both. The process finishes when an individual is found that is considered to be “fit enough”.

There are several paradigms of evolutionary computing, most notably genetic algorithms, genetic programming and evolutionary programming.

Genetic Algorithms (GAs) concentrate on the representation of chromosomes and the cross-over operations that combine chromosomes of existing individuals to produce off-springs.

Genetic Programming (GP) is a specialization of genetic algorithms. Similar to GAs, GP concentrates on the evolution of genotypes. The main difference is in the scheme used for representing individuals. Whereas GAs use string representation, GP represents individuals as trees that encode executable programs. The aim of GP is to evolve the programs. For each generation, each evolved program is executed to measure its performance within the problem domain. The performance is then used to quantify the fitness of that program.

Evolutionary Programming (EP) differs from GAs and GP in that EP emphasizes the development of behavioural models and not genetic models: EP is derived from the simulation of adaptive behaviour in evolution. The evolutionary process consists of finding a set of optimal behaviours from a space of observable behaviours. For this purpose, the fitness function measures the "behavioural error" of an individual with respect to the environment of that individual. More specifically, EP differs from GAs and GP in that no crossover is implemented, and only mutation is used to produce a new population.

2.6 Decision making in other EU projects

Among many EU projects related to Ambient Assisted Living (AAL), we investigated 23 projects mentioned in "Funded Projects for AAL Call 1" (see [AALcall1-2010]). Unfortunately, none of the projects provides enough publicly available information to judge if complex decision making is involved at all, and what methods, if any, are used to solve this problem.

We also investigated several generic open-source robotic platforms like RoboCare ([RoboCare2010]) and Robot Operating System ([ROS2010] and [Quigley2009]). None of the robotic platforms offers components specifically dedicated to general high-level decision making. Instead, they offer components that support lower-level decision making, mostly for sensor fusion as used in context enhancement. Typical examples are computer vision and its use for obstacle avoidance during navigation.

The above observation should not be surprising. It just hints that the vast body of research on decision making did not result in any generic, and universally applicable, techniques because high-level decision making is an application specific problem.

2.7 Conclusions

This section presented an overview of several issues involved in decision making for complex hardware/software systems, without giving detailed descriptions of various techniques used in this context. Instead, the presentation concentrated on several important dichotomies like centralized versus distributed, imperative versus declarative, synchronous versus asynchronous, exact versus approximate, adaptable versus adaptive, and so on.

There were two main reasons for taking this particular approach. First, the descriptions of particular techniques can nowadays be relatively easily found and studied in isolation, due to their proliferation on the Internet, while the synthetic picture we tried to present is not so readily available, and thus seems to us more useful. Second, as already noted in Subsection 2.6, decision making is application specific and at this time we do not know in full detail the real application the Florence project will build (the state of the art activity of WP3 is running in parallel to the investigation of scenarios in WP1), so we cannot decide on which particular techniques will be needed. Understanding the various dichotomies we presented will be beneficial when designing the architecture for the "decision making" subsystem in the subsequent stage.

3 Context enhancement

The decision making process of adaptive applications adjusts its behavior to the situation in which such applications and users interact. According to [Dey1999], context is the information that characterizes the situation of an entity (e.g., person, object, and place) that affects the user-application interaction. Sensory information obtained from disparate sources such as physical sensors, history data, a-priori knowledge about the environment and human inputs can be considered as context. Usually such raw contextual information is unsuitable to feed to the decision process of adaptive applications due to its poor quality and semantics. Therefore, raw contextual/sensory information should be combined to yield high quality contextual information that is, in some sense, better than the originally sensed information (e.g., more accurate, more complete, more dependable, etc). This is called sensor fusion or context enhancement, which is done continuously by monitoring and processing the raw contextual information. Based on this viewpoint, this section focuses on “context enhancement” and assumes “context monitoring” to be a commonplace process within context enhancement. Note that, in general, (enhanced) context information can be logged and used as input for high-level “monitoring applications”, which differs from context monitoring.

Most techniques, methods and issues mentioned for decision making in Section 2 are also applicable and relevant for context enhancement. To avoid redundancy, this section elaborates mainly upon some not already mentioned aspects that are specific to context enhancement or upon some aspects that are likely to be further studied or developed within the Florence project.

3.1 Overview

Context enhancement is achieved through a set of reasoning techniques to enhance context information semantics and quality of context-information, the result of the latter is expressed with Quality of Context (QoC) indicators. Fusing low level sensory information can yield context information with a high abstraction level; for example, it can lead to detecting user activities such as sleeping, reading, eating, etc. On the other hand, QoC encompasses a set of indicators to characterize context information. These indicators are generic and independent of the semantic interpretation of the context data. They, however, build a minimal ontology allowing common interpretation of the context in one important aspect of quality [Zimmer2006]. Important QoC indicators include precision, accuracy, freshness, spatial resolution, temporal resolution, and probability of correctness (accuracy) [Sheikh2008]. The following subsections elaborate on the generic functions and issues of context enhancement.

3.1.1 Generic functions

For context enhancement, the authors of [Snoeck2006] distinguish a set of functionalities, as illustrated in Figure 3-1. In this framework, context with formally defined semantics is referred to as knowledge, otherwise it is called data. Such a distinction between data and knowledge is inspired by [Ackoff1989] and [Alavi1999]. The functions of context enhancement are:

- Collect: to gather (raw) context from context sources (e.g., sensors) whose outputs represent a narrowly scoped aspect of the state of the world. These

sources are typically sensor readings or knowledge that was explicitly entered by an end-user, e.g., a scheduled appointment.

- Interpret: to assign meaning to data whose semantics are unknown to the system. This transformation of data to knowledge requires a model that describes how to map specific types of data inputs to knowledge. A model can be considered as an abstract representation of how the world works, e.g., which location coordinates represent someone's home.
- Infer: use a model to derive additional knowledge and conclusions from the obtained/known knowledge. For example, derive that a user is in a meeting based on his schedule, his physical location and the number of people in the vicinity.
- Learn: to automatically learn the models used for interpretation and inference. The existing models can be updated incrementally or rebuilt from scratch repeatedly. For robust model generation, human intervention may be required.
- Store: to store the required data, knowledge and models for future use and record.

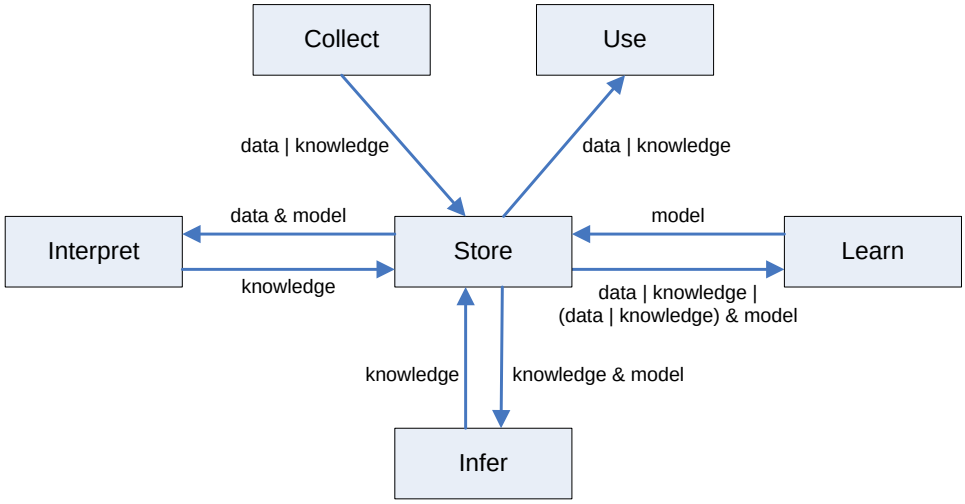

Figure 3-1: A typical reasoning architecture where '|' and '&' indicate a logical OR and AND, respectively (source [Sno2006]).

3.1.2 Important issues

Nowadays context aware services face rapid and vast changes in their deployment, usage and provisioning settings. To cope with such changes, automatic learning is required to build reliable and objective models for reasoning about the context. The learning can be offline or online, depending on whether the data required for learning is gathered before or during system operation. Learning methods can be classified as supervised learning, unsupervised learning [Bousquet2004], semi-supervised learning [Chapelle2006] and reinforced learning [Sutton1998], see also Subsection 2.5.2.

Many real-world problems intrinsically have uncertain properties. Example sources of uncertainty are: measurement errors of sensors, a limited capability of reasoning methods, ambiguity in the way knowledge is expressed by humans, etc. Uncertainty is generally the result of some information deficiency caused by conflicting, redundant, incomplete, imprecise, or contradictory information [Klir1999]. Reasoning about uncertainty may provide context-aware services with clues about the correctness of

the derived knowledge, allowing the receiver, being either human or other reasoning component, to choose between alternatives.

Proactive services are a subset of context aware services that require predicting of future contexts. The ability to predict future events (i.e., values of variables) enables a context-aware service to anticipate upcoming changes in the user's environment and hence to become proactive, see Subsection 3.2.4 for more information.

Context aware services must be delivered on time in order to achieve commercial success and provide a good user experience, i.e., to have a high degree of user acceptance. Therefore, reasoning techniques have to be fast enough by, for example, distributing the computational load over available resources. This requires manageability of the computational complexity and dealing with the scalability problem.

3.2 Focus areas

This subsection aims at describing those functionalities, methods and tools that are most relevant to the Florence services. The aspects described in this subsection will most likely – depending on the scenarios and use-cases to be indentified – be investigated (or used) in the project.

3.2.1 User activity detection

The automatic recognition of human task execution is studied in various research fields, including data mining, logic, probabilistic reasoning and computer vision. To offer useful services that are tailored to a user's implicit needs, the detection of high level goals and intentions is required. This is the focus of the plan recognition community [Carberry2001] and [Snoeck2007]. Plan recognition for inferring tasks has already been applied to real-world problems. For example, the Independent LifeStyle Assistant recognizes intentions of elderly in their home environment [Haigh2006] based on high-level sensor information as input. Chai et al [Chai2005] uses sequences of Wi-Fi signal-strength measurements to infer tasks and to recognize multiple simultaneous tasks. Sukthankar et al [Sukthankar2005] infers a soldier's behaviours from a set of military manoeuvres in multiple steps using low-level sensor data. There are some theoretic research works on plan recognition. For example, the approach of Avrahami et al [Avrahami2006] and the PHATT system of Geib et al [Geib2006] combine symbolic and probabilistic reasoning methods to deal with plan mixing, plan abandonment and missing observations. The approach of [Avrahami2006] includes constrains in the plan model but it requires observing the first action of the plan, possesses limited expressiveness of its plan models, and assumes observations to be consistent, certain and high-level. The PHATT system, however, does not utilize constraints and it requires knowing the likelihood of top-level goals and missing observations a-priori to rank hypotheses.

Sensor fusion techniques can be used for constructing a model that adequately represents human behaviour, which in turn can be used for recognising human tasks.
To this end, one can consider bottom-up statistical reasoning or top-down approaches based on domain knowledge. Statistical reasoning does not require explicit input from a domain expert, and can be designed to adapt to changing environmental circumstances and deviations in human behaviour. However, the mapping of statistically determined behavioural patterns into useful classifications that could

trigger supportive actions is not trivial. Subsection 3.2.2 describes one approach for addressing this issue, namely by asking the user to assist in creating such a mapping. In contrast, top-down approaches require a fully specified behavioural model a-priori, which limits their flexibility with respect to changing circumstances and requires labour intensive alignment of models with typical behaviours of individual users by a domain expert.

A method that is commonly used for the structured analysis of human tasks in a top-down approach is Hierarchical Task Analysis (HTA). HTA provides a suitable basis to describe human behaviour. The HTA descriptions, however, do not translate directly into models that can be automated. It is for further studies to translate HTA descriptions into formal models, and to combine this top-down approach with bottom-up reasoning with probabilistic models. A possible direction will be to explore the applicability of the Hierarchical Hidden Markov Models (HHMMs) [Fine1998] for representing human tasks described in HTA formats. HHMMs for the recognition of human tasks have not been studied yet (to the best of our knowledge). Computational complexity of the HHMM based methods is a major issue, which can be addressed by solutions like the heuristics in [Avrahami2007].

Example user tasks and activities that can be relevant for Florence servicers are sleeping, walking, cooking, TV watching, etc.

3.2.2 Interactive learning

Supervised learning provides a framework for learning a classifier in machine learning traditionally. Supervised learning assumes that there is a training set consisting of data points x (from some set X) and their labels y (from some set Y). The objective is to find a function f: X → Y that will accurately predict the labels of data points to be seen in the future. This framework, however, is a poor fit for many modern learning tasks because it assumes that all training points are labelled. Because the gathering of a sufficiently large and representative set of labelled training data requires a lot of manual effort from the user a-priori, only few labelled samples will be available in practice. One solution to this problem is to use an interactive learning approach, in which the learner automatically detects data patterns that occur frequently, and asks the user to label only these. This form of learning requires less effort from the user, which is also spread out over a larger timeframe.

The main question is how many training points are needed to learn an accurate classifier? In some situations labelling data is expensive especially if labels must be explicitly procured when there is abundant unlabeled data. In such cases the learning algorithm can actively query the user (or the trainer) for labels; starting with a large pool of unlabeled points. It must then strategically decide which data points it wants to label so that the labelling cost (e.g., the number of queries) remains limited within the budget. This type of iterative supervised learning is called *active learning* (see [Langford2009] for a tutorial). The learner chooses the examples to query their labels based on aggressive and adaptive strategies.

