www.iks-project.eu

AmI Case - Design and Implementation (D4.1)

Deliverable: D4.1 - AmI Case Design and Implementation
Delivery Date: 30.09.2011
Author(s): Sabine Janzen, Eva Blomqvist, Andreas Filler, Suat Gönül, Tobias Kowatsch, Alessandro Adamou, Sebastian Germesin, Massimo Romanelli, Valentina Presutti, Cihan Cimen, Wolfgang Maass, Senan Postaci, Erdem Alpay, Tuncay Namli, Gokce Banu Laleci Erturkmen
Filename: iks_d41_AmIcaseDesImpl.docx

Interactive Knowledge Stack for Semantic Content Management Systems

IKS is co-funded by the European Union and develops technology for intelligent content management

Table of contents

Document History

Ver-sion	Name	Date	Remark
V0.1	Sabine Janzen	11.04.11	Template
V0.2	Sabine Janzen	03.05.11	Integration of preliminary reports and existing parts
V0.3	Suat Gönül	05.05.11	Section 4
V0.4	Sabine Janzen	20.05.11	Integrating parts of section 5 & 6
V0.5	Sabine Janzen	27.06.11	Splitting of Part A & B for first QA cycle
V0.6	Sabine Janzen	28.06.11	Deleting former section 3, Integration of Introduction (Section 2) Revision of (new) Section 3
	Suat Gönül	28.06.11	Revised Section 4 (Removed footnotes, added samples from removed documents)
	Sabine Janzen	28.06.11	Revision of introduction in Section 5, Section 5.2 (now 5.1)
	Eva Blomqvist	28.06.11	Section 5.1 (now 5.2), 6.1 (now 6.2)
V0.7	Sabine Janzen	29.06.11	Section 6.1
	Sabine Janzen	30.06.11	Section 6.3
V0.8	Sabine Janzen	30.06.11	References
	Eva Blomqvist	30.06.11	Revision of Section 4, 5.2, 6.2
	Sabine Janzen	01.07.11	Revision of Section 1, 4, 5, 7
V0.9	Sabine Janzen	01.07.11	Document ready for task-internal QA
V1.0	Gokce Banu Laleci Erturkmen	04.07.11	Review of document
	Tobias Kowatsch	04.07.11	Review of document
	Massimo Romanelli	05.07.11	Review of document
	Sabine Janzen	05.07.11	Part A ready for QA
V1.1	Sabine Janzen	05.07.11	Extension of Part A with Part B sections to enable working on D4.1 during first QA cycle
V1.2	Suat Gönül	13.07.11	Initialized Section 8 – Opened slots for each module to be filled by responsible partners
V1.3	Tobias Kowatsch	18.07.11	Preliminary version of Section 10 – Evaluation of AmI Use Case
	Massimo Romanelli	05.07.11 (integrated 26.07.11)	Integration of preliminary report (Task 4.1 – Phase 10 – Selection and Ordering of Required Devices (Step 10.1))

V1.4	Andreas Filler	08.07.11 (integrated 26.07.11)	Section 9
V1.5	Andreas Filler	26.07.11	Section 8.3.2, 8.3.3, 8.4
V1.6	Sabine Janzen	08.08.11	Revision of Section 8, 9 and 10; Revision of Section 2, 4 and 5 according to QA review by Alex
V1.7	Suat Gönül	09.08.11	Input for Content Aggregator and Content Filterer in Section 8.5
V1.8	Tobias Kowatsch	09.08.11	Revision of Section 10 (minor wording changes and placement of qualitative feedback of the AmI experiment to the appendix); Appendix B
V1.8.1	Alessandro Adamou	09.08.11	Added SCEM reengineer and refactorer components.
V1.8.2	Massimo Romanelli	10.08.11	Integration of subsections in Section 8 and 9
V1.9	Andreas Filler	10.08.11	Integration of Progress Reports 6 to 8 as Section 7
V1.10	Andreas Filler	11.08.11	Adjustment of Table 5 in Section 9, Integration of Section 8.6
V1.11	Sabine Janzen	11.08.11	Revision of Section 7, 8, 9; Revision of Section 2 and 5 according to QA review by Gregor (on going)
V1.12	Sabine Janzen	15.08.11	Revision according to QA review by Gregor: • Transfer of software requirements part of Section 4 in Appendix A; Integration of ontological requirements part into section 5.2 → deleting section 4 • Revision of section 1, 2, 3 • Revision of Fig. 5 & 6, Section 4, Section 5.3 • Numbering of figures
V1.13	Suat Gönül	17.08.11	Generation of Section 10 and revision of Section 7 according to Gregor’s review
V1.14	Sabine Janzen, Tobias Kowatsch	18.08.11	Small Revisions in section 7 and 10; Integration of discussion in section 11; Integration of future work in section 12
V1.14.3	Massimo Romanelli, Alessandro Adamou	20.08.11	Relations to IKS objectives in sections 3.2 and 3.3
V1.15	Andreas Filler	21.08.11	Revision of Section 6 & 7 (Integration of new logical architecture and adaption of both Sections to the current state of the development)
V1.16	Sabine Janzen	22.08.11	Revision of section 6 & 7; integration of comments and questions to authors

V1.17	Suat Gönül	24.08.11	Revision of Section 6.5 and 6.6.6 based on the feedbacks of Andreas and Sabine
V1.18	Andreas Filler	24.08.11	Revision of Section 10, 10.2; Added Section 10.1
V1.21	Suat Gönül	25.08.11	Updates on Section 10.2, added section 10.3
V1.22	Andreas Filler	25.08.11	Revised Section 6; integrated VIE^2 part by Massimo in Section 10.3.7 (sent by mail)
V1.23	Massimo Romanelli	25.08.11	Revised Section 6
V1.24	Andreas Filler	26.08.11	Revision of Section 7 regarding comments by Sabine Janzen
V2.0	Sabine Janzen	26.08.11	Update of table in Section 10; Small revisions here and there before internal QA
V2.1	Sabine Janzen	31.08.11	Revision of Section 1-5, 11 and 12 according to QA by Tobias; integration of missing discussion part in Section 11
V2.2	Andreas Filler	01.09.11	Revision of Section 6-8 and 10 according to QA by Tobias and Massimo
	Tobias Kowatsch	01.09.11	Review of document according to QA by Massimo
V2.3	Andreas Filler	01.09.11	Document ready for QA
V3.0	Sabine Janzen	20.09.11	Finalization of D4.1 after QA and preparation for publication

Document Information

Item	**Value**
Identifier	D4.1
Author(s):	Sabine Janzen, Eva Blomqvist, Andreas Filler, Suat Gönül, Tobias Kowatsch, Alessandro Adamou, Sebastian Germesin, Massimo Romanelli, Valentina Presutti, Cihan Cimen, Wolfgang Maass, Senan Postaci, Erdem Alpay, Tuncay Namli, Gokce Banu Laleci Erturkmen
Document title:	IKS Deliverable – D4.1 Report: **AmI Case Design and Implementation**
Source Filename:	iks_d41_AmIcaseDesImpl.docx
Actual Distribution level	Public
Document context information	
Project (Title/Number)	Interactive Knowledge FP7 231527
Work package / Task	WP4/ T4.1
Responsible person and project partner:	Sabine Janzen, Furtwangen University
Quality Assurance / Review	
Name / QA / Release / Comment	Gregor Engels, Alex Conconi
Citation information	

Official citation	Sabine Janzen, Eva Blomqvist, Andreas Filler, Suat Gönül, Tobias Kowatsch, Alessandro Adamou, Sebastian Germesin, Massimo Romanelli, Valentina Presutti, Cihan Cimen, Wolfgang Maass, Senan Postaci, Erdem Alpay, Tuncay Namli, Gokce Banu Laleci Erturkmen; IKS Deliverable – D4.1 Report: AmI Case Design and Implementation (Public), 2011.

IKS in a Nutshell

“Interactive Knowledge Stack” (IKS) is an integrating project targeting small to medium Content Management Systems (CMS) providers in Europe providing technology platforms for content and knowledge management to thousands of end user organizations. Current CMS technology platforms lack the capability for semantic web enabled, intelligent content, and therefore lack the capacity for users to interact with the content at the user’s knowledge level. The objective of IKS therefore, is to bring semantic capabilities to current CMS frameworks. IKS puts forward the “Semantic CMS Technology Stack” which merges the advances in semantic web infrastructure and services with CMS industry needs of coherent architectures that fit into existing technology landscapes. IKS will provide the specifications and at least one Open Source Reference Implementation of the full IKS Stack. To validate the IKS Stack prototype solutions for industrial use cases ranging from ambient intelligence infotainment, project management and controlling to an online holiday booking system will be developed.

1 Executive Summary

The intelligent bathroom use case represents a "far out" vision of IKS for direct user interactions with embedded contents that are organized by a "Semantic CMS Technology Stack". This case combines advanced content and knowledge management with an ubiquitous computing scenario in a place everybody is familiar with - the bathroom. The use case shows how users can interact with contents in physical environments in a way that leaves the dimension of "small windows to the info sphere" as known by the "monitor paradigm" (Janzen et al., 2010).

The objective of D4.1 is to provide guidance for the design, implementation and evaluation of such interactive knowledge-supported ubiquitous environments. In D2.1 (Janzen et al., 2010), a *Situational Design Methodology for Information Systems (SiDIS, formerly known as CoDesA)* (Maass & Janzen, 2011) was introduced and results from first four tasks of that methodology were presented. In this deliverable, tasks 5 to 8 are applied. Therefore, key research challenges for the AmI case will be identified and supported by related work. One of these challenges is the development of a knowledge representation for highly dynamic environments. So, we started with the conceptual design of such a knowledge representation for the AmI use case; the approach consisting of a contextual and situational part will be introduced. We will present our procedure in designing and modelling the knowledge representation by means of Ontology Design Patterns (Gangemi, 2005; Gangemi & Presutti, 2009) that are provided by IKS and that are combined with conceptual models based on SiDIS. Furthermore, the development of a logical as well as technical architecture for the AmI case is elaborated. Afterwards, we will focus on the implementation of the technical architecture of the system, especially on the presentation of contents in the bathroom based on the semantic knowledge representation. Next, the results of the empirical evaluation of the resulting software integrated in the physical bathroom environment will be presented. Furthermore, the results of the re-iteration phase in T4.1 will be shown, validating the Beta version of IKS in the context of the real-time bathroom environment. D4.1 will be closed by a discussion of experiences and lessons learned when developing interactive knowledge-supported ubiquitous information systems by means of IKS.

2 Introduction

In Task 4.1 (T4.1) as well as in the previous Task 2.1 (T2.1), we applied the design methodology *SiDIS* (cf. Fig. 1) to design an interactive knowledge-supported ubiquitous environment. With completion of T2.1, we were able to present results for SiDIS task 1 to 4 (cf. D2.1). In T4.1, SiDIS tasks 5 to 8 are applied (cf. Fig. 1).