Since the learner chooses the examples, the number of examples to learn a concept can often be much lower than the number required in normal supervised learning. To decide which labels to query, the learner can select the query points at random. This so-called aggressive option, however, yields the same label complexity as supervised

learning. A better option is to choose the query points adaptively, e.g., start by querying some random data points to determine rough decision boundaries, and then gradually refine the boundary estimates by specifically querying points in its immediate vicinity.

Many algorithms are already available, most of which resemble the aggressive and adaptive sampling strategies. Although these are promising in experimental studies, there is a risk with this approach: the algorithm might focus on unimportant or even invalid examples and thus the set of labelled points no longer reflects the underlying data distribution. Recently considerable progress has been made on developing active learning algorithms with good statistical properties and labelling complexity; see for example [Beygelzimer2009].

In supervised learning the system learns, e.g., a classification model, by asking explicit questions from the user during system operation. Active learning tries to minimize the burden on users and as such it becomes an important approach for the Florence project attempting to integrate human intelligence with machine learning.

3.2.3 User localization

To monitor users and their movement patterns, especially inside (intelligent) homes as in the case of the Florence project, it is necessary to locate and trace (elderly) users. Robots may have built-in mechanisms for real-time localisation of themselves, i.e. for their own navigation inside these homes. But they cannot simply be dispatched to follow users wherever they go.

User localization can be done “indirectly” through tags or devices that are carried by users or “directly” through sensors in the infrastructure that capture physical characteristics of users (e.g., image, electromagnetic field, etc). The methods that indirectly detect users can further be divided into two categories depending on the role of the device carried by users. The device can “send beacons to infrastructure sensors” or “collect beacons sent by the infrastructure”. In the former, the beacons are sent either periodically or in response to some triggers from the infrastructure and the localization of devices is done by the infrastructure (mainly by a server somewhere in the infrastructure). In the latter, the device collects the beacons sent by the infrastructure and the localization can be done either on the device locally or in the backend remotely. In the last case of backend localization, the device should relay the collected information to the backend. Beacon collecting devices must be smart devices capable of running the localization applications. Beacon sending devices, however, are simple devices/tags just capable of reacting periodically or in response to external stimulus.

A good reference over existing indoor LBS (Location Based Systems) [Gu2009], providing an overview of some products and academic solutions. This section summarizes the main out of the shelf products for indoor localization because developing such solutions is considered out of Florence scope. These products are categorized according to the wireless technology used for localization signalling. Further, Table 3-1 indicates the localization method of most of these products according the operational modes defined above (i.e., detecting users directly/indirectly and the user devices being beacon-sending/beacon-collecting).

Ultra Wide Band (UWB) based: PLUS system from Time Domain [TimeDomain2010] is capable of delivering sub-meter accuracy and working through walls. Ubisense system [Ubisense2010] is capable of delivering 3D location with 15cm accuracy.

WiFi based: AeroScout system [AeroScout2010] uses WiFi tags that send beacons to the infrastructure. Ekahau system [Ekahau2010] is capable of delivering one meter accuracy. Other WiFi based products are Zebra enterprise solutions [Zebra2010] and Airetrak [Airetrak2010].

Bluetooth based: Zunith system [Zunith2010] is proximity based delivering room level accuracy. Topaz system [Topaz2010] from Tadlys also delivers room level accuracy.

RFID based: Active RFID from Axcess [Axcess2010] can send beacons periodically or in response to requests, having a typical accuracy of 10m. WhereNet system [WhereNet2010] from Zebra enterprise solutions has a typical accuracy of 2-3 meters.

Infra-Red (IR) based: Star Gazer [Star Gazer2010] from Hagisonic is capable of delivering 2cm accuracy in a small operational area. The technology is mainly used for robot navigation. Other IR based products are: Firefly [Firefly2010] from Cynernet with an accuracy of 3.0mm, OPTOTRACK PROseres from Northern Digital Inc [Northern2010] with an accuracy of 0.1-0.3mm, IRIS_LPS [IRIS2010] with an accuracy of 16cm (the latter seems to be non-commercial).

Ultra Sound (US) based: Sonitor system [Sonitor2010] delivers room-level accuracy. Sonitor signals do not penetrate solid walls and does not require line of sight signalling (thus it is possible to track hidden objects in a room). Others US based systems are Active Bat, developed by researchers of AT&T Cambridge, and Cricket, a mixed ultrasound and RF system. The latter two are not commercial products.

GPS (Global Positioning System) based: Hitachi Ltd. IMES concept [Forssell2009] extends the GPS to indoors by installing GPS base transivers in indoors. The system achieves an accuracy of about 10m (room or shopping-area levels).

Vision based: Eagle Vision system [Eagle2010] uses stereo cameras on the ceiling and head counting to detect the number of people in a waiting line, to forecast the waiting time, and to detect the bottlenecks in flow of people.

Table 3-1: localization methods of some localization products.

product	wireless technology	detecting users		device role w.r.t. beacons	
		indirectly	directly	Sending	collecting
PLUS	UWB	x		X	
Ubisense	UWB	x		X	
AeroScou	WiFi	x		X	
Ekahau	WiFi	x			x
Zunith	Bluetooth	x			x
Topaz	Bluetooth	x		X	
Axcess	active RFID	x		X	
WhereNet	RFID	x		X	
Star Gazer	infrared	x			x
Sonitor	ultrasound	x		X	
IMES (Hitachi)	GPS	x			x

Eagle Vision	vision		x		

The Indoor localisation is an active research area. Most work done here rely on WiFi, RFID, IR, Bluetooth and US technologies [Lim2006, Luca2006, Bargh2008]. The Florence project should seek efficient methods to fuse sensor information obtained from heterogeneous sensors (i.e., those in home infrastructure, BAN (Body Area Network) of users and robot sensors) to determine users' locations. Ideally, the solution must be robust and flexible and not depend on a specific sensor (i.e., to have graceful degradation if a sensor is missing). A key challenge here is to differentiate between users. Identification of users is easily possible in those approaches where users carry tags/devices, i.e., based on the indirect approach. In the direct approach, however, user identification is almost impossible.

3.2.4 Event prediction

A proactive system needs to predict upcoming events and react to them accordingly. We distinguish two types of predictions for upcoming events:

- Predicting the next outcome of an event, given a set of observed outcomes of that event and perhaps also based on the outcomes of some associated events.
- Predicting the moment in the future that a certain event occurs, given a set of observed timings of that event and perhaps also based on those of some associated events. This is called the first passage time prediction problem.

Many methods can be used for such event prediction problems. To this end, two important categories of methods are: Markov model based methods (e.g. data compression methods like PPM (Prediction by Partial Matching) [Cleary1984], CTW (Context Tree Weighing) [Willems1995]) and kernel density methods. The former can be used for the next outcome prediction problem and the latter can be used for the first passage time prediction problem. For example, Burbey and Thomas [Burbey2008] have used the PPM method for predicting users' future locations and Peddemors et al [Peddemors2008] have used kernel density functions for predicting the moment that a mobile device is going to experience out-of-range wireless connectivity.

Event prediction required by Florence applications and services will be based on "heterogamous" sensory data. Hereto applying Markov model and kernel density function based methods to heterogeneous sensory data will be a promising research direction. Universal source coding methods [Willems2006] can further be used for building prediction models that adaptively tune to an individual user or a specific environment.

3.2.5 Probabilistic reasoning methods

There are many algorithmic techniques for context enhancement. These techniques can be categorised as probabilistic methods (e.g., Bayesian theory, Dempster-Shafer theory) least square methods (e.g., Kalman filtering) and artificial intelligence methods (e.g., Fuzzy logic, neural networks, genetic algorithms). In Section 2 some relevant techniques were described. This section only describes two probabilistic techniques of Bayesian Networks and Dempster-Shafer theory in more detail because we will explore them for realizing the Florence context enhancement functionality.

Bayesian Networks (BNs) or also known as belief networks belong to the family of probabilistic graphical models [Ben-Gal2007]. BNs model knowledge about an uncertain domain by representing the set of the related random variables and their conditional independencies via a Directed Acyclic Graph (DAG). Each node in the graph represents a random variable. The edges between the nodes represent probabilistic dependencies among the corresponding random variables and are drawn by arrows between nodes.

For example, an edge from node X_i to node X_j represents a statistical dependence between them: a value taken by variable X_j depends on the value taken by variable X_i (i.e., variable X_i "influences" X_j) . Here X_i is referred to as a parent of X_j and, similarly, X_j is referred to as the child of X_i. One can further define the sets of "descendants" – the set of nodes that can be reached on a direct path from the node, or "ancestor" nodes – the set of nodes from which the node can be reached on a direct path. Such conditional dependencies are estimated by using statistical and computational methods.

Specifying a BN firstly requires defining the structure of a DAG by its nodes and the directed edges between them. This is called the qualitative part of the model. Secondly, specifying a BN requires specifying the parameters, i.e., Conditional Probability Distribution (CPD), at each node. This is called the quantitative part of the model. The CPDs at every node depend only on its parent nodes. For discrete random variables, this conditional probability for each of the feasible values is defined for each combination of values of its parents.

A BN specifies the JPD (Joint Probability Distribution) of the underlying random variables in a factored form. Using this BN, one can answer all possible inference queries by marginalization, i.e. summing out over "irrelevant" variables. There are often two types of inference:

- Predictive support for node X_i (also called top-down reasoning). This relies on evidence nodes that are connected to X_i through its parent nodes. For example, given the observation that a person uses a new uncomfortable chair installed at the person's office, what is the predictive support for the belief that the person will suffer from a backache.
- Diagnostic support for node X_i (also called bottom-up reasoning). This relies on evidence nodes connected to X_i through its children nodes. For example, given the observation that a person suffers from a backache, what is the diagnostic support for the belief on new uncomfortable chairs installed at the person's office.

In the examples above a statistical dependence is assumed between uncomfortable chairs and having a backache.

Usually the BN is unknown and it has to be learnt from the data. This so-called BN learning problem can be stated as: Given the training data and prior information (e.g., expert knowledge, casual relationships), estimate the graph topology (i.e., network structure) and the parameters of the JPD in the BN. Learning the BN structure is considered a harder problem than learning the BN parameters. Further, the case of partial observability when nodes are hidden or when data is missing is considered another obstacle for Bayesian learning.

Dempster-Shafer theory is a mathematical theory of evidence [Shafer1976] and it can be considered as a generalization of Bayesian statistical theory [Wu2003]. Dempster-Shafer theory extend a proposition like “the user is A” to the union of two propositions called Belief(A) and Plausibility(A). Belief(A), being a number in [0, 1] range, quantifies all pieces of evidence that supports the proposition of “the user is A”. Plausibility(A), being another number in [0, 1] range, quantifies the evidence that do not rule out proposition "the user is A" (i.e., it quantifies the proposition “the user might be A”). Plausibility(A) is equal to or larger than Belief(A). Based on a sensor’s observation, the probability that “the detected person is A” can be represented by a “confidence interval” of [Belief(A), Plausibility(A)]. For each proposition such as “the user is A”, the Dempster-Shafer theory gives a rule for combining different sensors’ observations or for combining a sensor’s subsequent observations, see [Wu2003] for details. By associating "belief" and "plausibility" to a statement, the Dempster-Shafer captures key features of the human perception-reasoning process [Wu2003]. As such, the Bayesian approach can be seen as a specific case of the Dempster-Shafer approach, which does not deal with "belief" and "plausibility" quantitatively.

3.3 Context enhancement in other EU projects

The CompanionAble project [CompanionAble2010] is concerned with the synergy creation and semantic integration of Robotics and Ambient Intelligence technologies to provide for a care-giver's assistive environment. The project has produced some scientific outputs related to context enhancement, which are briefly described in the following. In [Muller2008] the authors address the issue of detecting the intentions of users who want to initiate an interaction with the robot. The robot must be able to estimate who is willing to interact before the person reaches the robot. Since there is limited camera resolution, the required information is extracted from a person’s movement trajectory. From trajectory sequences some features are derived to classify the interaction interest. The selection and combination of useful features is done by an information theoretic approach based on the Mutual Information and Joint Mutual Information with respect to the user’s interaction interest. In [Istrate2008] the authors consider a remote monitoring system to detect a distress situation in the case of an elderly person living alone. The system monitors the environment sound and identifies everyday life sounds and distress expressions in order to make an alarm decision. In [Einhorn2009] the authors use Extended Kalman filters (EKF) to process a sequence of images taken by a single camera mounted in front of a mobile robot for reliable obstacle detection and avoidance. In [Rougui2009] the authors present an event detection system that detects sounds (distress expressions and vocal commands) in presence of background noise caused by e.g. water flushing, a hair dryer or a vacuum cleaner. In [Medjahed2009] the authors present a method for detecting the distress context for a telemonitoring system based on fuzzy logic. For elderly people living alone in the home environment, the system uses a new multimodal reasoning system called EMUTEM that gathers and fuses three types of sensors: (a) a set of microphones for remote monitoring of the acoustical environment of the elderly, (b) a wearable device to measure physiological data like heart rate, activity and posture to detect an eventual fall, (c) a set of infrared sensors to detect the presence of the person and his standing posture. The fuzzy logic decision module allows detecting older person’s distress events and location trustfully.

The Companionable is involved in designing a telemedicine platform that integrates communication services (like email, instant messaging, visiophony, agendas, alarms) and specific medical tools (like cognitive exercises, patient monitoring) with robotics

and smart homes [Simonnet2008a] and [Simonnet2008b]. The new platform will be the result of several telemedicine projects that have been started since 2000.