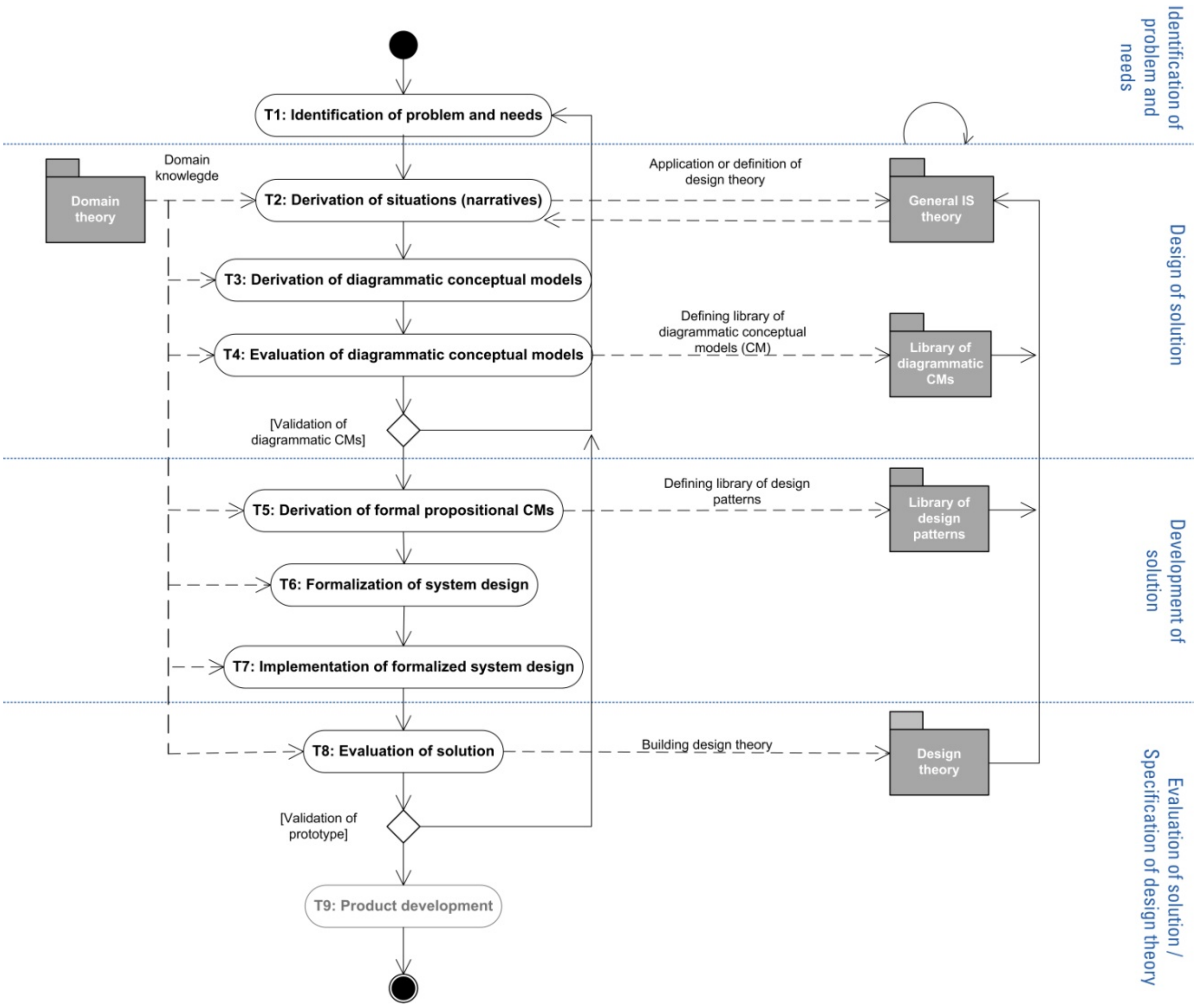

T1: Identification of problem and needs
Domain knowlegde
Domain theory
T2: Derivation of situations (narratives)
Application or definition of design theory
General IS theory
T3: Derivation of diagrammatic conceptual models
T4: Evaluation of diagrammatic conceptual models
Defining library of diagrammatic conceptual models (CM)
Library of diagrammatic CMs
[Validation of diagrammatic CMs]
T5: Derivation of formal propositional CMs
Defining library of design patterns
Library of design patterns
T6: Formalization of system design
T7: Implementation of formalized system design
T8: Evaluation of solution
Building design theory
Design theory
[Validation of prototype]
T9: Product development
Identification of problem and needs
Design of solution
Development of solution
Evaluation of solution / Specification of design theory

Fig. 1: SiDIS design methodology (Maass & Janzen, 2011)

With the finalization of T2.1, we had identified relevant usage situations of the interactive knowledge-supported ubiquitous environment with the help of empirical user studies (D2.1 - Janzen et al., 2010) (cf. SiDIS task 2). The situations were presented in form of narratives. Narrative thinking is context-sensitive, related to situations and articulated in temporal sequences, around intentions and actions of groups and individuals (Zukier, 1986). So, we assume that narratives are effective means for building and understanding situations of content-centred UIS because of their ability to provide discourse information and sequences of interactions between actors (Hinchman & Hinchman, 1997; Kuechler & Vaishnavi, 2008). First, we had a pool of 12 usage situations. They were evaluated within an empirical user study (n=46) concerning their relevance to the subjects. As a result, we were able to identify the following three best-ranked situations of the study in T2.1 that represent the starting point for our work in T4.1:

Situation 1. It's Thursday morning. I get site-specific weather information when I am brushing my teeth in the bathroom. Based on weather information and my calendar, free-time event suggestions are given (e.g. "Today, 8 p.m. - Sneak Preview at CinemaOne). Do you want to order tickets?

Situation 6. Happily, it's Saturday morning - weekend – I have taken a shower listening to my favourite music. Leaving the bathroom, my partner flits into the room. My music has become silent and my partner is welcomed by music from her/his own music collection. The music starts at the point in the playlist where my partner stopped listening the evening before. After a while, my partner says "Stop music" and the song fades out. My partner wants to see his/her personal news collage while taking a shower.

Situation 11. It's Monday morning. I am in the bathroom, brushing my teeth and listening to the news on the radio. Then, I take a shower. Now, the news messages are displayed in form of pictures and text at the glass door of the shower.

The objective of T4.1 is the validation of IKS in a real-time usage scenario by means of these identified narratives. Next, we will identify research issues of the AmI use case and illuminate appropriate related work (cf. Section 3). One of these research issues is the development of a knowledge representation for highly dynamic environments as represented by the AmI case. So, we start with the introduction of the conceptual design of such a knowledge representation based on IKS. In Section 4 and 5, we present the procedure of designing and developing the knowledge representation by means of Ontology Design Patterns that are provided by IKS and that are combined with conceptual models based on SiDIS. Furthermore, the development of a logical architecture for the AmI case is elaborated (cf. Section 6). Afterwards, the implementation of the architecture of the system and its integration into the bathroom, especially the presentation of contents in the bathroom based on the semantic knowledge representation is described (cf. Section 7 and 8). Next, the results of the empirical evaluation of the resulting software integrated in the physical bathroom environment will be presented in Section 9. Last, the results of the re-iteration phase in T4.1 will be shown, validating the Beta version of IKS in the context of the real-time bathroom environment (cf. Section 10) followed by a discussion of experiences and lessons learned when developing interactive knowledge-supported ubiquitous information systems by means of IKS and future work (cf. Section 11 and 12).

3 Research Issues of the AmI Use Case

Ambient Intelligence (AmI) has been characterized in many different ways, for instance "'Ambient Intelligence' implies intelligence that is all around us" (Maeda & Minami, 2006) or "A vision of future daily life […] contains the assumption that intelligent technology should disappear into our environment to bring humans an easy and entertaining life" (Crutzen, 2006). The diverse definitions assembled by Cook et al. (2009) highlight features that are expected from AmI technologies: context-aware, personalized, anticipatory, adaptive, transparent (i.e. disappearing), ubiquitous, and intelligent (Cook et al., 2009; Aarts, 2004). The vision of Ambient Intelligence focuses similarly on the human needs as on the technology development. This corresponds with the assumption of economists, that the fourth major wave following the classical economies will be the experience economy. The general believe is that people are willing to spend money on having experiences in everyday life (de Ruyter & Aarts, 2004).

There are diverse settings in which AmI applications can impact human's life, for instance education, smart homes, health monitoring and assistance, workplaces etc. Cook et al. (2009) present an overview of research projects addressing the different settings. In T4.1, we develop an intelligent bathroom as part of a smart home environment. Diverse artefacts and items in such a smart home can be enriched with sensors to gather information and even act independently with or without human intervention (Cook et al., 2009). Friedewald et al. (2005) specified four basic functions that are covered by AmI solutions in these home environments: (1) home automation; (2) communication and socialization; (3) rest, refreshing, entertainment and sport; and (4) working and learning.
For the development of the intelligent bathroom, we will focus on functions (2) and (3). The bathroom is already characterized by well-defined activities. Based on usage analyses, Lashina (2004) identifies three main application areas especially in the bathroom: infotainment, healthcare and beauty care. In general, changes in the use of private bathroom spaces have been observed. The solely functional space for personal hygiene transforms to a centre of care and comfort. This tendency is confirmed by the increasing number of private Jacuzzis, allocation of rising amount of space to the bathroom, more differentiated design of bathrooms and the trend in using electronics in the bathroom (Lashina, 2004).
AmI technologies should enable interactions between user and environment via "hands-free" interaction that means natural feeling human interfaces. Furthermore, they should allow ubiquitous wireless access to information, communication, services and entertainment (Aarts & Roovers, 2003; Ducatel et al., 2001). "The goal of ambient intelligence is not only to provide such active and intelligent technologies, but to weave them seamlessly into the fabric of everyday lives and settings and to tailor them to each individual's specific needs" (Cook et al., 2009). Fig. 2 shows the relationship between AmI and contributing technologies because AmI has a relationship to many areas of computer science. Cook et al. (2009) organize these contributing technologies into five areas: sense, reason, act, human computer interaction (HCI) and secure. We will adopt this organization to structure challenges of AmI systems in general as well as our research issues for T4.1 that will be elaborated in detail in the following sections.

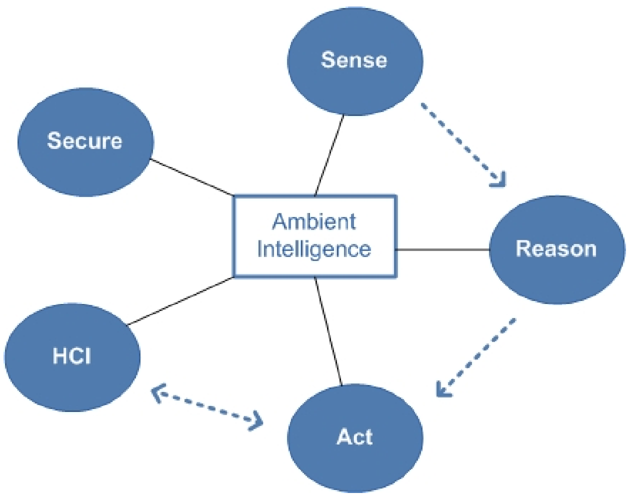


Fig. 2: Relationship between AmI and contributing technologies (based on Cook et al., 2009)

Sense – Ambient Intelligence is designed for real-world physical environments. Therefore, effective use of sensors, i.e. physical devices is necessary to perceive the environment. A discriminating factor for the distribution of AmI functionality in a network of devices is the availability of energy in device. This constrains the amount of information processing for a given device (Aarts & Roovers, 2003). In T4.1, we will not focus on the power aspect for devices in intelligent environments, but on the *management of networks of devices* (cf. Section 3.4)

Reason – The data gathered by perceiving the environment is used to reason about the environment. A number of types of reasoning must take place, for instance user modelling, activity prediction and recognition, decision-making or spatial-temporal reasoning (Cook et al., 2009). Based on a literature review, we identified several challenges concerning these aspects. Various inputs have to be analyzed to interpret the environment's state, e.g., sensor data or written or spoken language by the user (cf. Section 3.3). Furthermore, information and knowledge associated with the environment have to be represented. "Early experience in intelligent systems development shows us that intelligence isn't possible without knowledge; this is also true for AmI" (Ramos et al., 2008). This also covers the modelling, simulation and *representation of entities in the environment*, e.g. devices or persons (cf. Section 3.2). Besides the representation of both state of and knowledge about the environment, decisions or actions have to be planned for achieving a particular goal (cf. Section 3.1). To improve planning algorithms, the system has to learn about the environment; in the context of AmI, machine learning should base on observation of users and their reactions (cf. Section 3.4) (Ramos et al., 2008). On an architectural level, Sun et al. (2009) and Garáte et al. (2005) identified the *challenge of the dynamics* of service availability and service mapping in AmI environments. The availability of services is continuously changing. Furthermore, services have to be mapped automatically to specific situations, users, information or contexts (cf. Section 3.1).