Within the Persona project [Persona2010] the authors of [Braun2009] have developed a capacitive proximity sensor for fall detection based on wire antennae. The sensor setup consists of several capacitors on the floor that can measure capacitance changes caused by the presence of a human body on the floor of a room. In addition to fall detection, the sensor can be used for detecting forgotten medications and respiration monitoring from distance. The authors of [Stikic2008] explore two different techniques for significantly reducing the required amount of labelled training. One of the techniques is based on active learning, whereby the system actively asks which data the user should label.

3.4 Conclusions

This section presented an overview of context enhancement functionality, which can be instrumental for specifying the architecture and design of WP3 related components.

Based on our insight and experience at this early stage of the project execution, a number of key context enhancement functions and techniques were identified and the corresponding related work was summarised. These functions include:

- Detecting user activities/tasks, which is recognized relevant to most of candidate Florence service scenarios, enables detecting the state of users in being sleeping, walking, etc.
- In order to put users in loop, interactive learning aims at utilizing human intelligence, whenever appropriate, for facilitating the automatic labelling of for example user activities and tasks.
- Any proactive system such as Florence requires timely prediction of upcoming events. Using history information, event prediction is required to estimate the time or the type of the most likely event to occur.
- Localization of elderly users in indoors is another key functionality that is required for realizing candidate Florence service scenarios.

These context management functionalities can be realized using various techniques such as those mentioned in Section 2. Particularly, probabilistic methods are considered to be well suitable for the setting of the Florence project.

There are a number of (ongoing) projects related to service robotics and AAL services. Based on available documents and publications, it is recommended to follow the work of the CompanionAble and Persona projects on context enhancement closely.

4 Context management platforms

Many context management platforms – mostly for non-robotic settings – have been developed recently. This section gives an overview of those platforms that are considered suitable for supporting the context-aware services of Florence project because of their advanced features and because of the project partners' in-house expertise and knowledge about these platforms. This section starts with a generic introduction about context aware systems and then gives an overview of non robotic and robotic context management platforms.

4.1 Introduction

As illustrated in Figure 4-1, the intention of a context aware application is to use context information to improve a service or to create new services. Common examples are location based services which for instance use the current location as context information to provide information about places around the user. According to [Dey1999], "a system is context-aware if it uses context to provide relevant information and/or services to the user, where relevancy depends on the user's task."

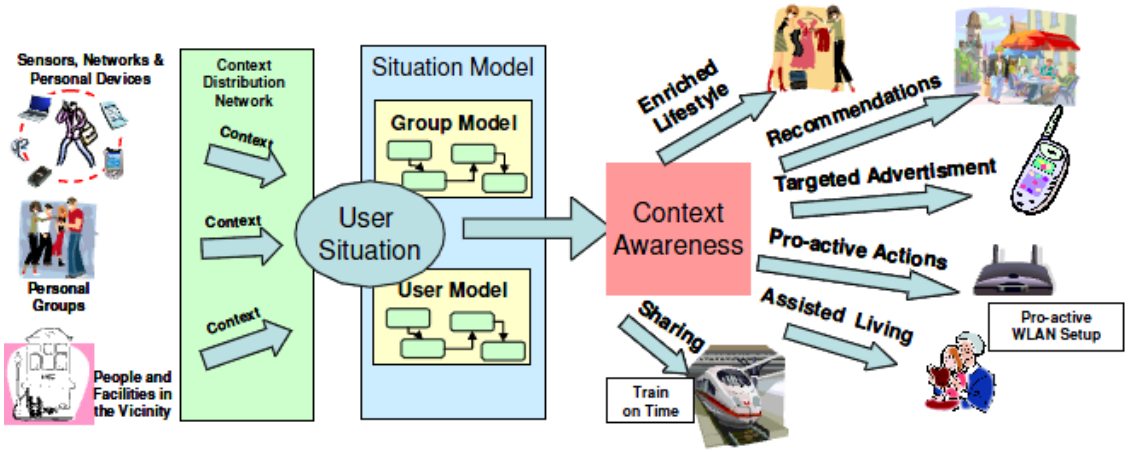

Figure 4-1: an illustration of context aware system.

The task of a context management platform is to act as middleware between context sources and a context aware application. Therefore it should provide a unified interface to access the attached context sources. In the following subsections some context management platforms are described.

4.2 Non-robotic platforms

There are quite some context management platforms developed for non-robotic settings. This subsection gives an overview of most relevant one to the Florence project.

4.2.1 Context management framework of NEC

The Context Management Framework (CMF) of NEC (NEC Europe Laboratories) was initially developed for the MAGNET Beyond project [Magnet2010] as part of the 6th Framework Programme. Furthermore, the development was continued during the

SPICE project [Spice2010] in the 6th Framework Programme and driven further during the 7th Framework Programme in the OPEN project [Open2010].

Aside from EU Projects, the CMF of NEC was demonstrated at the Integrated Systems Europe fair in Amsterdam in a cooperation between NEC Europe Laboratories, NEC Display Solutions and Instore Media Sweden, a digital signage software vendor. The NEC CMF was used as middleware to show the possibilities of context aware digital signage.

The NEC CMF is OSGi based [OSGi2010] and runs on most devices that support a Java virtual machine, ranging from regular computers to Windows Mobile and Android mobile phones. The programming interface is based on XML-RPC, and its main concepts are currently being standardized as part of the Context API in the Open Mobile Alliance (OMA) Next Generation Service Interface (NGSI) [OMA2010]. The NEC CMF has been designed as a distributed middleware that facilitates access to contextualized sensor information, i.e. sensor information that is related to a real world entity. It consists of multiple *Context Agent*s that can be grouped in clusters and clusters of clusters. As shown in Figure 4-2, a context agent consists of three main components:

- Retrievers which provide context information from various data sources like sensors or web services,
- Processing Units which can be used to operate on available context information,
- A local storage component.

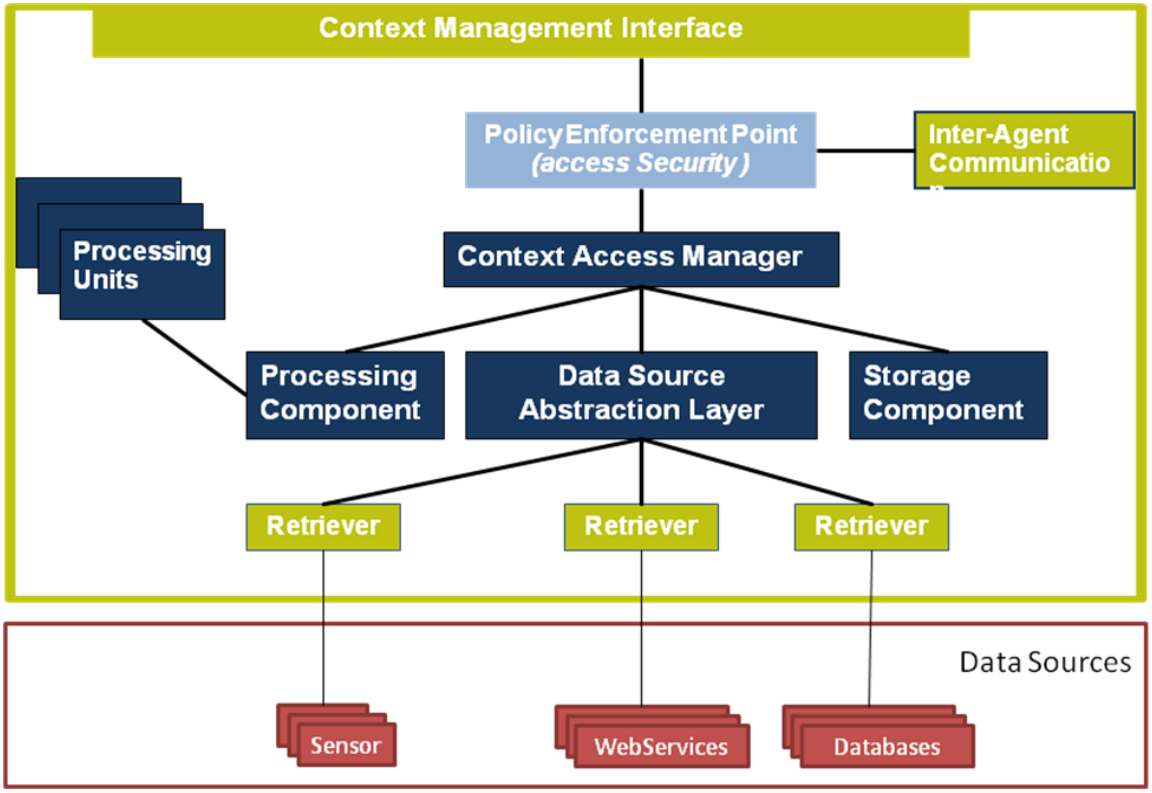

Figure 4-2: Context Agent of NEC CMF.

Each agent has access to information provided by other connected agents and is able to enforce policies that define which applications and other agents can access which entity/attribute pairs. Each cluster of Context Agents, see Figure 4-3, has a central Context Agent called Context Management Node which is responsible for distributing the index information about the accessible context information through the whole cluster. Additionally the Context Management Node is used for the cluster to cluster communication.

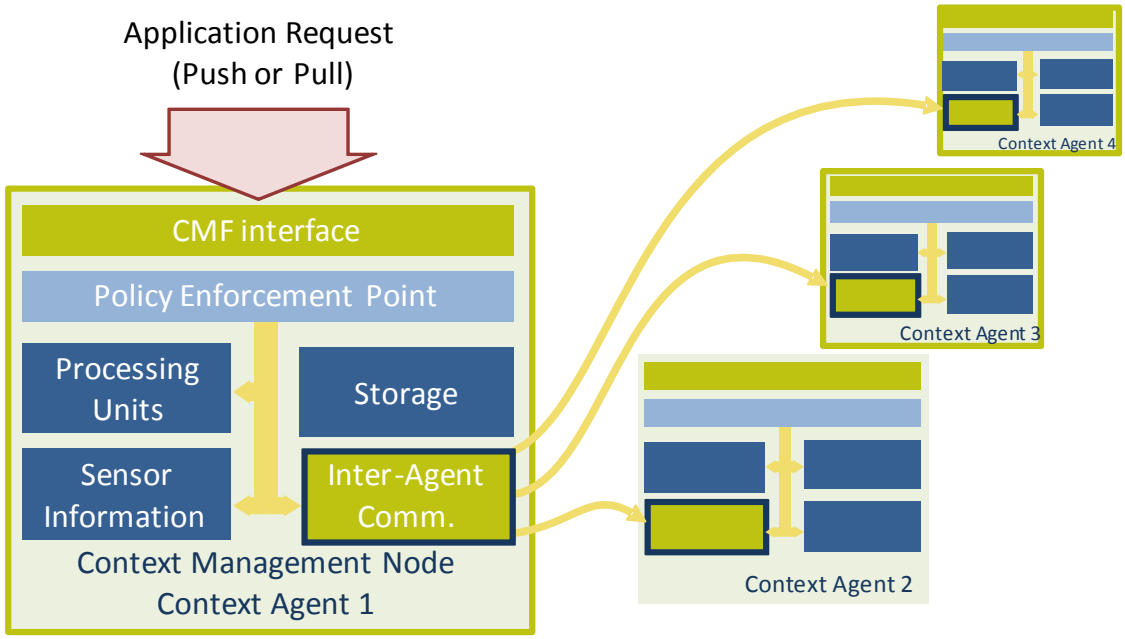

Figure 4-3: A cluster of Context Agents.

Information is accessed using a declarative Context Access Language (CALA) that is based on an entity/attribute data model. CALA provides query, subscribe, insert, delete, and update operations that operate on a whole set of *Context Agents*. Thus applications can access information in a fully declarative way and do not need to address individual agents directly. They are able to use a unified interface to access information generated by any sensor. CALA does support type based and identifier based access to entities. Entities can inherit various attributes and also multi value attributes.

Through the abstraction concepts of a retriever, the NEC CMF is capable to support almost every type of data source. The retrievers that are already available for these data sources include:

- Home automation/smart home area retriever for sensors like: door open / window open sensor, controllable power socket sensor (on/off), remote control (used as manual sensor), weight scale, light pulse counter (to measure on an energy meter, barometer sensor, humidity sensor, smart-meter for electrical power consumption, temperature sensor, brightness sensor, and pressure mat.
- Localization retriever for sensors like GPS position, RFID events and information sensor, Bluetooth proximity sensor, nearby WLAN sensor, Ultrasonic Indoor Positioning System (UIPS) sensor (developed by NEC China-Labs)

- Movement retriever for sensors like: gesture event sensor (SUN SPOT), acceleration sensor, inclination (tilt) sensor, and computer vision sensor for audience measurement and tracking.
- Generic retriever for sensors like: Siafu Context Simulator, HTTP Bridge for third party sensor integration, REST Retriever, Serial Port Retriever, IMS presence sensor.
- Miscellaneous retriever for sensors like: barcode event sensor, calendar information sensor (MS Exchange).

4.2.2 Context management framework of Novay

The Context Management Framework (CMF) of Novay has first been developed within the AWARENESS project [Wegdam2005] which ran from April 2004 until April 2008 and was part of the Freeband program. The goal of the project was to design and create an infrastructure that would assist in deploying and managing context sensors for the sake of creating context aware (mobile), pro-active applications. Key features of the infrastructure are:

- Reasoning to higher level and better quality context information,
- Efficient exchange and distributed processing of context information in dynamic and pervasive environments,
- End-user controlled handling of the privacy aspects.