Act – After interpreting, learning and planning, AmI systems are required to act in order to change the state of the environment (Ramos et al., 2008). This also includes the management of the network of devices. We adopt this challenge in sections 3.1 and 3.4.

HCI - Human computer interaction in AmI environments requires *innovative, context-aware and natural user interfaces* instead of traditional interfaces (Ramos et al., 2008) (cf. Section 3.3). "AmI should be made easy to live with" (Cook et al., 2009). This challenge covers the answers to the questions: What do users want? How do people live their life? How do they use spaces? What about their willingness for interacting with AmI environments? (Sun et al., 2009; Shadbolt, 2003; Aarts, 2004). Lashina (2004) derives interaction requirements that are specific for the bathroom context, e.g., users often have their hands wet in the bathroom, therefore interaction via touch screens could be fault-prone; noise sources like running water in the bathroom creates difficult acoustic conditions for using speech control.

Secure – The aspect of assuring privacy protection (see for example Courtney (2008) or Dinev and Hart (2006)) will not be a focus of T4.1.

In T4.1, we have explored how to bring semantic content management into an intelligent environment of the aforementioned characteristic by means of an Ubiquitous Information System (UIS) (Maass & Janzen, 2011). In the following, the aforementioned research challenges will be elaborated according to their state of the art.

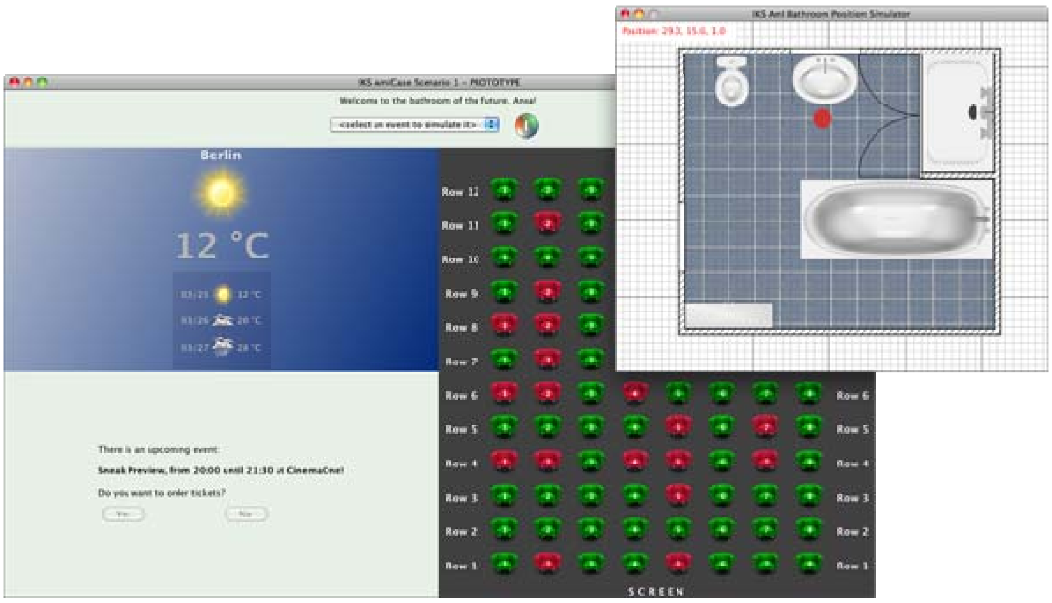
3.1 Architectural Issues regarding Ubiquitous Information Systems

After finalization of T2.1 in February 2010, we started to implement a first prototype of the best-ranked situation 1 (cf. D2.1) during the first developer's camp in Furtwangen. Furthermore, the camp should link T2.1 with T4.1 and offer space for a get-together of developers in T4.1. These developers identified skill sets and exchanged knowledge concerning interfaces and their existing tools, etc. In addition, that developer's camp was the kick-off of a technical

feasibility study. Afterwards, the simulator prototype of situation 1 was further developed and the team discussed lessons learned from the developer’s camp, e.g., the need for standardized event and dialogue-based events. In a second developer’s camp in Saarbrücken, the simulator prototype was extended according to ambient aspects (e.g., the movement of a user in the bathroom). Furthermore, the team derived guidelines for AmI architecture design. Both AmI case developers camps generated new input for a discussion of the architecture for the UIS. Furthermore, the meetings helped to better understand the tools and workflows used by other project partners.
The major result of the two developer's camps was an implementation of a feasibility study of situation 1 (see Fig. 3):

> *Anna gets site-specific weather information when she is brushing her teeth in the bathroom. Based on weather information and her calendar, free-time event suggestions are given, e.g. "Today, 8 p.m. - Miss Marple Night at CinemaOne. Do you want to order tickets?*

This prototypical implementation allowed us to simulate a person entering a bathroom according to that situation. Hereby, a semantic knowledge base was used for an event-based and dialogue-based interaction in the simulated bathroom. All contents were retrieved from external data sources and directly integrated into the semantic knowledge base.
Within the literature review considering architectural issues when developing such UIS, we detected that the pure technical part of UIS consists of various devices, modules and services. Such a system requires the integration of external services to feed the environment with knowledge. Furthermore, UIS need extensible architectures to enable the seamless integration of new modules and services at run-time. Service Oriented Architecture (SOA) has become a “de facto architecture” to integrate disparate systems in a loosely coupled way. However, such architecture should be enhanced with service discovery and orchestration to implement desired business tasks from a set of loosely coupled functional units.
Within IKS, a reference architecture for semantic content management systems has been developed, which also follows this approach. For this reason the logical architecture development within our task is based on the results provided by Christ and Nagel (2011).


IKS amiCase Scenario 1 - PROTOTYPE
<<select an event to simulate it>>
Berlin
12 °C
03/25 12 °C
03/26 20 °C
03/27 28 °C
Yes
No
Row 12
Row 11
Row 10
Row 9
Row 8
Row 7
Row 6
Row 5
Row 4
Row 3
Row 2
Row 1
SCREEN
IKS AmI Bathroom Position Simulator
Position: 29.3, 15.6, 1.0

Welcome to the bathroom of the future, Anna!

There is an upcoming event:
Sneak Preview, from 20:00 until 21:30 at CinemaOne!
Do you want to order tickets?

Fig. 3: Screenshots of prototypical implementation of situation 1

SOA is widely implemented through Web Services. Web services are described through Web Services Description Language (WSDL), which is an XML-based language. WSDL describes four critical pieces of data:

- Interface information describing all publicly available functions
- Data type information for all message requests and message responses
- Binding information about the transport protocol to be used
- Address information for locating the specified service

However WSDL describes the web services at the syntactic level and lacks the semantic expressivity. Semantics can enable reuse, discovery and orchestration of Web services. W3C has an important initiative in this context, namely the OWL-S Service Specification (previously DAML-S). OWL-S is an OWL-based Web service ontology, which supplies Web service providers with a core set of mark-up language, constructs for describing the properties and capabilities of their Web services in unambiguous, computer-interpretable form.
REST (Representational State Transfer) which has become popular and achieved significant success in web development, is one style of developing web services. The current version of the WSDL (WSDL 2.0) provides support for describing RESTful web services. There are also efforts (Sheth, 2007; Kopecky, 2008) that target the semantic annotation of RESTful web services.
Based on experiences in T2.1 and T4.1 and the results of the AmI case developer's camps, we derived several guidelines for UIS architectures. There are *UIS guidelines* independent of a specific implementation that focus on AmI specific aspects. Furthermore, *implementation guidelines* address computer science perspectives.

UIS guidelines

- Architecture development process should be a mix of both a top-down and a bottom-up approach to combine general AmI case ideas with learnings from early prototypes.
- Meta level descriptions of situations should be mapped automatically at runtime.
- Beside the description of modules also services inside the AmI Architecture as well as their data flow should be described.
- Architecture should be able to integrate contents from different external data sources (e.g., IKS capable systems, linked data sources, regular databases, non-semantic sources).
- Architecture should offer semantic lifting capabilities for different forms of content.

Implementation guidelines

- Light-weight protocols should be used to enable the simple creation of new modules and services as well as their integration at runtime.
- Similar to the whiteboard approach a semantic data bus should be used to orchestrate the communication between several AmI modules.
- The modules should realize the concept of encapsulation by clearly defined service interfaces in order to ensure high flexibility and fewer dependencies.
- The integration of modules at runtime should be managed by semantic service orchestration to allow the creation of highly flexible AmI environments.
- The implementation of the resulting architecture should be based on open standards.

These architectural issues regarding IKS-supported UIS will be applied within the design and the implementation of the AmI use case (cf. Section 6 and 7).

3.2 Knowledge Representations for Intelligent Spaces

As already mentioned above, a number of types of reasoning have to take place in order to provide semantically enhanced content interactions in an AmI environment. In this section we present a brief discussion of interesting research challenges of knowledge representation, and state of the art of knowledge representation for AmI related to those issues. Classically, knowledge representation for ambient intelligence has been mostly about storage of data, e.g. sensor data, and representing the physical context of the actors involved. Many, models of context have been proposed, and more recently several of these have also been expressed as ontologies (Wang et al., 2004; Kim & Choi, 2006; Ou et al., 2006). Using a combination of rules and ontological definitions, reasoning tasks are performed on these ontologies, e.g. both checking consistency of states, as well as deriving abstract implicit 'situations' from a concrete context. The latter is also one of the main intentions behind the DOLCE Descriptions and Situation (Gangemi & Mika, 2003) ontology design pattern, where concrete situations can be modelled to satisfy abstract partial descriptions of things like plans, contexts, abstract situations etc.
However, most context ontologies that exist today are still restricted to physical context only, possibly in combination with some user attributes and preferences. Interesting extensions to such approaches would take into account also formal models of the social context and information context when reasoning over content and user interactions. The intuition is that the social context can change the meaning and relevance of different content, i.e. the content is interpreted differently in different social situations or environments. The information context can also impact the interpretation and relevance of content, e.g. content can be more or less relevant depending on the previous knowledge of the user or the previous content the user has consumed or produced.
Content representation has a long history in fields such as e-Learning and CMS. Current content representations in commercial CMS are usually based on some notion of content types, e.g. notion such as pictures or text or more detailed types such as published articles and unpublished drafts, and topic, i.e. what the content is about. Behrendt et al. (2005) modelled so called Knowledge Content Objects (KCOs) from different perspectives, e.g. description of the content itself, description of the organization, description of business model and legal aspects, presentation information, and trust and security aspects. Although this model can act as an inspiration, in the IKS AmI use case the focus is not primarily on the business or trust aspects of content, but rather on creating a pluggable content infrastructure. Hence, in this case a challenge would be to create a general content model, where new content sources and content types can be plugged without effort, even at runtime of the system. The technical realization of such a pluggable architecture may take inspiration from Semantic Web services (Martin et al., 2007) and their application according to the REST approach (Filho & Ferreira, 2009).
Another challenge is the construction of the ontologies, e.g. how can methodologies for AmI system development and ontology engineering be combined and possibly even unified? There are numerous ontology engineering methodologies available, and more recent ones (such as the NeOn methodology, cf. Suarez-Figueroa & Gomez-Perez 2009) usually tries to unify different aspects and tasks, to create a more flexible methodology. A recent approach is also to exploit the notion of ontology design patterns (ODPs) (Gangemi, 2005; Gangemi & Presutti, 2009) in ontology engineering, where a methodology for rapid prototyping of ontologies has emerged (Presutti et al., 2009), called eXtreme Design (XD). While AmI specific methodologies, such as SiDIS (Maass & Janzen, 2011), focus more on the behaviour of the system and its interactions with the users, XD and other ontology engineering methodologies focus on the static aspects of knowledge, e.g. what needs to be represented and in what logical form. By using ODPs the ontologies built will be modular and flexible; however, a challenge would be to extend ODPs by additionally mapping them to some system behaviour, rather than only to a static knowledge representation and some reasoning services. An

additional challenge is to study and combine approaches such as SiDIS (Maass & Janzen, 2011) and XD (Presutti et al., 2009), resulting in an improvement of both methodologies and a possible unification.
In the end, every knowledge representation needs to rely on a formal language for its semantics and its syntactical representation. In IKS it is agreed that open standards is an important cornerstone, hence, also the knowledge representations should use open standards. The main formalism for representing data on the Semantic Web is the Resource Description Framework[1] (RDF). It is a World Wide Web Consortium (W3C) specification for representing information on the Web. The RDF data model is based on triples; where subject and predicate are resources identified by Uniform Resource Identifiers (URIs), while the object can either identify a resource or a literal value. The RDF Schema[2] (RDFS) provides a base mechanism for making terminologies shareable, by providing specifications and additional constructs for expressing classes and, for instance, subsumption relationships. The Web Ontology Language[3] (OWL) provides the framework for extending RDFS models to full-fledged ontologies. The second version of the OWL language, OWL 2, which became a W3C recommendation in late 2009, adds some interesting new features that could potentially be very valuable for reasoning on contexts and situations, e.g. the possibility to define property chains. To investigate the new possibilities of OWL 2 with respect to reasoning over content and context is a challenge on the language level. On top of ontologies we can define rules, where a rule is an implication axiom with an antecedent and a consequent for the definition of conditional facts. The Semantic Web Rule Language[4] (SWRL) is one rule language, which is supported by a large variety of software implementations of OWL. These issues concerning knowledge representations in intelligent spaces have influenced the design and the development of the knowledge representation of the AmI use case (cf. Section 4 and 5).
It can be observed that on the mere knowledge representation side, the AmI Case vision incorporates a significant portion of general IKS requirements and shares most of its goals in terms of ontology design, inference and rule execution. A highly apparent implication of this shared vision is the possibility to reuse a noteworthy amount of ontology modules designed and developed for the AmI Case. Several of these are indeed being included as part of the default IKS ontology network.
In more general terms, the knowledge representation aspect in the AmI Case is virtually unhindered by the transition from the traditional desktop paradigm to that of the intelligent multimodal environment. It is here shown how fundamental issues addressed in this use case relate to the objectives of the IKS as per the project's Description of Work:

AmI Focus	**IKS Objectives**[5]
Modelling changes and transformations of items in the AmI interaction context.	"*Describe the interdependencies between […] the static representation of artefacts (objects) and the representation of changes in these artefacts.*" (p. 39)
Pattern-based ontology modules that model the relationships between users and content or knowledge items.	"*Investigation of new metaphors for user interaction with knowledge based on ontology design patterns*". (p. 39)
Modular knowledge representation approach whose elements can be easily aligned with controlled vocabularies or uncommon representation schemas.	"*Transformation patterns […] and easy-to-use modules for semantic interoperability*" across representation languages. (p. 39)
Harmonizing external context-related data in	Reuse and integration of "off-the-shelf" rea-

[1] http://www.w3.org/RDF/
[2] http://www.w3.org/TR/rdf-schema/
[3] http://www.w3.org/TR/2009/PR-owl2-overview-20090922/
[4] http://www.w3.org/Submission/SWRL/
[5] For more details on the IKS objectives see „Interactive Knowledge Stack for small to medium CMS/KMS providers: Description of Work".

accordance with the AmI-internal knowledge representation model.	soning technologies and existing rule languages. (pp. 39)

3.3 Discourse Management in Intelligent Spaces

As mentioned earlier, the targeted environment for future bathrooms should have ambient intelligence to bring humans an easy an entertaining life (Crutzen, 2006). This should be accomplished with innovative, context-aware and natural interfaces (Ramos et al., 2008). For humans, natural interaction is meant to be multi-modal (e.g., using speech and gestures) and hence, our intention should be to integrate multi-modal input and output channels in the ambient environment, wherever appropriate, to give the user the feeling of an interactive habitat that feels natural to her. Sonntag (2010) describes, that a realization of such intelligent user interfaces needs explicit models of the discourse of the interaction, the available information material, the domain of interest, the task and/or models of a user, at best in ontological form. These models are then used in a dialogue system which is defined as a computer system that interacts on a turn-by-turn basis and in which spoken natural language interface plays an important part in the communication (Fraser, 1997). Recently, these systems have been extended to multimodal dialogue systems that process two or more combined user input modes in a coordinated manner with multimedia system output (Oviatt, 2002). Bui (2006) enumerates the usual components of a dialogue system as:

Input
Multimodal input of a dialogue system is usually a subset of the various modalities, e.g., speech, pointing gestures, gaze, facial expressions, pressing a button, and so on. To be precise, input can furthermore classified in active and passive input where active input describes the user's intended input to express certain commands and passive input is defined as user input that is unobtrusively and passively monitored without requiring any explicit command to a computer (Oviatt, 2002).

Fusion
The Input component only collects the various modalities and forwards these to the Fusion component. Here the raw input signals are extracted processed, recognized and eventually merged (fused). We distinguish between feature-fusion, i.e. fusion that merges low-level feature information, and semantic-fusion, meaning the integration of semantic information derived from parallel inputs (Bui, 2006).

Dialogue Manager
The dialogue manager forms the core component of a dialogue system and is responsible for updating the dialogue context, providing context-dependent expectations for interpretation, interfacing with model processing to coordinate behaviour and reasoning and deciding what context to express next and when (Traum & Larsson, 2003).

General Knowledge
The term General Knowledge subsumes different types of models that are required and used by the dialogue manager and the fusion respectively, fission components. Here we can distinguish models that store dialogical knowledge (discourse history and task model) from models that store general knowledge (world model, task model, user model). The discourse history represents events that happened earlier in the conversation and may be needed to resolve references (e.g., anaphora and ellipsis). The task model is a representation of the current task the user performs and stores information to be gathered during the dialogue to reach a certain goal. A world model represents general background information that is inde-

pendent from the current domain whereas a domain model supports the system with specific information about the domain, e.g., the bathroom.

Fission
Fission basically describes the reversed process of Fusion. Foster (2002) classifies multimodal fission into the three categories of content selection and structuring, selection of modalities and the coordination of output. Fulfilling of these steps usually requires knowledge of the available output modalities and devices to support the planning in a reasonable way.

Output
The last component, Output, simply forwards the data from the previous component and usually has not much functionality. However, the output of channels should be coordinated so that the resulting output forms a coherent presentation (Bui, 2006). According to the usage of task and dialogue models, there are several ways to classify dialogue management approaches. We follow Bui (2006) and distinguish four categories:

- **Finite-state and frame-based approaches:** Bui (2006) describe finite-state based approaches as the simplest form to develop dialogue management systems, as “the dialogical structure is represented in the form of state transition networks in which the nodes represent the system’s utterances and the transitions between the nodes determine all the possible paths through the network”. The key advantage of these models is their simplicity but the lack of flexibility and hence naturalness makes them only feasible for certain small, closed domain tasks. An example of a system using this approach is the Nuance automatic banking system (McTear, 2002). Frame-based approaches allow a more flexible modelling of the dialogue as it is split into frames that take the analogy of a slot-filling task with a predetermined set of information to be gathered. Several systems have been developed based on the frame approach (Luz, 1999) but still, the systems’ next actions are fairly limited and are not capable to model natural dialogues.

- **Information state and the probabilistic approaches:** An information state-based theory of dialogue consists of five main components (Traum & Larsson, 2003): a description of informational components, a formal representation of these components, a set of dialogue moves that trigger the update of the information state, a set of update rules that govern the updating of the information state and an update strategy to decide which rule(s) to apply at a given point. A number of systems have been developed using the TrindiKit toolkit which implements this theory. Furthermore, the dialogue system of the SmartKom project has been developed based on a combination of the conversation games theory and information state based approaches, using the MULTIPLATFORM architecture as middleware.

- **Plan-based approaches:** Plan-based approaches are based on the theory that humans follow a certain plan to achieve a goal in the dialog, including changes to the mental state of the listener. Here, speech acts are used as input information (Searle, 1969) and the task of the listener is to recognize and respond to the speaker’s underlying plan. However, plan-based approaches are criticised, as the process of plan-recognition and planning are combinatorial hard to accomplish and may be not decidable.

- **Collaborative agent-based approaches:** Collaborative approaches are based on the fact that dialogues can be viewed as a collaborative process between agents (human and computers), where they work together to achieve a mutual understanding of the dialogue (Bui, 2006). The collaborative approach tries to capture the motivations behind a dialog and the mechanisms of dialogue itself. An example of an implemented version of this theory has been realised in systems like, e.g., COLLAGEN or TRIPS. The advantage of this theory lies in the capability to deal with more complex dialogues that incorporate problem solving but brings the disadvantage of more complex resources and processing than the other approaches.

Selecting the type of system for a certain domain is the first step in developing interaction systems. Obviously, the complexity of the underlying models increases from finite state- to agent-based approaches and conversational, cooperative agents would seem to be the obvious choice, as they imitate human behaviour. On the other hand, simpler dialogues (domains) often do not require much functionality of these systems and simpler designs may be as efficient with less effort in development and maintenance. However, in T4.1 we will target a bathroom environment. Therefore, we want to offer a natural interaction to the user and hence, we need a system that comes closest to model human conversation. We decided to use the ODP (Ontology-based Dialogue Platform) as a basis for our work in T4.1. This system has been developed by the company SemVox and emerged from the projects SmartKom and SmartWeb. Hence, this system is based on cooperative, information-state based & plan-based technologies and uses an ontological representation of its underlying models.

The focus of our research within the human-computer interaction lies in developing a canny combination of real-time interaction and more classical dialogue-based interaction. Real-time interaction means here a special form of short, event-triggered interaction that needs immediate response for the user to preserve effectiveness and usability (e.g., switching on the light by pressing a button). On the other hand, there exists dialogical interaction that needs the dialogue manager with all its knowledge to cope with the incoming information (e.g., resolve references and disambiguations, planning next steps). Furthermore, such interaction needs call backs to the user for clarification and hence, is allowed and inevitably required to take more time than the event-triggered interaction. This partition in two different ways of interaction has to be handled carefully as clashes between the two parts have to be avoided (e.g., the system is turning the light on and immediately switching it off again). This could be solved with a simple rule that decides which system (event-triggered interaction manager or dialogue-based interaction manager) has to handle the current input or by letting each system use the information and let it decide what to do (with a mediator to solve the mentioned clashes).
These architectural issues regarding IKS supported UIS will be applied within the design and the implementation of innovative, context-aware and natural user interfaces in the AmI use case (cf. Section 6 and 7).
The investigation in the AMI Case focus on what we called the "far out" vision of Interactive Knowledge. Under this perspective, the challenging issue of interacting with knowledge in standard CMS environments where the "environmental context of interaction" is limited to a classical desktop environment becomes a wider dimension encompassing semantics of the environment so that the following topics in conjunction with the general IKS objectives become relevant and can be fully addressed:

AmI Focus	**IKS Objectives**[6]
Orchestration of interaction with multiple input/output devices semantic knowledge on the base of the semantic representation of the system.	Offering key solutions or at least the necessary tools and methodology flexibility needed for addressing upcoming interaction scenarios in instrumented environments. (p. 41)
Multi-modal, multimedia semantic driven in-	Extending "*traditional content management*

[6] For more details on the IKS objectives see „Interactive Knowledge Stack for small to medium CMS/KMS providers: Description of Work".

teraction with content.	*systems to cover intelligent, multimodal and interactive behaviour at the user interface*" (p. 14)
Intuitive and simple interaction in a complex ambient scenario in "dialogical interaction unfriendly conditions".	Supporting "*intuitive and self exploratory interaction with knowledge at user level*" in ambient scenarios. (p. 12)
Scalable interaction framework for CMS.	Providing a valid "*semantics based infrastructure which is geared towards highly usable knowledge management systems at large scale and for a broad variety of uses*" (p. 15)

More in general, the realisation of the AMI Use Case has to be considered as an ideal investigation platform for testing and enhancing IKS interaction concept scalability in upcoming interaction scenarios.