The CMF has also been used and further developed within other European projects like the SPICE (IST-FP6), AMIGO (IST-FP6) and iNEM4U (FP7) project.

Figure 4-4 shows the layered architecture of the AWARENESS project, where Novay CMF is represented by the infrastructure and ECA (Event-Condition-Action) components. On top there is the application layer encompassing context aware applications and sensors providing application-specific context information to other applications. The ECA engine realizes application behaviour as delegated by the application. This involves interaction with the infrastructure layer to become informed of context events, invocation of application-specific actions according to the delegated behaviour. Privacy enforcement ensures the enforcement of privacy policies over releasing the personal context information.

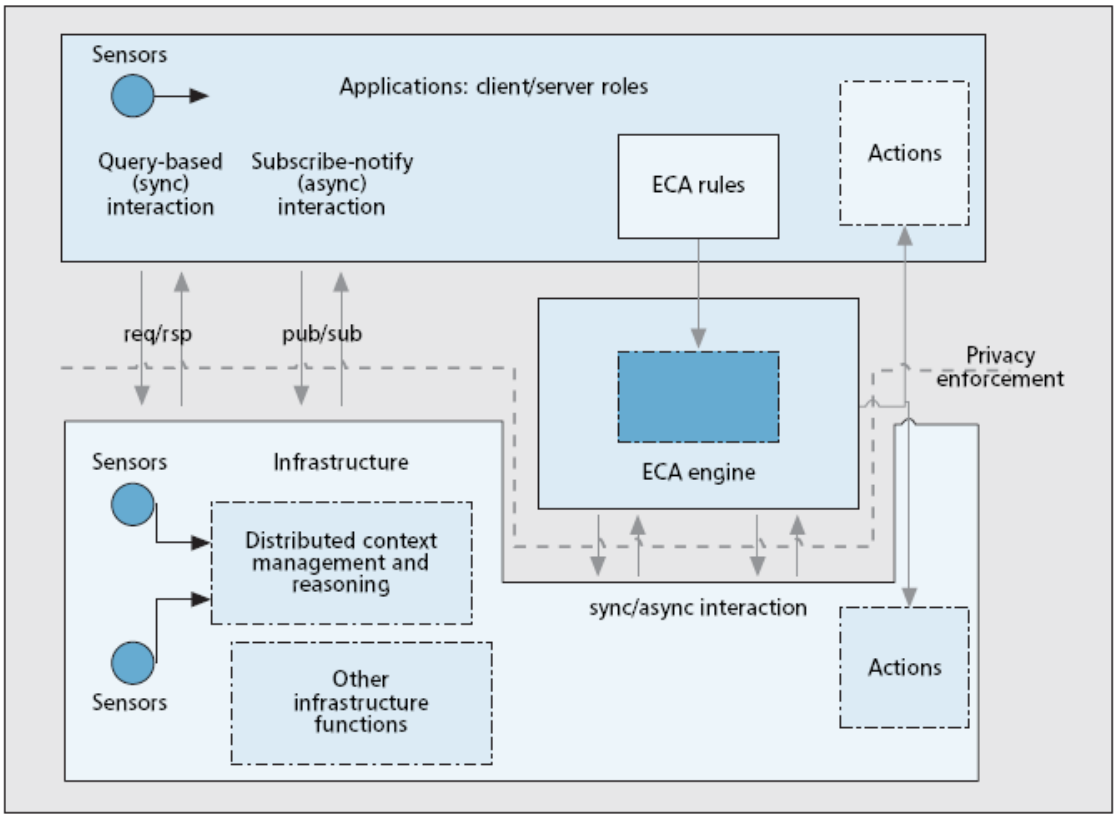

Figure 4-4: layered architecture of the AWARENESS project (source [vanSinderen2006]).

Distributed context management and reasoning constitutes the core of the infrastructure layer, see Figure 4-5. In addition to context collection and reasoning, this functionality provides other functions such as identity management, storage, and discovery. For more details the interested readers are referred to [vanSinderen2006].

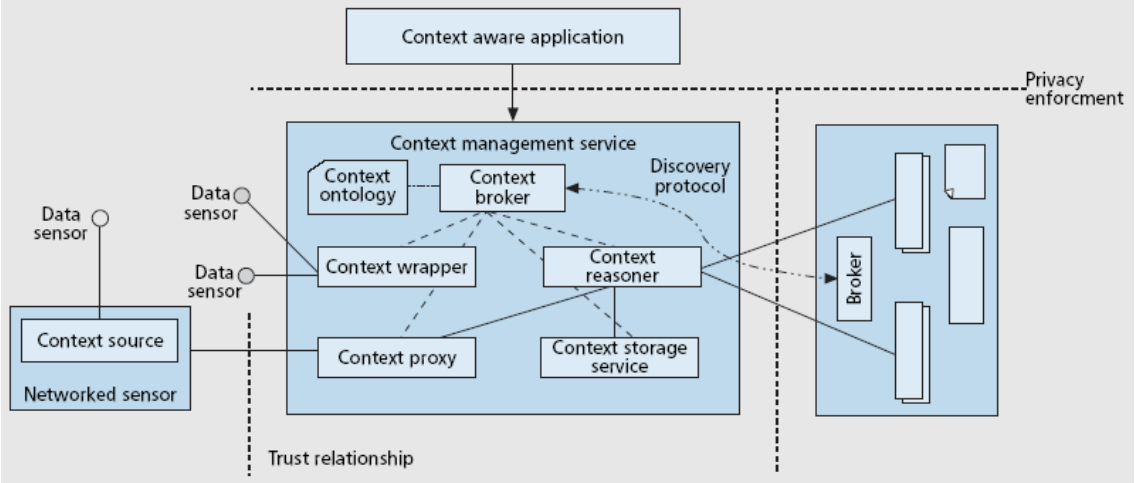

Figure 4-5: Architecture of Novay CMF (source [vanSinderen2006]).

The generic framework of CMF relies on Remote Procedure Calls (RPC) for communication between the various components. These generic components need to be hosted in a container. Container implementations are available for Symbian,

Windows and Linux, all are implemented using Java. Within a container components can find each other, create new components and invoke methods using the RPC mechanism as long as the components implement the proper interfaces.

Upon starting a container some default components will be initialised, based on a configuration file. One typical default component would be the context broker. It acts as a gatekeeper to the outside world and can offer different access points, like TCP or Bluetooth sockets, to which clients can connect. Upon successful authentication, which is handled by the access interface, a client can get references to other interfaces like for instance a context source interface which can supply a certain type of context information.

The current implementation of Novay's CMF supports various kinds of RPC protocols like XML-RPC over TCP or Bluetooth, Web Services and WS-Discovery. The RPC layer hides the underlying protocol, so any component implementing a component interface will have support for all underlying RPC protocols. CMF containers can be created as separate Java processes, but can also be initialized as OSGi bundle.

4.2.3 Xensor

Xensor ([Xensor2010] and [terHofte2007]) is a research instrument for social sciences to capture and give insight in objective data about human behaviour and their (social) context (location, proximity, communication, etc). It features a modular context gathering platform developed for Windows Mobile smart phones. Developers can create their own modules for communicating with different types of sensors and log the data gathered by the sensor. These modules can run in the background for as long as the service is running or they can be triggered by incoming events like an incoming phone call or a button press. A number of example modules are available, for example:

- GPS Sensor: logs location in a certain interval
- Audio Sensor: logs audio recordings upon a key press
- Bluetooth Sensor: scans for nearby devices
- Calendar Sensor: logs Microsoft Exchange data
- Experience Sensor: can ask the user for input at given or random intervals

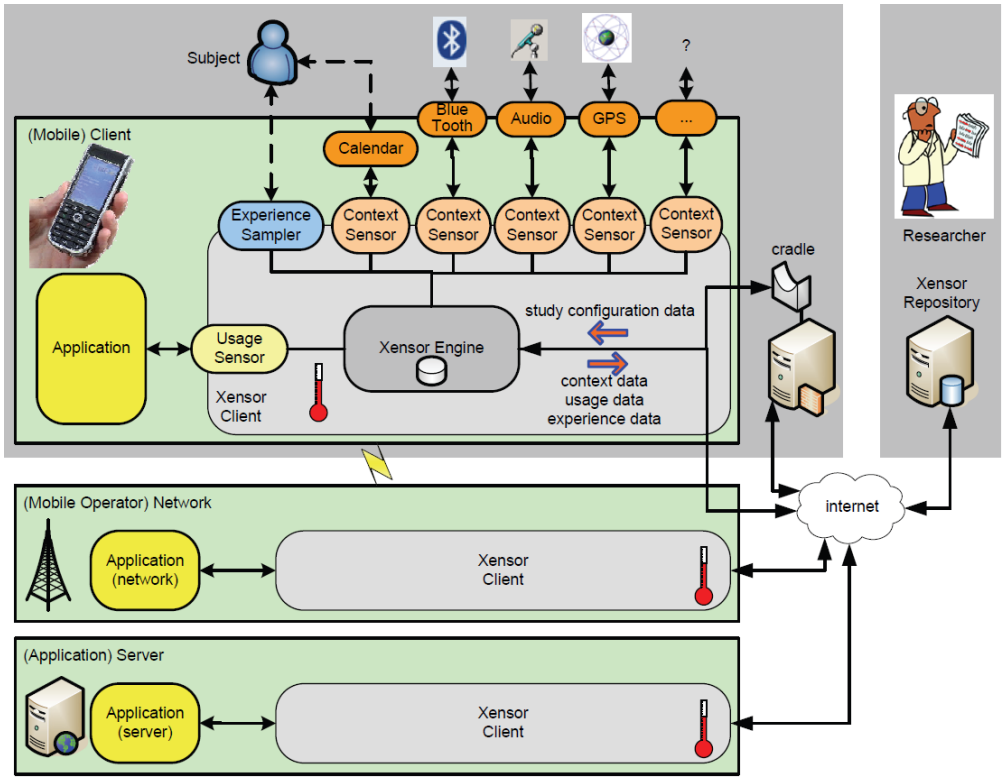

Figure 4-6: Xensor platform for Windows Mobile smart phones (source [terHofte2006]).

Data is being stored securely on the storage card of the smart phone itself. The data can be obtained either physically by acquiring the storage card or by uploading the data to a central repository which can be done at a set time, say for instance during the night, or upon putting the device in a cradle.

4.2.4 IYOUIT

IYOUIT ([Boe2008] and [IYOUIT2010]) is a social context gathering platform that runs on Nokia Series 60 smart phones. It has been initially developed as part of the European MobiLife project (FP6). Due to its success it has been further developed by DoCoMo Euro-Labs and Novay after the project had finished. It gathers context information from users which can be shared amongst friends. Certain types of information are gathered automatically like a user's location (based on either GSM Cell triangulation or GPS) and a user's mood (which has to be entered by the user itself).

IYOUIT focuses on four aspects:

- Share: community-based context sharing,
- Life: guidance and life support,
- Blog: automatic contextual blogs,
- Play: context-driven games.

IYOUIT has its own distributed Context Management Framework (CMF) to gather context data in a flexible way and to reason about the combination of various context streams. It includes various context sources that provide, for example, the whereabouts of buddies, scanned Bluetooth and WLAN beacons, local weather,

photos, sounds, observed products, books or messages [Boehm2008b]. The CMF architecture of IYOUIT is shown in Figure 4-7.

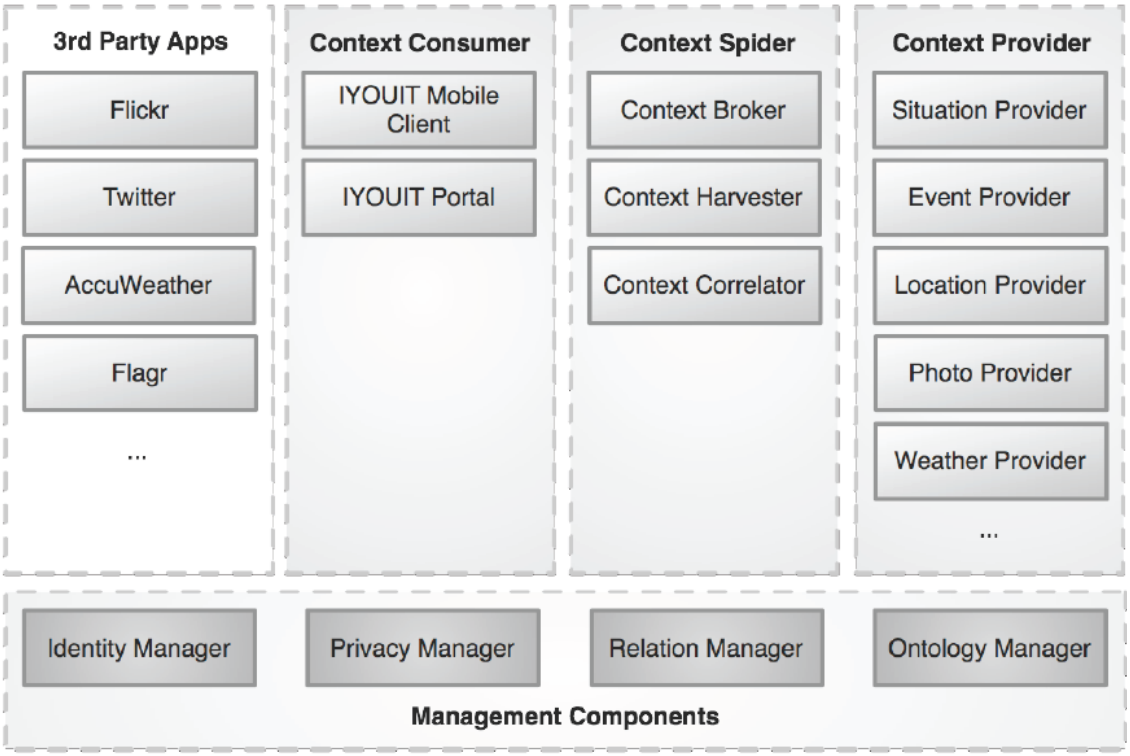

Figure 4-7: The CMF architecture of IYOUITY (source [Boehm2008b]).