3.4 Management of Devices in the Bathroom

Device management in AmI environments addresses the presentation of contents in a network of devices. This process can be separated into several steps, which are listed below:

(1) **Device Detection:** Which devices are available in the current context?
(2) **Service Detection:** Which services are provided by the currently available devices?
(3) **Device Selection:** Which device is the best presentation device for the presentation of a specific content in the current context?
(4) **Device Integration:** Integrating the device by integrating the software parts which are necessary to communicate with the device
(5) **Content Presentation:** Presentation of the content by using the provided services

Fuchs et al. (2006) provides an approach to automatically identify the best presentation device available in the current context by using semantic reasoning, user profiles and simple rights constraints. Hofer et al. (2003) describes a Java-based framework that provides information of specific devices based on the current context that can be used for device management. A rather hardware-based approach is presented by Eichhold et al. (2008), a combination of hardware and software for small Java-based sensor devices, which could be used as a basis for sensor devices in AmI environments. There are several standards for detecting devices and services (cf. step 1 & 2):

- **SSDP and UPnP -** The combination of Simple Service Discovery Protocol (SSDP) and Universal Plug and Play (UPnP[7]) enable the unique discovery, addressing and controlling of specific devices. Based on SSDP, a device can broadcast its availability within a standard computer network by using the User Datagram Protocol (UDP). Through a combination of several standard technologies on the network layer like IP, UDP, multicast and several protocols like TCP and HTTP devices can exchange notification messages, for instance, in XML or SOAP format. Every device is identified by a unique IP address and provides a device description that contains information like the producer, the serial number, URLs providing control, event and presentation services. Every service provides information about his commands and parameters. Several standards for device protocols are available to ensure standardized forms of service descriptions.

[7] http://upnp.org/sdcps-and-certification/standards/sdcps/

- **HAVi -** HAVi[8] is an audio and video device interoperability standard based on IEEE-1394 ("Firewire") networks. HAVi enables all devices within such a network to exchange and react on controller messages within the network. The major components are a device registry and four specific managers. The Event Manager manages the exchange of events between devices whereas the Stream Manager handles real-time stream exchange between several devices. The management of available resources and the scheduling of actions is addressed by the Resource Manager. The Device Control Module (DCM) Manager dynamically integrates and ejects devices.
- **Bonjour -** Similar to UPnP and HAVi, Bonjour[9] provides automatic device discovery and the access of device services in IP-based networks. Bonjour is a multicast DNS (mDNS), DNS-SD (Service Discovery) and IPv4LL implementation by Apple Inc. that represents a lightweight approach and requires a minimum of configuration. Since Bonjour also works with Unicast DNS Service Discovery it is not necessary that all devices are in the same local network and thus, also services over several sub-networks can be discovered and accessed. A Java bridge to use Bonjour via Java is available as well.

A more dynamic approach of providing and accessing services over IP-based networks is the use of Semantic Web Services (SWS). SWSs are modelled with the Web Service Modelling Ontology (WSMO[10]), which contains (1) ontologies used within the description, (2) Web Service descriptions including capabilities and interface descriptions, (3) client goals the service can handle and finally (4) mediators that ensure the interoperability between different ontologies. WSMO discovery web services can be found by providing a goal that should be achieved by a specific service.
By combining approaches of self-registering devices that offer specific services and address user models in a specific context, the next step is the selection of the best of all possible options of devices. This device selection process (cf. step 3) is also part of the device management and could be done by matching user preferences with predefined rule sets.
After the selection of the appropriate device(s) it is necessary to access the devices as described in the service description (cf. step 4). Even though the service interfaces are already well known to the system it is still a difference between the evaluation and control of a device, because the device needs specific software-modules or has to handle content that is provided. This could be realized, for instance, by the Open Services Gateway (OSGi[11]) which can provide both parts: (1) service descriptions and implementations which realize the service itself and (2) device-specific libraries and source code to perform the presentation of a content object or the sensor input analysis. This is specified in the OSGi UPnP Service Protocol.
The major research issue for the device management in T4.1 will be the combination of several sensing and presentation devices and the environment itself. As we have seen in the former Sections there are many technologies that can be used for each of these scenarios. But the challenge within the intelligent space that is being developed in T4.1 will be to combine these two forms of devices (sensors and presentation devices) with the representation of the current context, the user's preferences and finally, the situational discourse.
These architectural issues regarding the device management in IKS supported UIS will be applied within the design and implementation of the AmI use case (cf. Section 6 and 7).

[8] http://www.havi.org/pdf/white.pdf
[9] http://developer.apple.com/mac/library/navigation/index.html?filter=Bonjour
[10] http://www.wsmo.org/TR/d10/v0.2/d10.pdf
[11] http://www.osgi.org/About/Technology

4 Design of Knowledge Representation for the AmI Use Case

When building intelligent spaces such as our intended bathroom that is supported by UIS, the modeling of knowledge representations for encapsulating rooms, users, groups, roles etc. represents a fundamental design challenge (Peters & Shrobe, 2003; Lassila, 2005; Berners-Lee et al., 2001). This is due to the specific characteristics of UIS. They are dynamic, i.e. objects (information, persons, computing devices, and so on) within the environment are not stable over the life-time of the application (May, 2008); instead they might appear and disappear and create an evolving environment. It is also not guaranteed that the knowledge about an environment state is complete, accurate and up-to-date (May, 2008; Dooley et al., 2006). This leads to an "interoperability nightmare" (May, 2008; Lassila, 2005). So, UIS need evolving knowledge models that represent the "world" of the UIS, i.e., the evolving environment in real-time. The kind of services and information needed during the life-time of the UIS usually cannot be defined at design time; they have to be requested dynamically (May, 2008; Lassila, 2005).
According to the aforementioned characteristics of UIS, we assume that a knowledge representation for UIS has to be a surrogate of the real world that means it should contain representative objects for each item of the intelligent space (Peters & Shrobe, 2003) *(Req A)*. Furthermore, Lassila (2005) introduces that such knowledge representations should create a "serendipitous interoperability" that allows to discover and utilize objects not seen before in the intelligent space as well as an efficient operating in the sense of focusing on relevant knowledge items in specific situations (Lassila, 2005; Peters & Shrobe, 2003) *(Req B)*. Concerning the evolving knowledge models of an intelligent space, Davis et al. determines that the knowledge representation has to impose on what the UIS can see; how it looks at the world in the form of a set of ontological commitments (Davis, 1993; Lassila, 2005) *(Req C)*. Last, knowledge representations in an ubiquitous computing scenario have to deal with the fact that the UIS has to have temporally and spatially unrestricted access to the "information sphere" that means information stored locally or on the web (Maass & Janzen, 2011) *(Req D)*.

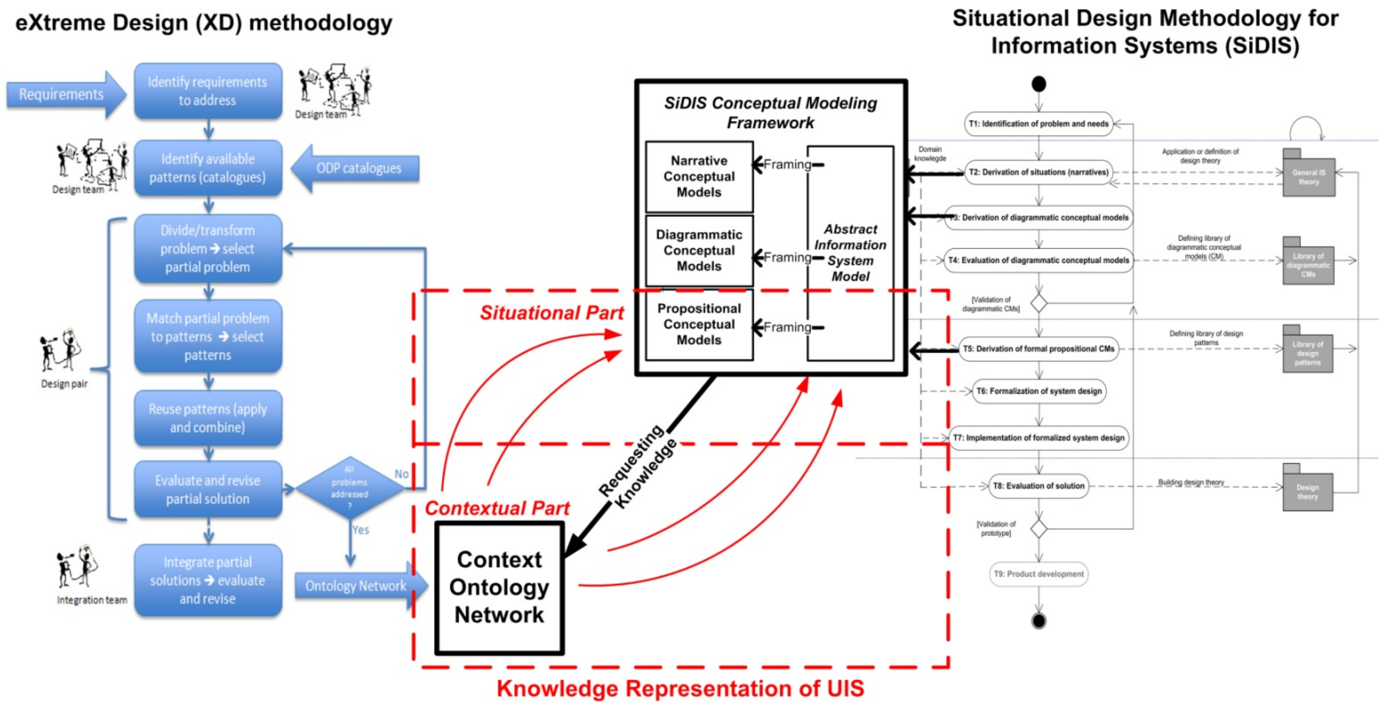

eXtreme Design (XD) methodology
Requirements
Identify requirements to address
Design team
Identify available patterns (catalogues)
ODP catalogues
Divide/transform problem → select partial problem
Match partial problem to patterns → select patterns
Design pair
Reuse patterns (apply and combine)
Evaluate and revise partial solution
All problems addressed ?
No
Yes
Integrate partial solutions → evaluate and revise
Integration team
Ontology Network
SiDIS Conceptual Modeling Framework
Narrative Conceptual Models
Diagrammatic Conceptual Models
Propositional Conceptual Models
Framing
Abstract Information System Model
Situational Part
Contextual Part
Requesting Knowledge
Context Ontology Network
Knowledge Representation of UIS
Situational Design Methodology for Information Systems (SiDIS)
T1: Identification of problem and needs
Domain knowlegde
T2: Derivation of situations (narratives)
Application or definition of design theory
General IS theory
T3: Derivation of diagrammatic conceptual models
T4: Evaluation of diagrammatic conceptual models
Defining library of diagrammatic conceptual models (CM)
Library of diagrammatic CMs
[Validation of diagrammatic CMs]
T5: Derivation of formal propositional CMs
Defining library of design patterns
Library of design patterns
T6: Formalization of system design
T7: Implementation of formalized system design
T8: Evaluation of solution
Building design theory
Design theory
[Validation of prototype]
T9: Product development

Fig. 4: Approach for designing the semantic knowledge representation for the intelligent bathroom

When designing the knowledge representation for the intelligent bathroom, our approach consisted of two main components: (1) formalized conceptual models developed within the application of **SiDIS** (Maass & Janzen, 2011) and (2) the Context Ontology Network developed by means of the **eXtreme Design (XD) methodology** (Presutti et al., 2009) (cf. Fig. 4). Therefore, we combine the artefacts of two design methodologies. In the following, we will explain the need for this combination.
We use SiDIS to ensure an integrated design of information systems considering users, social interactions, and physical surroundings besides plain technical issues (Maass & Janzen, 2011; Janzen et al., 2010). The design methodology offers a holistic view regarding situations supported by information systems that enables a comprehensive understanding of interactions within complex socio-technical systems, e.g., the intelligent bathroom. XD (Presutti et al., 2009) provides a highly flexible design methodology for computational ontologies, and in particular rapid prototyping of such ontologies. In addition, it focuses on reuse, and incorporation of Semantic Web best practices in the form of Ontology Design Patterns (ODPs).

An ODP is a modeling solution to solve a recurrent ontology design problem, and ideally it encodes some best practice in the field (Gangemi & Presutti, 2009). The combination of both methodologies has several advantages when developing knowledge representations for dynamic environments as described by the intelligent bathroom:

- Flexible design of computational ontologies, and in particular rapid prototyping of such ontologies that represent each item known by the intelligent space *(cf. Req A)*
- Semantic representation of concrete usage situations of the intended UIS by means of a conceptual modeling approach *(cf. Req B)*
- Dynamic linkage of situational structures and detailed semantic representations of items in the intelligent space at run-time *(cf. Req A&B)*
- Ontological framing of situations in the sense of “ontological commitments" that impact how the UIS looks at the world *(cf. Req C)*
- Highly modular ontologies, supporting temporally and spatially unrestricted access to information updated at run-time *(cf. Req D)*

Situational Design Methodology for Information Systems (SiDIS). The conceptual modeling framework of SiDIS consists of three conceptual models (CMs) as shown in Fig. 4: *narrative CMs, pattern-based diagrammatic CMs (Pre-Artifacts)*, and *formalized propositional CMs*. All CMs are framed by the *Abstract Information System Model (AISM)* (cf. Req C). AISM combines three classes for conceptual models of information systems (Lamb & Kling, 2003; Lechner & Schmid, 2001; Orlikowski & Barley, 2001) with the additional level of physical entities that is required for an Ubiquitous Information System (UIS) that will support the T4.1 intelligent bathroom environment. So, the model consists of *Information Sphere, Social System, Service System* and *Physical Object System* (Maass & Janzen, 2011). T4.1 intends to realize an UIS for the presentation of interactive knowledge within an intelligent bathroom environment. According to the AISM, the UIS will contain an Infosphere covering information objects retrieved from an IKS capable CMS. Furthermore, there will be agents, for instance users that take specific roles and interact with each other within the Social System. The AmI case UIS will offer several services (Service System) that operate partly via physical objects, e.g., devices or furniture in the bathroom (Physical Object System).
Based on translation procedures narrative CMs, i.e. the aforementioned descriptions of usage situations in natural language, are transformed into diagrammatic CMs, called Pre-Artifacts (cf. Fig. 4). Pre-Artifacts emphasize requirements on social structure, information objects, physical objects, and services in usage situations in a diagrammatic form. The core entities identified in narratives are assigned to these conceptual categories. Similarly, relations that connect these entities are specified. Finally, Pre-Artifacts are translated into the propositional CMs formalized in OWL (cf. Fig. 4). So, machine-processable CMs are gained

that can be used as part of the knowledge representation representing usage situations of the UIS (Maass & Janzen, 2011).

eXtreme Design (XD) methodology. XD (Presutti et al., 2009) denotes an approach to ontology engineering on the Semantic Web, which focuses on reuse of ODPs, and other resources on the Semantic Web, in order to rapidly produce an ontology network supporting a set of ontological requirements. XD deals with several types of requirements, i.e. both Competency Questions (CQs) (Gruninger & Fox, 1994), specifying the storage and retrieval needs for the KB, as well as contextual statements expressing constraints and restrictions, e.g. for the KB to be consistent, and reasoning requirements, expressing what information should be inferred by means of a reasoning engine. Requirements are elicited from user stories, similar to the narrative CMs of SiDIS. The requirements are divided into small coherent sets, where each set is treated separately to derive module, and then integrated into the overall ontology network through alignments (and possibly refactoring). Each module is realized through finding appropriate Content ODPs (CPs) (a specific type of ODP available as building blocks, e.g. represented in OWL) on the Semantic Web, e.g. in repositories like ODP portal (Gangemi & Presutti, 2011), specializing, extending and composing them into modules tailored to the intelligent bathroom. By applying XD the resulting ontology network becomes highly modular in its architecture, which both supports future changes and updates, as well as efficient handling of the KB when it is aligned to the situation representation, e.g. for a certain situation not all the modules need to be loaded and reasoned upon.

Resulting knowledge representation of the UIS. In summary, the resulting knowledge representation of the UIS consists of (1) propositional CMs of the SiDIS conceptual modeling framework, (2) specialized CPs, originating in the Semantic Web, and (3) a dynamic alignment between them, for their joint use within the system (cf. Fig. 4).
In short, propositional CMs represent the structure of specific bathroom situations derived from narrative CMs by means of a diagrammatic sub step. As one part of the intended knowledge representation, they consist of information objects, roles, services and physical objects in a computational form. These concepts represent just “pointers" to detailed knowledge covered by the second part of the knowledge representation - the context ontology network (cf. Fig. 4), i.e. propositional CMs “highlight" relevant entities that reside somewhere within the context ontology network and which concern a particular situation (cf. Fig. 5). Fig. 5 illustrate the difference between situation and context; the latter covers all information items known by the UIS whereas in a specific situation only a subset of items is relevant.
We refer to the set of propositional CMs as the *situational part* of the knowledge representation whereas the context ontology network represents the *contextual part* of the knowledge representation. The latter covers the overall and general context of the UIS and contains all information known by the UIS (cf. Req A). For instance, a propositional CM may declare that in the specific UIS situation an information object ”event suggestion” is required and point to the detailed representation of “current event suggestions” in the contextual part of the knowledge representation. The combination of the situational part with the contextual part meets the need for creating a “serendipitous interoperability“.The UIS knowledge representation allows the integration of unknown objects from the Semantic Web, through CPs, as well as focusing on relevant knowledge in specific situations (cf. Req B). The design and development of both parts will be elaborated in the next sections.

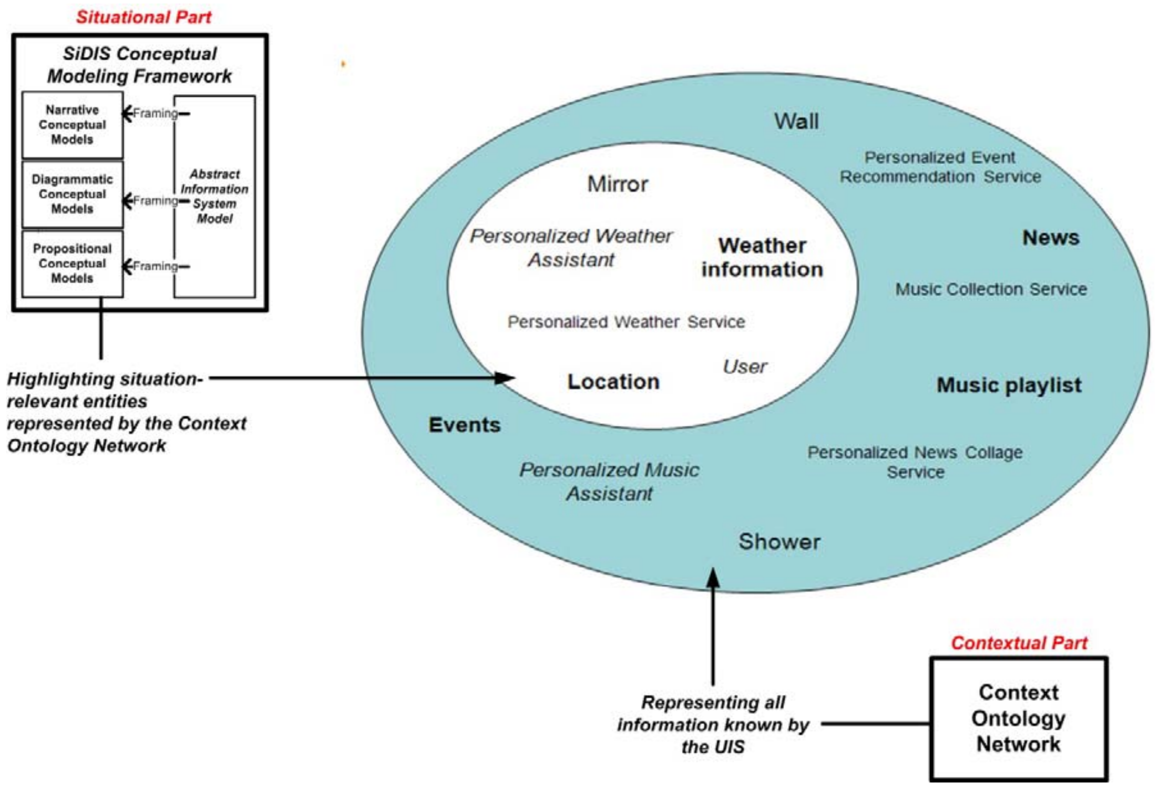

Situational Part
SiDIS Conceptual Modeling Framework
Narrative Conceptual Models
Diagrammatic Conceptual Models
Propositional Conceptual Models
Framing
Abstract Information System Model
Highlighting situation-relevant entities represented by the Context Ontology Network
Wall
Personalized Event Recommendation Service
Mirror
Personalized Weather Assistant
Weather information
News
Music Collection Service
Personalized Weather Service
User
Location
Music playlist
Events
Personalized News Collage Service
Personalized Music Assistant
Shower
Contextual Part
Context Ontology Network
Representing all information known by the UIS

Fig. 5: Illustration of context and situation regarding knowledge items of the UIS knowledge representation

4.1 Situational Part of the Knowledge Representation

Situations can be seen as instance-based descriptions of interactions between entities in an environment understandable by selected ontologies. Conceptual Models (CM) described by a Conceptual Modelling Language (CML) capture such situations (Hult et al., 2006). So, the situational part of the knowledge representation of the UIS consists of propositional CMs derived from narrative and diagrammatic CMs (Pre-Artifacts) (Maass & Janzen, 2011). As mentioned before, narrative CMs represent identified usage situations that are described by narratives in SiDIS task 2. The narrative CMs are translated into Pre-Artifacts (SiDIS task 3). Afterwards, the Pre-Artifacts are formalized as propositional CMs (SiDIS task 5).

In the following, the modelling of Pre-Artifacts will be elaborated. When beginning to model diagrammatic CMs in T4.1, we detected that we need a more guiding approach in SiDIS task 3 – the definition of diagrammatic CMs (Pre-Artifacts). Uncertainties occurred when defining the diagrammatic structures of narratives because of the large range of relation types as well as the comprehensive latitude in defining and modeling Pre-Artifacts. As described in D2.1 (Janzen et al., 2010), there were five steps for deducing a Pre-Artifact from a narrative: (1) Extraction of terms according to AISM, (2) Assignment of terms to categories, e.g., services, (3) Representation of categorized terms and their relations according to AISM, (4) Description of Pre-Artifact, and (5) Validation of Pre-Artifact based on competency questions. The difficulties in modeling Pre-Artifacts resulted from step 1-3. Extracting and assigning terms as well as the definition of relations based on AISM, offered a wide range of opportunities that had a negative effect on the resulting Pre-Artifacts. Depending on the modeling person, each Pre-Artifact possessed other structures and relations between the concepts.