Main components of IYOUIT CMF, as shown in Figure 4-7, are:

- Management Components encompass an Identity Manager to ensure a secure authentication of entities; a Privacy Manager to enable access control to sensitive information, an Ontology Manager to allow for the usage of domain specific knowledge, and a Relation Manager to represent and reason about the social network of users.
- Context Spiders include a Context Broker to discover all registered components, a Context Harvester to retrieve context data from multiple distributed components, and a Context Correlator to align distinct context streams based on their temporal correlations.
- Context Providers encapsulate the basic context data sources at a quantitative level and aggregate / abstract them to a qualitative level. Each context provider is designed to realize domain specific services.
- Context Consumers & 3rd Party Applications are context consumers on mobile clients and Web portals of 3rd parties to use IYOUIT CMF core components through the Privacy Manager and Identity Manager. Example 3rd party applications are: Flickr, Twitter and Google Earth.

4.2.5 PERSIST

The PERSIST project [PERSIST2010] intends to develop a Personal Smart Space (PSS) with devices that will be able to integrate context information and reasoning capabilities to provide tailored and enriched services to users, keeping in mind user preferences, intentions, etc. It is based on a Context Awareness Platform that is

capable of collecting information from different providers, aggregating/storing contextual information, and proving contextual information to requesters [PERSIST2008].

In order to maximise interoperability between different providers, the platform uses "Context ML", which is an xml-based language providing a meta-model for the representation of the context information (providers and consumers). Figure 4-8 presents the High-level architecture proposed for the PERSIST project. It consists of the following layers:

- Layer 1 – Devices. The devices, depending on their processing and networking capabilities, can implement part of the PSS or simply interact with the PSS framework.
- Layer 2 – System Run-Time Environment. This is an abstraction layer between the underlying device operating system and the PSS software in order to be platform independent.
- Layer 3 – Overlay Network Management. It provides the PSS architecture with a Peer-to-Peer (P2P) management and communication layer.
- Layer 4 – Service Run-Time Environment. It provides a container for the PSS services. It supports service lifecycle management features and provides a service registry, as well as, a device registry. Moreover, it allows for service management in a distributed fashion among multiple devices within the same PSS.
- Layer 5 – PSS Framework. This is the core of the PSS architecture. Its functions include discovering and composing PSS and 3rd party services, as well as, managing context data and user preferences. It also supports automatic learning of preferences and inference of user intentions. This information, together with data from recommender systems, enables the proactive behaviour of the PSS platform. Grouping of context data and preferences, as well as, conflict resolution is also provided by the PSS Framework layer.

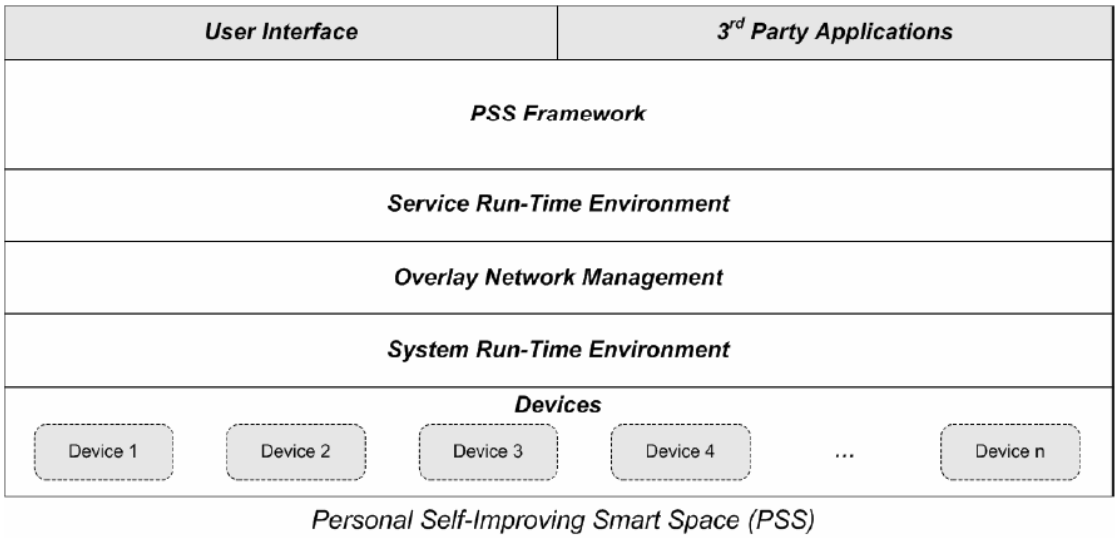

Figure 4-8: High-level architecture of the PERSIST Project (source [PERSIST2008]).

4.3 Robotic platforms

To best of our knowledge, there is no robotic platform targeted explicitly at context management. Context management is used as a tool for robotic platforms to enhance their function and make the robots more flexible and robust. Robots which use context management explicitly are Kompai, CompanionAble, PEARL and Carebot for example. They are further described in Deliverable 5.1. To be able to gather context information, all robotic platforms need a set of sensors like vision systems. These sensor data will then be processed and evaluated to react accordingly to the measured situation. A simple example would be a reaction to the illumination in a room with a light sensor which will turn the background light of a display on if it is getting dark.

4.3.1 SHARE-it

SHARE-it, [Share-it2010] and [Urdiales2008], is a FP6 project which concluded in 2009. Its main goal has been the development of an intelligent platform to support elders and people with mild cognitive impairment and mobility problems, allowing them to live autonomously in their preferred environment for a longer time.

The SHARE-it decision system is supported by a Multi-agent System, which extracts and combines information coming from different sources (e.g., environment, patient, autonomous wheelchair or mobility device) in order to infer high level information and takes autonomous decisions in order to help the elder. The Multi-agent System is distributed among the different components of SHARE-it as shown in Figure 4-9. Part of it is integrated into an OSGi framework, which also provides access to services outside SHARE-it.

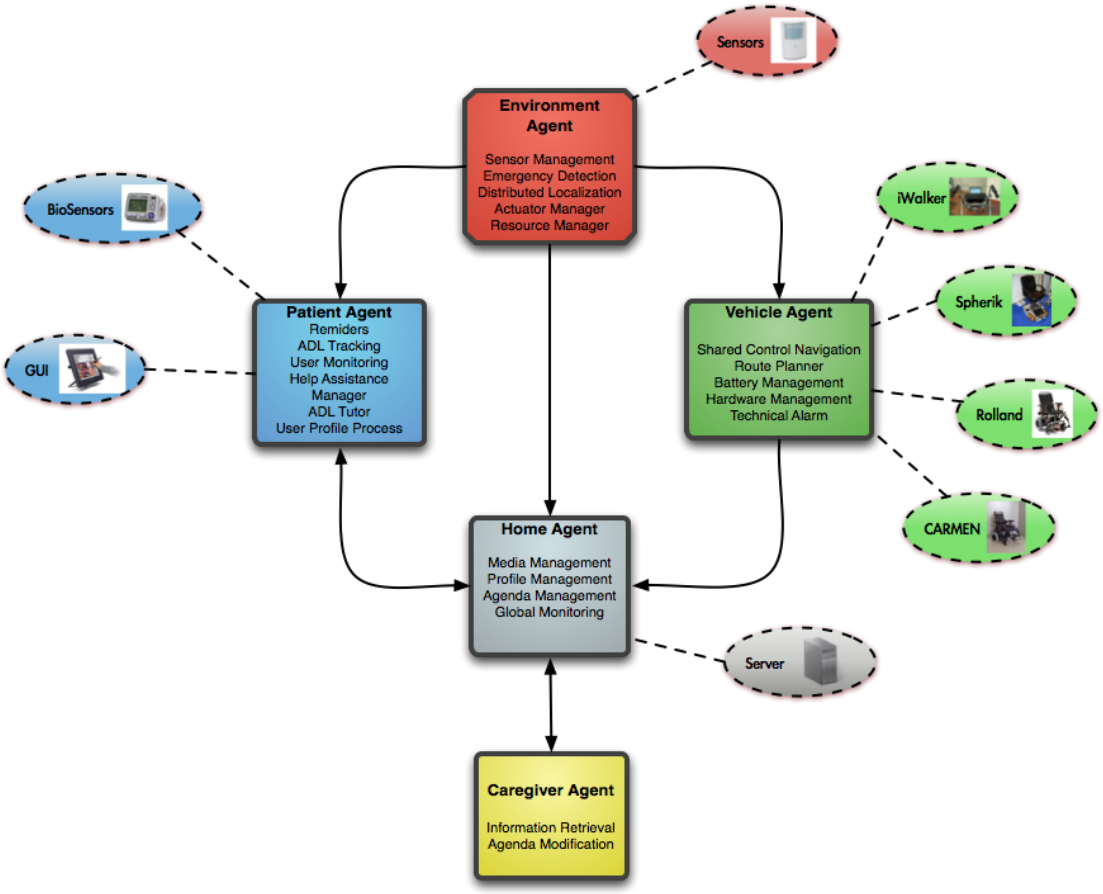

Figure 4-9: General Architecture of the SHARE-it Agent System (source [Rubio2010]).

The SHARE-it system is based on three different components:

- A set of mobility platforms which are in direct contact with the user (e.g., intelligent walker, robotic wheelchairs and an autonomous chair called Spherik). These adapt themselves to the user's assistance requirements through a tailored cognitive model that is based on the interpretation of biosensor data and on disability profiles assessed by a medical team.
- A home automation system which retrieves information about the environment of the user (e.g., localisation, opening and closing of doors, presence, lights).
- A distributed decision system which runs both on the mobility devices and in a central server. The server is placed inside the home of the user and is in charge of making dynamic decisions.

The communication between different parts of the system is carried out via a combination of SOAP-based web services and a GPL-licensed system, called "Distributed and Layered Architecture" (DLA) and developed at the University of Málaga [Pérez2002].

4.3.2 Genie of the Net architecture

Figure 4-10 shows Genie of the Net architecture, an example of an agent-based architecture for context-aware service management for controlling personal robots from [Roehning2001]. It is a software architecture that captures various processes, namely: collecting information from sensors and databases, recognizing the context

based on this information, choosing the relevant actions to serve the user on the basis of the recognized context, and performing the chosen actions. The architecture also allows utilizing other components of the smart environment.

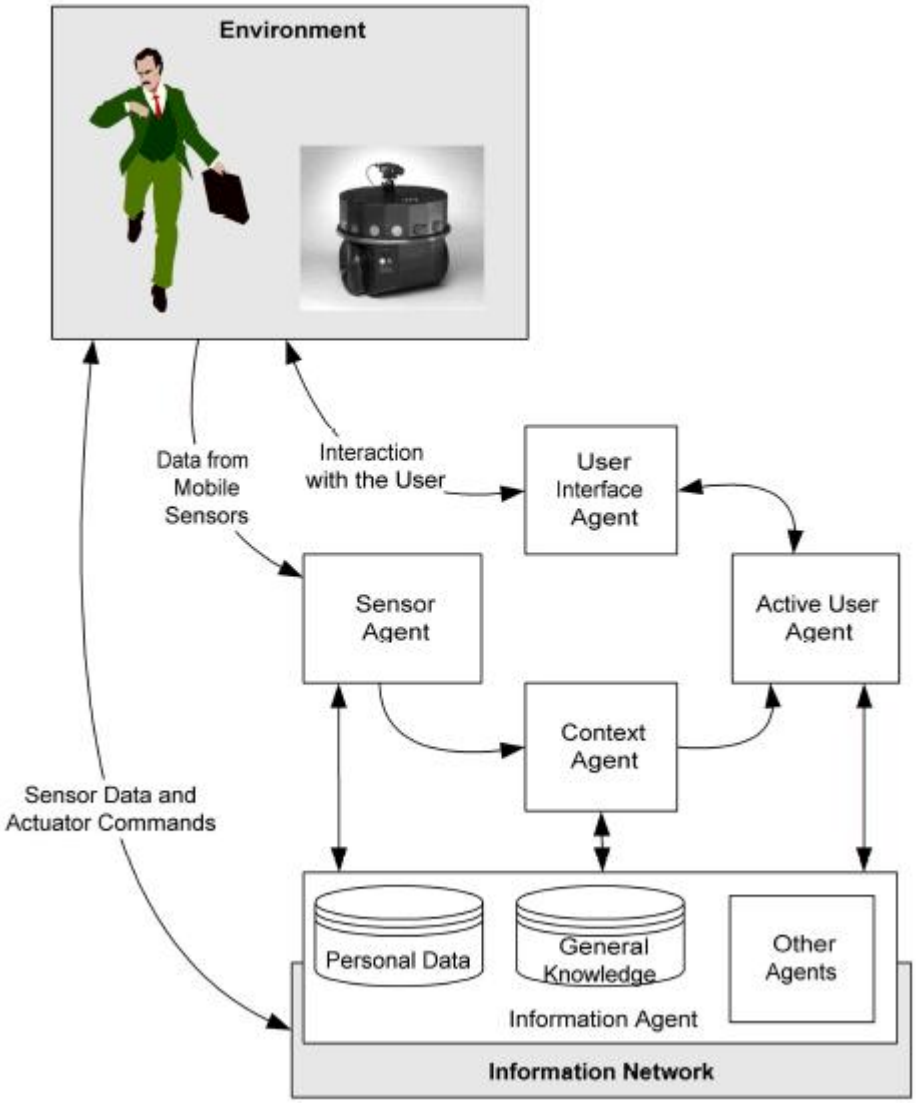

Figure 4-10: Genie of the Net architecture (source [Roehning2001]).

Main components of the Genie of the Net architecture shown in Figure 4-10 are:

- Sensor Agent that collects and pre-processes data from sensors.
- Context Agent that recognizes the user's context from this data and other available information.
- Information Agent that offers information for the other agents and stores the sensor data and a description of the recognized context for later use.
- Active User Agent that exploits the recognized context in choosing actions serving the user. In addition to scanning the context, the Active User Agent cooperates with the other agents and reasons based on a user behaviour model (e.g., the user's access rights, habits, plans, etc).
- User Interface Agent that presents information to the user on the request of the Active User Agent.

Personal robots inhabit the environment and are controlled by the Genie system. The objective is to enable robot and human co-operation [Roehning2001].

4.4 Conclusions

As mentioned in Subsection 4.3 there is no robotic platform targeted explicitly at context management to the best of our knowledge. Context management is used as a tool for robotic platforms to enhance their function and make the robots more flexible and robust. Hence the best approach for the Florence project will be to adapt one platform that fits the best to the needs of the Florence setting. In this section a number of candidate platforms are described that can be adopted for the Florence project.

In adopting and adapting any context management platform(s) for the Florence system, one must take the following issues into consideration:

- Various data sources will be needed for the Florence system. The ability to integrate all possible data sources is the main feature to be considered for selecting the context management platform in order to serve the needs of the robot and the sensor infrastructure.
- The Florence System will not only be distributed through the home of elderly but it will also connect health care providers with the home. Therefore, a fully distributed architecture for context management will be required.
- Various scenarios will require enhanced contextual information. Therefore the context management platform should have a flexible structure in order to integrate the required pre-processing components.
- In order to integrate additional data sources for enhancing the QoC (e.g. integrate an additional temperature sensor in a room) the context management platform should provide a context data structure that does not change the access routines when a new sensor source is added.
- Finally, there should be a unified way to discover and access the context sources.

5 Actuation triggering

Actuation triggering describes the process needed to execute a command that is given by users. The result of actuation is a physical effect in the real world. These physical effects can range from a light being switched on to a service being activated on the PC. Actuation triggering can be divided into two parts: controllable hardware which can be actuated, and management of the actuation which translates and routes an actuation request to the hardware. The state of the art for each of these will be described in the following.

5.1 Actuation management solutions/approaches

Actuation Management can be done in different levels of abstraction. This can go from very hardware specific single commands to fuzzy requests which require a set of single actuations. Over the years there have been several approaches to address these different levels of abstraction, some of which are described in the following subsections.

5.1.1 BACNet

BACnet was specifically designed for building automation systems. In January 2003 it was standardized by ISO as standard 16484-5 [Snoonian2003]. BACnet defines 23 virtual object types which represent typical functions of components in building automation systems such as analogue input, binary output, schedule and calendar that can be grouped together. BACnet systems allow remote control via the WWW from anywhere. For the internal communication between the sensors and actuators BACnet defines a set of message types. In the case that two different messages are being received by a single device, BACnet provides 16 levels of priority for command messages coming to rate the importance of a message or command.

5.1.2 LonWorks

Aside from the LonTalk communications standard for the communication within a LonWorks network, LonWorks does conclude the Neuron Chip from Echelon Corporation [Echelon2010]. The chip defines the communication and supports several types of media (fibre optics, coaxial or infrared). The network can communicate with the Web server i.LON 100, which bridges the LonTalk network and the Internet to allow easy monitoring. LonWorks provides a set of standard objects for applications on top of the network. These objects define input and output of devices analogous to BACnet standard object types.

In contrast to the priorities-mechanism used in BACnet, the last command always wins in LonWorks [Snoonian2003]. BACnet is compatible with LonTalk.

5.1.3 ZigBee

ZigBee [ZigBee2010] is especially designed for Wireless Sensor and Actuator Networks and therefore focuses on the issues relevant in the wireless world, for instance, low power consumption. ZigBee has two main concepts for Actuation. For the interoperability of devices a public application profile “Home Controls-Lighting”

(HCL) is provided for manufactures to build compliant devices. The interoperability of ZigBee Compliant Products is only guaranteed inside the ZigBee WSAN-islands and therefore technology bound. For actuation ZigBee provides control loops to map events to actuation. Devices are addressed directly.

5.1.4 Sensei project

In the Sensei project [Bauer2009] a concept for an actuation framework was developed that focused on the ability to translate high level actuation requests into low level actuations. Therefore Sensei introduced the following concepts, see also Figure 5-1:

- Actuation Task describes a desired physical effect.
- Resource User represents any kind of human or machine that can request an Actuation Task.
- Actuation Space is a shared memory space to store requested Actuation Tasks. In addition, the Actuation Space provides a high level abstraction to control physical actuators and allows dynamic binding to concrete physical actuators.
- Actuator Controller controls one or more Actuator Resources and is aware of which kind of Actuation Tasks it can process. An Actuator Controller watches the Actuation Space and looks for Actuation Tasks it can process. Once an Actuator Controller takes care of an Actuation Task it maps the high level request from the Actuation Task to low level commands for the Actuation Resource. Additional the Actuator Controller monitors the Actuation Task for any kind of changes
- Actuator Resource represents a unified interface to the physical actuator.

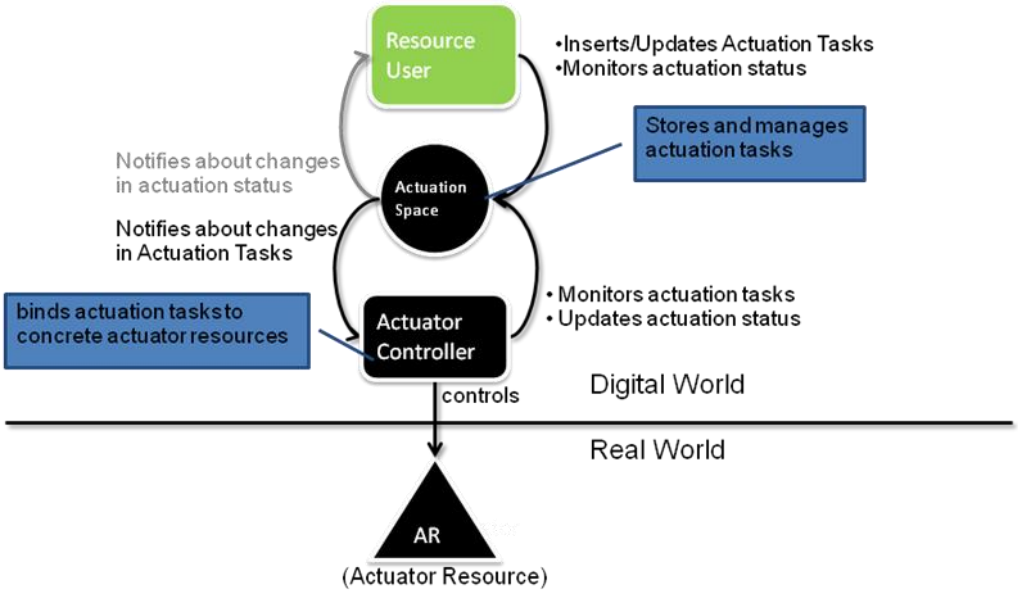

Figure 5-1: Illustration of Sensei concepts (source [Bauer2009]).

5.1.5 Home automation

In the Home Automation area actuation is a basic feature. Existing solutions in this area are mostly focused on directly interfacing sensors with actuators. Through the wide spectrum of available and used communication technologies and standards this interfacing is most times done by software and vendor specific. Aside from the problems in terms of interoperability, home automation systems do a low level management where actuators are directly addressed by their ID, Name or something similar. Normally sensor events get directly mapped to actuations. Although some systems provide a kind of scripting engine [IPSystem2010] to describe more complex tasks, the core concept is still a one-to-one mapping between a sensor event and an actuation. High level requests, like 'switch out lights in a certain room', would require a unique scripting for each room since every light has to be addressed directly. Some systems try to compensate this by a hierarchy for the actuators and sensors.

5.1.6 KUKA light weight arm

The notion of actuation triggering is indirectly addressed within the domain of service and assistance robotic arms, in which the robotic system is to work close to, or even with, a human. Security issues require developing solution ensuring a high level of security. The KUKA light-weight arm, developed in collaboration with the Institute of Robotics and Mechatronics, is such an example. On this system a monitoring loop is permanently controlling the occurrence of contacts with a person, to correctly adapt its dynamics.

5.2 Actuation hardware

This subsection provides an overview of the available devices that can be used as actuators. Furthermore, it briefly describes the functionality of such devices and one example of the services that are realizable with these devices.

5.2.1 Personal computer as actuation hardware

The PC (Personal Computer) itself can be used as actuator for the services/applications installed on it. Basically every application that provides an API to control it can be adapted to act as an actuator. An example would be the VOIP Skype application which provides an API to control the Skype VOIP service (e.g., receive call, block call, make call) and the Skype frontend (e.g., show incoming calls, open instant messenger or video call) [Skype2010]. With some interfacing between the API and the Actuation framework all these functionalities can be used as actuator.

A list of possible PC based services that could be used as actuators would also basically include any kind of communication service like email, SMS, or even social networks like Twitter or Facebook. For instance these services could be used to make an automated emergency call.

Additionally every periphery device for the PC which provides an API to control can be used as actuators.

5.2.2 Robot actuation and robot based actuation

We consider robots to be a collection of controllable peripheral devices like servo engines, wheels and cameras. These are in turn connected to a central controlling unit like a PC or an API-accessible microcontroller [ROS2010]. On a basic control level (e.g. move forward or backward) we can therefore consider that the actuation of a robot practically doesn't differ from that of a periphery device on a PC.

What makes the real difference is the possibility of combining rather simple services of a robot to deliver more complex services. For instance, the simple service of moving into a direction and using an attached camera could be combined into a service that follows the user. Depending on the setup of a robot, these more complex actuation services need to be generated and managed by the actuation management but could also be directly provided by the robot API.

5.2.3 Actuators for home automation

The Home automation area offers a wide range of devices which can be used for actuation. Aside from the standard controllable power socket there are actuators for almost every part in the home which can be automated. Some examples are described below.

Figure 5-2-(a) shows a heater regulator which replaces the original control element of a heater and allows for instance combined with a temperature sensor automatic regulation of the temperature in a room. Figure 5-2-(b) shows a window opener & closer which replaces the original handle of the window and allows automated opening and closing. A possible scenario would be to automate the ventilation of a room in a way that windows are only opened if no one is in the room. Figure 5-2-(c) shows a shutter controller. These controllers are available in different variations to ether put on top of an existing shutter cord or as full replacement.

a b c

Figure 5-2: (a) Heater control, (b) Window Opener, (c) Adapter for the EcoRoll shutter system (source [ELV2010]).

Figure 5-3 (a) and (b) show dimmable and controllable light sockets or switches. Figure 5-3 (c) shows a controllable door lock. Controllable door locks allow remote locking, unlocking and opening of a door. Those could be used to automatically open the door as soon as the elderly or a person of trust (family or emergency services for instance) approaches the door.

a b c

Figure 5-3: (a) dimmable light socket, (b) dimmable switch for direct wiring, (c) Homematic KeyMatic door lock system (source [ELV2010]).

In the home automation area different protocols like X10, FS20 or ZigBee are used to achieve communication. Based on these protocols there are also generic actuators which can be wired to any corresponding hardware, namely:

- Two state or three state switch (shown in Figure 5-4-(a), (b) and (c)).
- Switch key (shown in Figure 5-4-(d))
- Dimmers (shown in shown in Figure 5-4-(e))

All the predefined actuators are based on one or more of the above generic actuators.

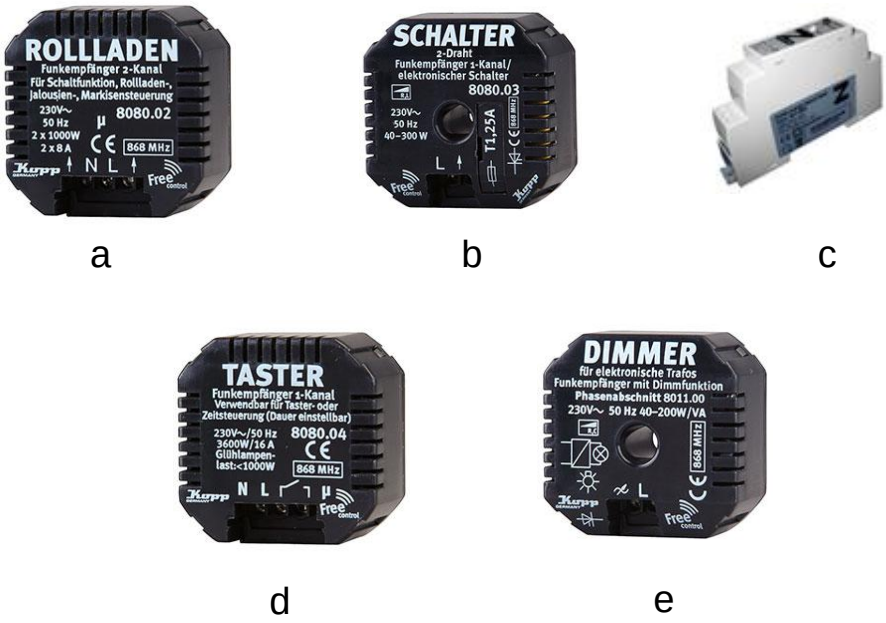

Figure 5-4: (a) 3 State Kopp Switch, (b)2 State Kopp Switch, (c) 2 State ZigBee Switch, (d) Kopp Switch Key, (e) Kopp Dimmer (source [ELV2010]).