So, we decided to integrate a further "conceptual level" between the modeling person and the AISM. This level consists of a pool of Pre-Artifact patterns that represent single situational structures based on AISM. The modeling person can use the expressiveness of the patterns to define Pre-Artifacts. So, the range of modeling opportunities is restricted and canalized in the sense of a better guidance.

4.1.1 Pre-Artifact Patterns

Analysis of Pre-Artifacts in several research projects (IKS amongst others) showed reoccurring structures similar to the notion of design patterns as used in architecture (Alexander, 1977) and Software Engineering (Dey & Abowd, 1999). They represent means for reusable CMs for Information Systems. We identified seven Pre-Artifact patterns (cf. Fig. 6) that are elaborated in the following (Maass & Janzen, 2011).

(1) Role Interaction Pattern: This pattern describes a situation in which two or more role-taking actors interact with one another by exchanging information objects supported by an interface service, e.g., mail communication between sender and receiver. To express the difference between services that represent direct interfaces to users and services that operate on an internal level exclusively, the notion of interface services and internal services is used. The interaction between roles is described by a generic property called *r-interacts*. The interface service is only used as a communication channel.

(2) Service takes Role Pattern: This pattern represents a situation in which a role is taken by an interface service. For instance, Wikipedia provides information and takes a role with connotated social attributes, such as reputation and credibility.

(3) Service uses Information Object Pattern: An internal or interface service receives information objects without human interventions. This is a simplification of the Service Interaction pattern with the distinction that a providing service is not important for the CM. For instance, stock information used by a local service and received from a cloud infrastructure.

(4) Service Interaction Pattern: This pattern describes the interaction relationship of two interface or internal services with no interaction with human actors. Within this interaction that is represented by *s-interacts,* an information object is used. The interaction relationship between services is described by *s-interacts* while roles are connected by *r-interacts* as mentioned before. For instance, a local temperature service sends data to a central weather service. In contrast to the Role Creates Information Object pattern, this pattern supports system designs that do not use role-based designs on service level.

(5) Role uses Information Object Pattern: In situations with direct manipulation of information objects, this pattern allows to express that a role receives an information object by using an internal or interface service. That means a role-taking actor can actively receive an information object supported by a service. For instance, a CEO uses a business intelligence service for accessing corporate sales information.

(6) Role uses Service Pattern: This pattern describes a situation with a role-taking actor creating an information object. Therefore, the actor uses a service that supports the creation of an information object, e.g., a nurse creates a status report for a patient by a healthcare reporting service.

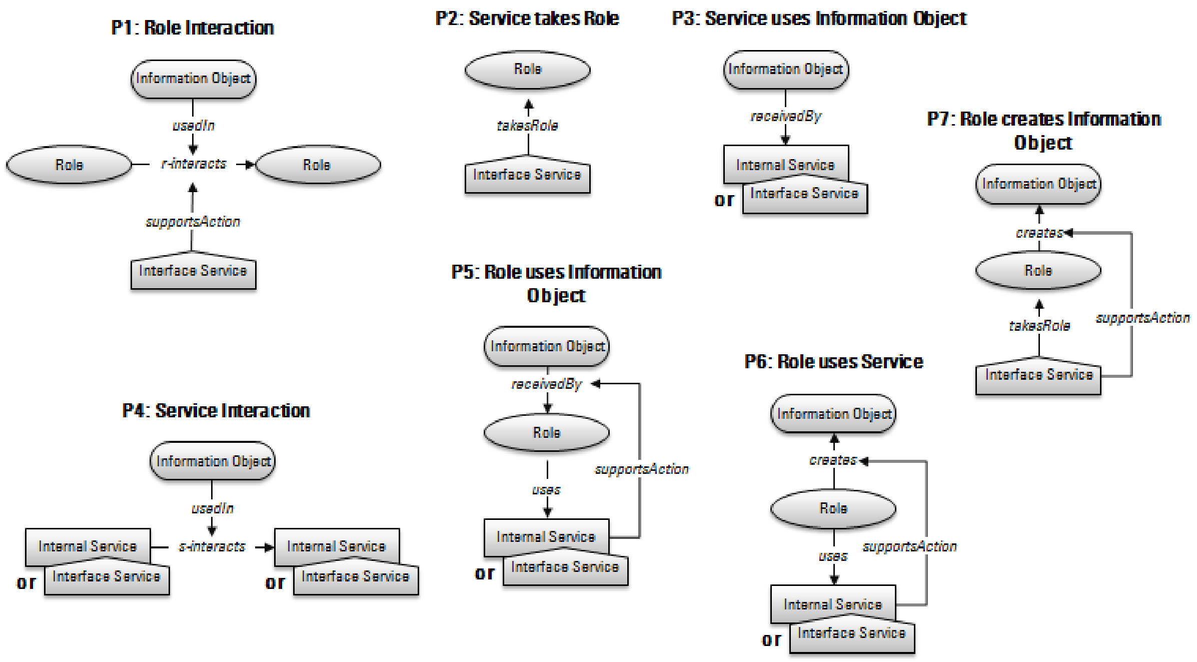


Fig. 6: Pre-Artifact patterns

(7) Role creates Information Object Pattern: By this pattern a service creates an information object by taking a role which links an information object to a service. This pattern supports role-based system designs. For instance, a vital sign monitoring system can take a role that allows it to create emergency alerts. Created alerts are directly linked with this service via a role.

4.1.2 Defining Pre-Artifacts by means of Patterns

The construction of Pre-Artifacts is guided by a method consisting of five steps regarding the instantiation and combination of Pre-Artifact patterns to define the diagrammatic CM (Maass & Janzen, 2011). Each step pursues a sub goal for constructing the Pre-Artifact step by step and proclaims specific Pre-Artifact patterns that help to achieve the objectives (cf. Tab. 1). Next, an example will be given for deriving a diagrammatic CM from a narrative CM based on Pre-Artifact patterns.

Tab. 1: Appliance of Pre-Artifact patterns within steps of defining Pre-Artifacts

Patterns P / Steps	Step 1	Step 2	Step 3	Step 4	Step 5
P1: Role Interaction	-	x	-	-	-
P2: Service takes Role	-	-	x	-	-
P3: Service uses Information Object	-	-	-	x	-
P4: Service Interaction	-	-	-	x	-
P5: Role uses Information Object	-	-	-	-	x
P6: Role uses Service	-	-	-	-	x
P7: Role creates Information Object	-	-	x	-	-

Step 1: Definition of *Information Objects* in *Infosphere*. All information objects that occur in the narrative have to be defined as Information Objects (IO) in the Infosphere. Attention should be paid to the aspect that information objects that will be created in a situation always have generic sources. Fig. 7 shows this modeling step by means of an exemplary Pre-

Artifact that shall represent the narrative of situation 1: “*It's Thursday morning. I get site-specific weather information when I am brushing my teeth in the bathroom.”* The figure shows that the modeling person has specified the goal “*Getting weather information for user‘s location*” that is assigned to the user in the situation. Furthermore, the information object *site-specific weather information* is defined. This information object has to be created in the situation based on the required information objects *global weather information* and *location*.

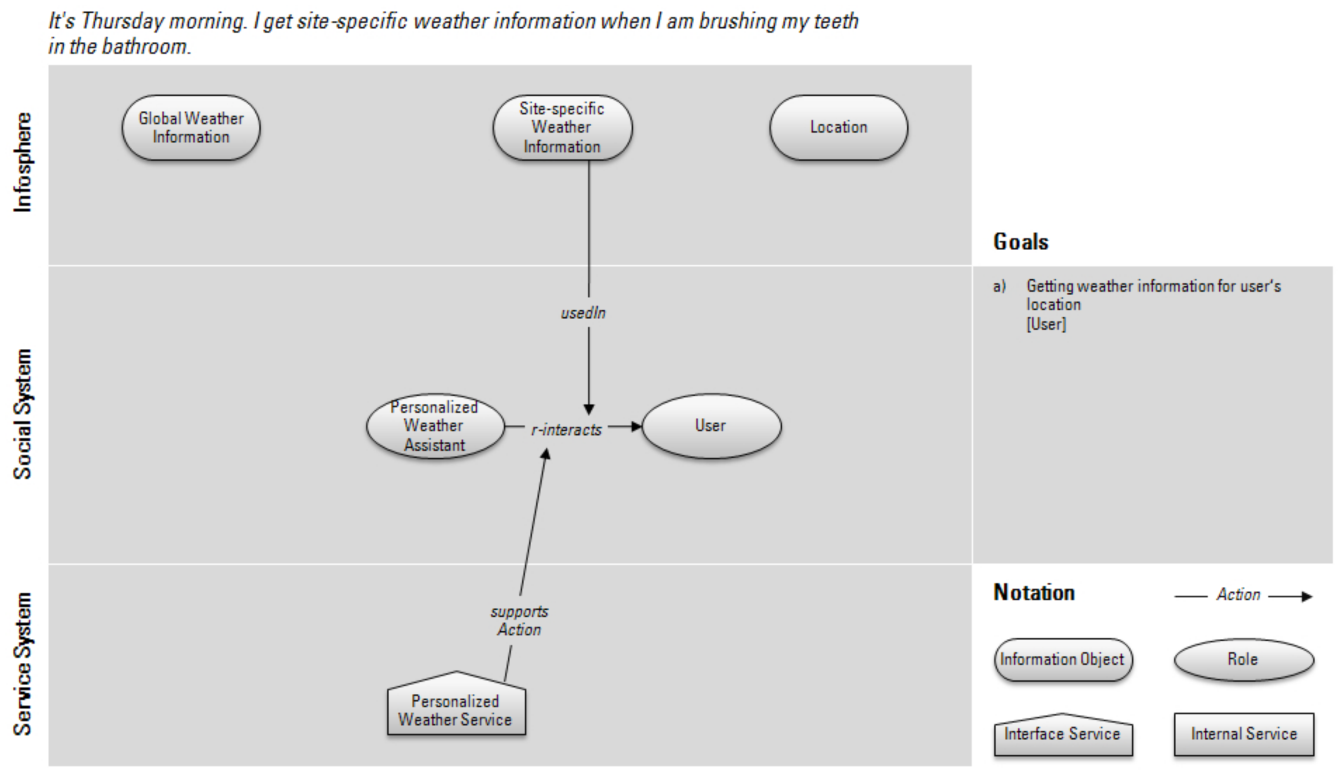

It's Thursday morning. I get site-specific weather information when I am brushing my teeth in the bathroom.
Infosphere
Social System
Service System
Global Weather Information
Site-specific Weather Information
Location
usedIn
Personalized Weather Assistant
r-interacts
User
supports Action
Personalized Weather Service
Goals
a) Getting weather information for user's location [User]
Notation
Action
Information Object
Role
Interface Service
Internal Service

Fig. 7: Definition of user-system interaction related to information objects

Step 2: Definition of user-system or user-user interactions related to *Information Objects*. Within this step, interactions between users or user and system related to new generated information objects have to be defined. These interactions take place between *Roles* in the *Social System* exclusively. Interactions between user and system are always supported by a service of the *Service System*. The requirements of this step are fulfilled by the application of the *Role Interaction pattern* exclusively. In the exemplary Pre-Artifact (cf. Fig. 7) an interaction between a *Personalized Weather Assistant* and the *User* was modeled that is supported by a *Personalized Weather Service*. Subject of the interaction is the IO *site-specific weather information*.