5.2.4 KUKA robot arm

One originality of the KUKA LWR (Light Weight Robot) implementation is that the sensors are placed after the gear-box, allowing thus to obtain a very precise measurement and control of the joint torque, and, within a soft robotic context, a very fine control of collision so that the behaviour of the arm in such situations gives an impression of physical reaction and adaptation to contact [Albu-Schaffer2009]. Nevertheless, such a system does not enable to reduce the impedance of the manipulator, which is, through its high-frequency characteristics, directly linked to the magnitude of impact loads.

The limit currently cited of such joint-torque-based control is the risk of delay in the robotic response. In this context, transferring the compliancy directly onto the physical element constituting the robotic system makes sense. For instance, soft actuators provide an intrinsic compliance that enables obtaining a safer mechanism than one can get through the use of a feedback control combined with a stiff actuator. For example, an elastic element placed in between the actuator and the robotic link enables to limit the high frequency impedance of the actuator, while the low-frequency impedance can be managed with a linear feedback system. As an example Rotary actuators with Elastic Chambers have similar working properties to fluid motors, and provide characteristics similar to natural muscles, such as the possibility to vary the stiffness independently of the position or force control [Jordan2009].

Another interesting approach is to partition the torque generation into low and high frequency components with actuators located so that their effect on contact impedance is minimized while their contribution to control bandwidth is maximized. In this sense, low-frequency components (like elastic actuators) are located remotely from the control joint, which is very interesting since such components require much larger

actuators than high-frequency ones do. This system has enabled to reduce the joint inertia of a two-axis prototype by a factor of ten [zinn2004].

5.3 Conclusions

Most existing solutions for actuation management are focusing on interfacing actuators and sensors in (Wireless) Sensor and Actuator Networks. High level requests like "turn on all lights in living room" are not considered and have to be done by the application on top.

Solutions like ZigBee are limited to a certain technology. In the Florence project a flexible structure will provide the possibility to adjust the environment to the needs of the elderly. The ability to use high level requests with an abstraction layer as proposed in the Sensei project will reduce maintenance. Existing high level requests do not need to be adapted if new hardware is installed. The abstraction layer also gives the ability to integrate any actuation hardware which provides an API to control it.

6 Conclusions

This deliverable gave an insight over several issues involved in decision making for complex hardware/software systems. Since decision making is application specific and since the Florence scenarios are not fully defined at this stage, the presentation of the decision making is aimed at clarifying the various dichotomies that will be beneficial when designing the “decision making subsystem” of the Florence project in the subsequent stage.

For context enhancement, the deliverable presented a high-level functional description. Further, “user activity detection”, “interactive learning”, “indoor localization of users” and “event prediction” are identified as the key enablers of Florence context enhancement functionality (based on the current overview of candidate Florence services). Probabilistic methods are considered to be well suitable for the setting of the Florence project.

Since there is no robotic specific context management platform to the best of our knowledge, the best approach for the Florence project is to adapt one state of the art platform that fits the best to the needs of the Florence project. There are quite a few platforms developed by project partners as described in the deliverable. Hereto the requirements of the Florence system should carefully be taken into consideration.

Most existing solutions for actuation management are focusing on interfacing actuators and sensors in (wireless) sensor and actuator networks. High level requests like "turn on all lights in living room" are not considered and have to be done by the application on top. In the Florence project a flexible structure can provide the possibility to adjust the environment to the needs of the elderly. The ability to use high level requests with an abstraction layer as proposed in the Sensei project can also reduce maintenance efforts.

7 Glossary of terms

3APL: Artificial Autonomous Agents Programming Language
AAL: Ambient Assisted Living
ACP: Algebra of Communicating Processes
API: Application Programming Interface
BDI: Belief, Desire and Intentions
BN: Bayesian Network
CALA: Context Access Language
CCS: Calculus of Communicating Systems
CMF: Context Management Framework
CSP: Communicating Sequential Processes
CTW: Context Tree Weighing
DLA: Distributed and Layered Architecture
ECA: Event-Condition-Action
EP: Evolutionary Programming
EKF: Extended Kalman Filter
FIFO: First-In-First-Out
FSM: Finite State Machine
GA: Genetic Algorithm
GP: Genetic Programming
GALS: Globally Asynchronous, Locally Synchronous
GPS: Global Positioning System
HCL: Home Controls-Lighting
HHMM: Hierarchical Hidden Markov Models
HTA: Hierarchical Task Analysis
IR: Infra Red
LAGS: Locally Asynchronous, Globally Synchronous
LBS: Location Based Systems
LWR: Light Weight Robot
NGSI: Next Generation Service Interface
OMA: Open Mobile Alliance
OO: Object Oriented
QoC: Quality of Context
PC: Personal Computer
PPM: Prediction by Partial Matching
PSS: Personal Smart Space
RPC: Remote Procedure Call
UIPS: Ultrasonic Indoor Positioning System
US: Ultra Sound
UWB: Ultra Wide Band
WP3: Work Package 3

8 References

[3APL2010] “3APL (triple-a-p-l) An Abstract Agent Programming Language” website: http://www.cs.uu.nl/3apl/, last accessed on 24/06/2010.

[AALcall1-2010] “Funded Projects for AAL Call 1” website: http://www.aal-europe.eu/calls/funded-projects-call-1, last accessed on 24/06/2010.

[Ackoff1989] Ackoff, R. L., "From Data to Wisdom," Journal of Applies Systems Analysis, vol. 16 pp. 3-9, 1989.

[AeroScout2010] AeroScaout website: http://www.aeroscout.com/content/difference, last accessed on 26/06/2010.

[Alavi1999] Alavi, M. and Leidner, D., "Knowledge management systems: issues, challenges, and benefits," Communication of AIS, vol. 1, no. 14, 1999.

[Albu-Schaffer2009] Albu-Schaffer, A.; Eiberger, O.; Fuchs, M.; Grebenstein, M.; Haddadin, S.; Ott, C.; Stemmer, A.; Wimbock, T.; Wolf, S. & Hirzinger, G., “Anthropomorphic Soft Robotics - from Torque Control to Variable Intrinsic Compliance”, In Proceedings of International Symposium on Robotics Research, Lucerne, Switzerland, 2009.

[Airetrak2010] Airetrak website: http://www.airetrak.com/wifi/wifi_products.php, last accessed on 26/06/2010.

[Avrahami2006] Avrahami-Zilberbrand, D. and Kaminka, G.A., "Hybrid Symbolic-Probabilistic Plan Recognizer: Initial Steps", in AAAI Workshop on Modelling Others from Observations Menlo Park, California, USA, 2006.

[Avrahami2007] Avrahami-Zilberbrand D. and Kaminka, G.A., “Incorporating Observer Biases in Keyhole Plan Recognition (Efficiently!),” in National Conference on Artificial Intelligence, pp. 944-949, 2007.

[Axcess2010] Axcess website: http://www.axcessinc.com/products/tags.html, last accessed on 26/06/2010.

[Bargh2008] Bargh M.S., and de Groote, R., “Indoor Localization Based on Response Rate of Bluetooth Inquiries,” In Proceedings of ACM MELT 2008, San Francisco, USA, September 2008.

[Bauer2009] Bauer, M., Gessler, S., Huang, V., López, F., Meissner, S., Rossi, M., Strohbach, M., Villalonga C., and Petcu, A., “Sensor Information Services”, Sensei Deliverable 2.1, 2009.

[Ben-Gal2007] Ben-Gal I., “Bayesian Networks”, in Ruggeri F., Faltin F. & Kenett R., Encyclopedia of Statistics in Quality & Reliability, Wiley & Sons, 2007.

[Beygelzimer2009] Beygelzimer, A., Dasgupta, S., and Langford, J, “Importance weighted active learning”, International Conference on Machine Learning, 2009.

[Boehm2008] Boehm, S., Koolwaaij, J., Luther, M., Souville, B., Wagner, and M., Wibbels, M., "IYOUIT -- Share, Life, Blog, Play," In Proceedings of the International Semantic Web Conference 2008 (ISWC 2008), October 2008.

[Boehm2008b] Boehm, S., Koolwaaij, J., Luther, M., Souville, B., Wagner, M., and Wibbels, M., "Introducing IYOUIT", In proceedings of the Semantic Web - ISWC 2008, pp. 804-817, LNCS, 2008.

[Bousquet2004] Bousquet, O., von Luxburg, U., and Rätsch, G., "Advanced Lectures on Machine Learning", 1st ed., 2004.

[Braun2009] Braun, A. and Hamisu, P., "Using the Human Body Field as a Medium for Natural Interaction", In Proceedings of the 2nd ACM International Conference on PErvasive Technologies Related to Assistive Environments (PETRA'09), 2009.

[Burbey2008] Burbey I., and Thomas, L.M., "Predicting future locations using prediction-by-partial-match", In Proceedings of the ACM MELT 2008, San Francisco, California, USA, September 2008.

[Carberry2001] Carberry, S., "Techniques for Plan Recognition," in User Modelling and User-adapted Interaction, pp. 31-48, 2001.

[Chai2005] Chai, X., and Yang, Q., "Multiple-Goal Recognition From Low-level Signals", in Proceedings of the AAAI 2005 Pittsburg, PA, USA, 2005, pp. 3-8.

[Chapelle2006] Chapelle, O., Schölkopf, B., and Zien, A., "Semi-supervised learning Cambridge", MA, USA: MIT Press, 2006.

[Cleary1984] Cleary, J.G., and Witten, I.H., "Data compression using adaptive coding and partial string matching", IEEE Transactions on Communications, Vol. 32 (4), pp. 396--402, April 1984.

[Companionable2010] The Companionable project website: http://www.companionable.net/, last accessed on 28/06/2010.

[Dey1999] Dey, A.K. and Abowd , G.D., "Towards a Better Understanding of Context and Context-Awareness", In Proceedings of the 1st international symposium on Handheld and Ubiquitous Computing (HUC'99), 1999, retrieved on 23/06/2010 from http://nl.ijs.si/~damjan/NeOn/99-22.pdf.

[DoW2009] Description of Work document of the Florence project, retrieved on 04/07/2010 from https://doc.novay.nl/dsweb/View/Collection-27371.

[DTree2010] "Classification Trees" website: http://www.statsoft.com/textbook/classification-trees/, last accessed on 24/06/2010.

[Eagle2010] Eagle Vision website: http://www.eaglevision.nl/, last accessed on 26/06/2010.

[Echelon2010] Echelon Corporation website: http://www.echelon.com, last accessed on 24/06/2010.

[Einhorn2009] Einhorn, E., Schroeter, Ch., Gross, H.-M. “Monocular Obstacle Detection for Real-world Environments.” In Proceedings of Autonomous Mobile Systems (AMS 2009), Karlsruhe, Germany, pp.30-40, Springer 2009.

[Ekahau2010] Ekahau website: http://www.ekahau.com/products/real-time-location-system/overview.html, last accessed on 26/06/2010.

[ELV2010] ELV electronic website: http://www.elv.de, last accessed on 25/06/2010.

[Engelbrecht2002] Andries Engelbrecht, “Computational Intelligence: An Introduction”, Wiley & Sons, 2002.

[Fine1998] Fine, S., Singer, Y., and Tishby, N., “The Hierarchical Hidden Markov Model: Analysis and Applications”, Machine Learning, vol. 32, no. 1, pp. 41-62, 1998.

[Firefly2010] Firefly website: http://www.cybernet.com/interactive/firefly/index.html, last accessed on 26/06/2010.

[Forssell2009] Forssell, B., “Indoor Positioning, Indoor Message System Evaluated”, GPS World, April 1, 2009, website: http://www.gpsworld.com/wireless/indoor-positioning/indoor-message-system-evaluated-7168, last accessed on 26/06/2010.

[Geib2006] Geib, C.W., "Plan Recognition", in Adversarial Reasoning, Eds. Kott, A., and McEneaney, W.M., Boca Raton, London, New York: Chapman & Hall/CRC, 2006, pp. 77-100.

[Gu2009] Gu, Y., Lo, A., Niemegeers, I.G., “A Survey of Indoor Positioning Systems for Wireless Personal Networks”, IEEE Communications Surveys and Tutorials 11(1), 2009.

[Harel1987] Harel, D., "Statecharts: A visual formalism for complex systems", Science of Computer Programming, 8(3), pp.231–274, June 1987.

[Haigh2006] Haigh, K.Z., Kiff, L.M., Mayers, J., and Ho, G., "The Independent LifeStyle Assistant (I.L.S.A.): Lessons Learned", Assistive Technology, vol. 18, no. 1, 2006.

[Hoare1985] Hoare, C.A.R., "Communicating Sequential Processes", Prentice Hall, International Series in Computer Science, 1985.

[IPSustem2010] IP-Symcon company website: http://www.ip-symcon.de/, last accessed on 24/06/2010.

[IYOUIT2010] The “IYUOIT project” webpage: http://www.iyouit.eu, last accessed on 22//6/20101.

[IRIS2010] IRIS_LPS website: http://www.tk.informatik.tu-darmstadt.de/de/research/smart-environments/mundo-iris/, last accessed on 26/06/2010.

[Istrate2008] Istrate, D., Binet, M., and Cheng, S. “Real Time Sound Analysis for Medical Remote Monitoring”, In Proceedings of the IEEE Engineering in Medicine and Biology Society Conference (EMBC’08), Vancouver, Canada, 2008.

[Jordan2009] Jordan, M., Pietrusky, D., Mihajlov, M., and Ivlev, O., “Precise Position and Trajectory Control of pneumatic Soft-Actuators for Assistance Robots and Motion Therapy Devices”, In Proceedings of IEEE 11th International Conference on Rehabilitation Robotics, Kyoto, Japan, June, 2009.

[Klir1995] Klir, G.J., Yuan, B., “Fuzzy Sets and Fuzzy Logic: Theory and Applications”, Prentice Hall PTR, 1995.

[Klir1999] Klir, G.J. and Wierman, M.J., “Uncertainty-Based Information”, 2nd edition, Heidelberg: Physica-Verlag, 1999.

[Lamorte2009] Lamorte, L., Venezia, C., “Smart Space a new dimension of context”, Proceeding of the PERSIT Workshop on Intelligent Pervasive Environments, Edimburgh, 2009.

[Lim2006] Lim, H., Kung, L., Hou, J., and Luo, H., “Zero-configuration, robust indoor localization: Theory and experimentation”, In Proceedings of IEEE INFOCOM, 2006.

[Langford2009] Langford, J., and Dasgupta, S., “Active Learning”, tutorial at the 26th International Conference on Machine Learning (ICML 2009), 26 August 2009, available at http://videolectures.net/icml09_dasgupta_langford_actl/, last accessed on 26/05/2010.

[Luca2006] De Luca, D., Mazzenga, F., Monti, C., and Vari, M., “Performance Evaluation of Indoor Localization Techniques Based on RF Power Measurements from Active or Passive Devices”, EURASIP Journal on Applied Signal Processing, 2006.

[Magnet2010] The “MAGNet Beyond project” website: http://cordis.europa.eu/fetch?CALLER=PROJ_ICT&ACTION=D&CAT=PROJ&RCN=80699, last accessed on 17/06/2010.

[Medjahed2009] Medjahed, H., Istrate, D., Boudy, J., Dorizzi, B. “A Fuzzy Logic System for Home Elderly People Monitoring (EMUTEM)”, Fuzzy Systems 2009, 23-25 March 2009, Prague, ISBN 978-960-474-066-6, pp. 69-75.

[Muller2008] Müller, St., Hellbach, S., Schaffernicht, E., Ober, A., Scheidig, A., Gross, H.-M., “Whom to talk to? Estimating user interest from movement trajectories”, In Proceedings of the 17th IEEE Int. Symposium on Robot and Human Interactive Communication, (RO-MAN 08), Munich, Germany, pp. 532-538, IEEE, 2008.

[Northern2010] Northern Digital Inc website: http://www.northerndigital.com/index.html, last accessed on 26/06/2010.

[OMA2010] Open Mobile Alliance, “OMA Next Generation Services Interface V1.0”, http://www.openmobilealliance.org/Technical/release_program/NGSI_v1_0.aspx, last accessed on 10/06/2010.

[OPEN2010] The “OPEN Project” website: http://giove.isti.cnr.it:88/, last accessed on 17/06/2010.

[OSGi2010] “OSGi Alliance” website: http://www.osgi.org, last accessed on 10/06/2010.

[Pearl1988] Pearl, J., “Probabilistic Reasoning in Intelligent Systems”, Morgan Kaufmann Publishers Inc. 1988.

[peddemors2008] Peddemors, A., Eertink, H., and Niemegeers, I., “Density Estimation for Out-of-Range Events on Personal Mobile Devices,” In Proceedings of ACM Mobility Models 2008, May 2008.

[Pérez2002] Pérez, E.J., Poncela, A., Urdiales, C., Bandera, A., and Sandoval, F., "Survey navigation for a mobile robot by using a hierarchical cognitive map", Cognitive Processing, Vol. 3, No. 3-4, pp. 99-106, 2002.

[PERSIST2008] PERSIST Deliverable D2.4, “Initial Architecture Design”, Persist Project, December 2008.

[PERSIST2010] The “PERSIST project” website: http://www.ict-persist.eu/, last accessed on 23/06/2010.

[Persona2010] The Persona project website: http://www.igd.fraunhofer.de/igd-a1/projects/persona/, last accessed on 28/06/2010.

[Polkowski2003] Polkowski, L., "Rough Sets: Mathematical Foundations," Physica-Verlag Heidelberg, Advances in Soft Computing series, 2003.

[Poole1998] Poole, D., Mackworth, A., Goebel, R., “Computational Intelligence: A Logical Approach”, Oxford University Press, 1998.

[Quigley2009] Quigley, M., Gerkey, B., Conley, K., Faust, J., Foote, T., Leibs, J., Berger, E., Wheeler, R., Ng, A., "ROS: an open-source Robot Operating System", ICRA2009 IEEE Int. Conf. On Robotics and Automation, Kobe, Japan, May 12-17, 2009.

[RoboCare2010] the RoboCare project website: http://robocare.istc.cnr.it, last accessed on 24/06/2010.

[Roehning2001] Roehning J., and Riekki, J., "Context-Aware Mobile Robots: Part of Smart Environment", Workshop - Human interaction with mobile robots: an interdisciplinary research challenge, March 30, 2001.

[ROS2010] “Robot Operating System” website: http://www.ros.org, last accessed on 24/06/2010.

[Rougui2009] Rougui, J.E., Istrate, D., Souidene, W. "Audio Sound Event Identification for distress situations and context awareness", Engineering in Medicine and Biology Conference 2009, September 2-6, 2009, Minneapolis, USA, pp. 3501-3504.

[Rubio2010] Rubio, C., “SHARE-it Tutorials”, Master thesis, Universidad Politécnica de Cataluña, 2010, http://upcommons.upc.edu/pfc/handle/2099.1/8987?locale=en, last accessed on 07/07/2010.

[Saerbeck2009] Saerbeck, M. and Holenderski, L., "Configuration Versus Programming in User Interfaces for Autonomous Devices", in Proceedings of the 1st International Conference on Adaptive and Self-adaptive Systems and Applications, Athens/Glyfada, Greece, November 15-20, 2009.

[Shafer1976] Shafer, G., “A Mathematical Theory of Evidence”, Princeton University Press, 1976, ISBN 0-608-02508-9.

[Share-it2010] The “Share-it project” website: http://www.ist-shareit.eu, last accessed on 23/06/2010.

[Sheikh2008] Sheikh, K., Wegdam, M., and van Sinderen, M., “Quality-of-Context and its use for Protecting Privacy in Context Aware Systems”, Journal of Software, Vol 3, No 3 (2008), 83-93, Mar 2008.

[Shi2005] Shishkov, B., Dockhorn Costa, P., and van Kranenburg, H. “Architectural Specification of the AWARENESS Service Infrastructure”, Deliverable D2.10, Freeband AWARENESS project, 2005.

[Simonnet2008a] Simonnet, Th., Couet, A., Ezvan, P., Givernaud, O., Hillereau, P. “Telemedicine platform enhanced visiophony solution to operate a Robot-Companion”. In Proceedings of the International Joint Conferences on Computer, Information and Systems Sciences and Engineering (CISSE08), Bridgeport, USA, 2008.

[Simonnet2008b] Simonnet, Th, and Rocaries, F., “Collaborative tools for a telemedicine PLATFORM: monitoring, communication and storage”, In Proceedings of the International Association for development of the information society (IADIS), Amsterdam, 2008.

[Skype2010] Skype Developer Zone website: https://developer.skype.com/, last accessed on 24/06/2010,

[Snoeck2006] Snoeck, N. & Salden, A., “Contextual Reasoning in Theory and Practice”, AWARENESS project Deliverable Dn1.6, Publication no.: TI/RS/2006/093, 2006.

[Snoeck2007] Snoeck, C.M., van Kranenburg, H., and Eertink, H., “Plan recognition in smart environments”, In Proceedings of the 2nd conference on Digital Information Management (ICDIM 2007), pp. 713-716, 2007.

[Snoonian2003] Snoonian, D., “Smart Buildings”, IEEE Spectrum, pp. 18–23, September 2003.

[Sonitor2010] Sonitor System website: http://www.sonitor.com/, last accessed on 26/06/2010.

[SPICE2010] The “SPICE Project” website: http://www.ist-spice.org/, last accessed on 17/06/2010.

[StarGazer2010] Hagisonic website: http://www.hagisonic.com/, last accessed on 26/06/2010.

[Stikic2008] Stikic, M., Van Laerhoven, K., and Schiele, B. “Exploring Semi-Supervised and Active Learning for Activity Recognition”, In Proceedings of the 12th IEEE International Symposium on Wearable Computers (ISWC’08), pp. 81-88.

[Sukthankar2005] Sukthankar G., and Sycara, K., "A Cost Minimization Approach to Human Behavior Recognition", in Conference on Autonomous Agents and Multi-Agent Systems (AAMAS) 2005, pp. 1067-1074.

[Sutton1998] Sutton, R. S. and Barto, A. G., “Reinforcement Learning: An Introduction”, MIT Press, 1998.

[terHofte2006] Ter Hofte, G.H., “Collecting user experience in context with SocioXensor”, Poster made for demo at CeBIT 2006, Hannover, Germany, March 11-12, 2006, retrieved on 02/07/2010 from https://doc.telin.nl/dsweb/Get/Document-60647/SocioXensor_Poster_CeBIT2006.pdf.

[terHofte2007] Ter Hofte, G.H., “What's that Hot Thing in my Pocket? SocioXensor, a smartphone data collector,” In Proceedings of the e-Social Science 2007, the 3rd International Conference on e-Social Science (ESS 2007), 2007.

[TimeDomain2010] Time Domain website: http://www.timedomain.com/, last accessed on 26/06/2010.

[Topaz2010] Topaz system website: http://www.tadlys.co.il/Pages/Product_content.asp?iGlobalId=2, last accessed on 26/06/2010.

[Ubisense2010] Ubisense website: http://www.ubisense.net/en, last accessed on 26/06/2010.

[Urdiales2008] Urdiales C., Peula J., Barrué C.,Pérez E., “A new collaborative-shared control strategy for continuous elder/robot assisted navigation”, Gerontechnology 2008; 7(2):229

[vanSineren2006] Van Sinderen, M.J., Van Halteren, A.T., Wegdam, M., Meeuwissen, H.B. & Eertink, E.H., “Supporting Context-aware Mobile Applications: an Infrastructure Approach”, In IEEE Communications Magazine, vol. 44, nr. 9, 2006, pp. 96-104

[Wegdam2005] Wegdam, M., “AWARENESS: A project on Context AWARE mobile NEtworks and ServiceS”, In Proceedings of IST/05, 2005. Retrieved on 23/06/20101 from http://www.eurasip.org/Proceedings/Ext/IST05/papers/293.pdf.

[WhereNet2010] WhereNet system website: http://zes.zebra.com/products/rtls/tags-and-call-tags/index.jsp last accessed on 26/06/2010.

[Willems1995] Willems, F.M.J., Shtarkov Y.M., and Tjalkens, Tj.J., "The Context-Tree Weighting Method: Basic Properties," IEEE Trans. Inform. Theory, vol. IT-41, pp. 653-664, May 1995.

[Willems2006] Willems, F.M.J., "Using a Universal Coding Technique for Sources with Large Alphabets to Estimate Differential Entropy," Allerton 2006, retrieved on 31/03/2009 from http://www.sps.ele.tue.nl/members/F.M.J.Willems/RESEARCH_files/CTW/decomptreeweighting.pdf.

[Wu2003] Wu, H., Siegel, M., and Ablay, S., "Sensor Fusion Using Dempster-Shafer Theory II: Static Weighting and Kalman Filter-like Dynamic Weighting", in Proceedings of IMTC 2003 - Instrumentation and Measurement Technology Conference, Vail, CO, USA, 20-22 May 2003

[Xensor2010] The "Xensor project" website: http://www.telin.nl/index.cfm?language=en&context=1107&id=1108, last accessed on 23/06/2010.

[Zebra2010] Zebra enterprise solutions website: http://zes.zebra.com/products/rtls/tags-and-call-tags/index.jsp, last accessed on 26/06/2010.

[ZigBee2010] ZigBee Alliance website: http://www.zigbee.org, last accessed on 24/06/2010.

[Zimmer2006] Zimmer T, "QoC: Quality of Context – Improving the Performance of Context-Aware Applications", Advances in Pervasive Computing, Adjunct Proceedings of Pervasive 2006.

[zinn2004] Zinn, M., Khatib, O., Roth, B. and Salisbury, J. "A New Actuation Concept for Human-Friendly Robot Design: Playing it safe", Robotics and Automation Magazine, volume 11, pages 12-21, June 2004.

[Zunith 2010] Zunith website: http://www.zonith.com/products/zonith-indoor-positioning-module/, last accessed on 26/06/2010.