Step 3: Definition of *Roles* taken by *Services.* Next, an interface service has to be defined that takes a role for creating the new information object that will be used in the interaction. Therefore, the service has to take a role in the interaction. To manage this step, the *Role Creates Information Object pattern* is applied to define the creation of the information object by a role taken by a service. In our example, the Personalized Weather Service takes the role of the Personalized Weather Assistant that creates the IO site-specific weather information. The interface service supports this action indirectly (cf. Fig. 8). To express the plain role-taking by a service without a creating function, the *Service takes Role* pattern can be applied.

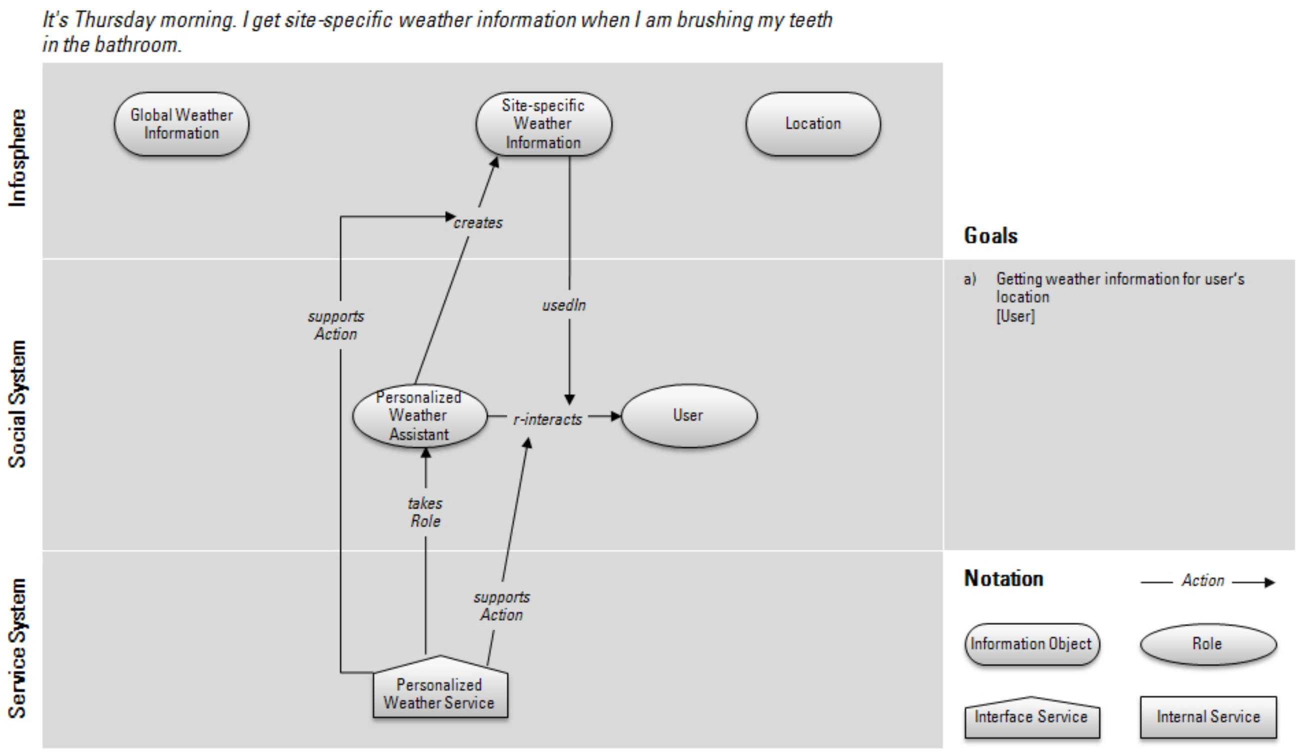

It's Thursday morning. I get site-specific weather information when I am brushing my teeth in the bathroom.
Infosphere
Social System
Service System
Global Weather Information
Site-specific Weather Information
Location
creates
supports Action
usedIn
Personalized Weather Assistant
r-interacts
User
takes Role
supports Action
Personalized Weather Service
Goals
a) Getting weather information for user's location [User]
Notation
Action
Information Object
Role
Interface Service
Internal Service

Fig. 8: Definition of roles taken by services

Step 4: Definition of supporting *Internal Services.* To create new information objects, generic information sources are needed as mentioned before. The interface service that supports the creation of a new IO needs access to these sources. Therefore, *Internal Services* for all remaining information objects in the Infosphere are specified. The interaction between services regarding the information objects is realized by applying the *Service Interaction or Service uses InformationObject* pattern. The exemplary Pre-Artifact (cf. Fig. 9) shows the definition of two internal services *Weather Service* and *User Context Service* that feed a *Personalized Weather Service* with global weather information and location data.

Step 5: Definition of user initiative. If a user role initiates an interaction with the system that means using the system in a proactive way, this situation is modeled by using the *Role uses Service* or *Role uses Information Object* pattern (not required cf. Fig. 9). The role uses a service to create or receive an information object, for instance, the user wants to leave a message for another user. This action is indirectly supported by a service.

According to the results of the user study in T2.1 (Janzen et al., 2010), we translated the narratives of the best-ranked situations 1, 6, and 11[12] into Pre-Artifacts. In case of high complexity of a narrative, multiple Pre-Artifacts were generated to avoid an overloading of a single diagrammatic structure.

[12] Situation 1 splitted into Pre-Artifact 1A-C; situation 6 splitted into Pre-Artifact 6A-C; situation 11 represented by Pre-Artifact 11A

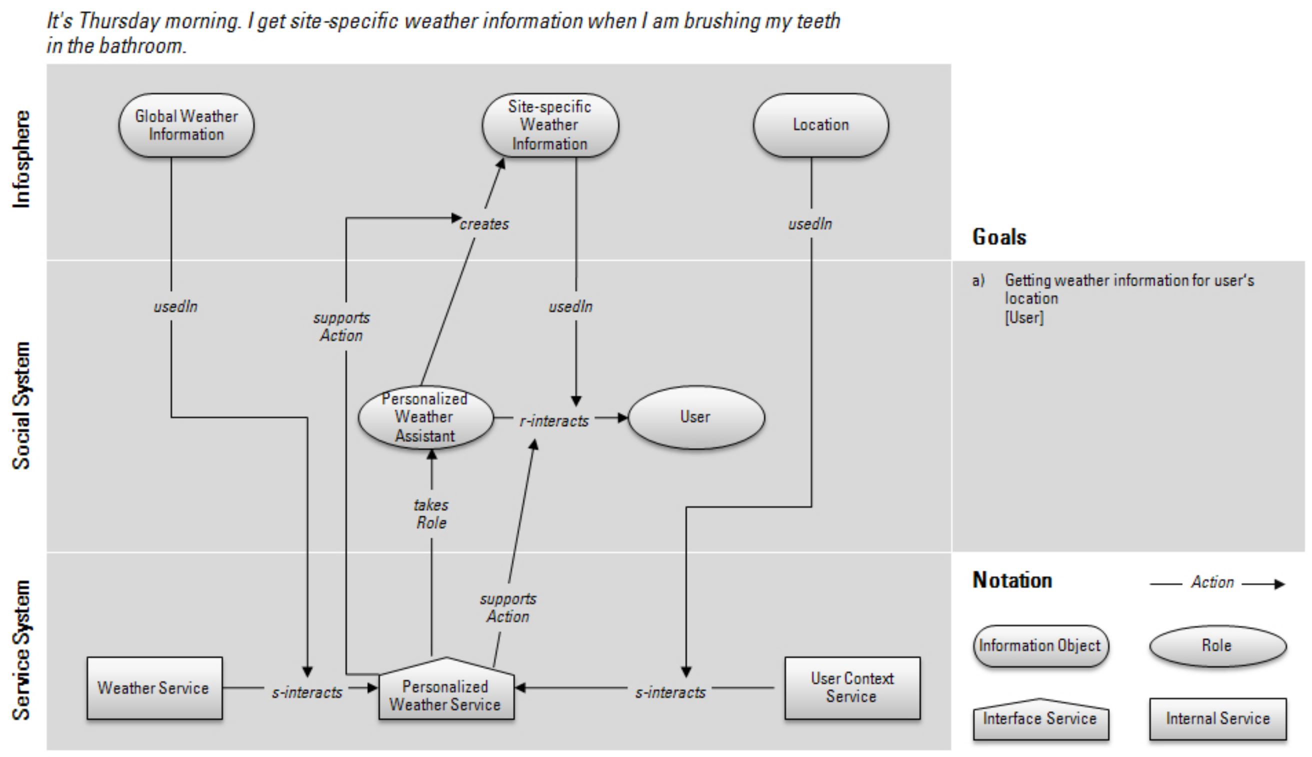

It's Thursday morning. I get site-specific weather information when I am brushing my teeth in the bathroom.
Infosphere
Social System
Service System
Global Weather Information
Site-specific Weather Information
Location
creates
usedIn
usedIn
supports Action
usedIn
Personalized Weather Assistant
r-interacts
User
takes Role
supports Action
Weather Service
s-interacts
Personalized Weather Service
s-interacts
User Context Service
Goals
a) Getting weather information for user's location [User]
Notation
Action
Information Object
Role
Interface Service
Internal Service

Fig. 9: Definition of supporting internal services

4.2 Contextual Part of the Knowledge Representation

In the context of designing the contextual part of the knowledge representation, ontological requirements were specified. The requirements considered, were of three types:

1. Competency Questions (CQs) (Gruninger & Fox, 1994) – Expressing queries that the KR needs to be able to respond to, e.g. as SPARQL queries submitted against it.
2. Contextual Statements – Expressing constraints and restrictions that should hold for all the facts stored in the KR, i.e. in order for it to be consistent.
3. Reasoning Requirements – Expressing the inferences that the KR should enable, through specifying what input to the KR will be provided and what output the ontology network should produce based on that input.

An example of a CQ is “At what time and day of the week did a certain event occur?”, while the constraint that every event recorded in the KR has to have a start time, while end times are optional (since we record on-going events), is an example of a contextual statement. A reasoning requirement related to events could be to be able to classify user actions into certain event categories, e.g. to identify that an event involving a user walking to the sink picking up an electric toothbrush and turning it on belongs to the abstract category “user starting to brush teeth”.

The Pre-Artifacts as well as the Software Requirements of T4.1 (cf. Deliverable of T2.1 and Appendix A) were used as clarifications of the narratives and to guide the requirements elicitation. The Pre-Artifacts impact the ontological requirements by detailing the interactions described in the narratives, e.g. explicitly specifying what physical objects and information objects we need to be able to store facts about. Software Requirements can help to clarify how the services existing in the Pre-Artifacts need to be represented. For instance, some services may result in reasoning requirements for the ontology, while other services are software services that use the facts of the context representation (through the alignment to the situations), i.e. resulting in CQs and contextual statements rather than reasoning requirements. Initially competency questions for the situations 1, 6 and 11 were determined. The competency questions (CQ) were assigned to situations and slightly generalized to represent all three situations (cf. Tab. 1).

Tab. 1: Competency questions of situation 1

Anna gets site-specific weather information when she is brushing her teeth in the bathroom. Based on weather information and her calendar, free-time event suggestions are given, e.g. "Today, 8 p.m. - Miss Marple Night at CinemaOne. Do you want to order tickets?"

	Competency question	**Competency question adjusted to situation**	**Possible answer**
1	What people are in the current location?	Who is in the room?	Anna
2	What is the spatial orientation of this person?	In what direction is Anna looking?	Towards the mirror
3	What is the current location?	Where are we?	In the bathroom, by the mirror
4	What are the available devices in this location?	What are the available screens in this bathroom?	A mirror screen, and …
5	What action is this person performing at the moment?	What is Anna doing?	Entering the room/Brushing her teeth
6	Who was performing what action at what time?	What was Anna doing at 9am?	Brushing her teeth
7	What event just happened?	What event type can be inferred from the current context and the new fact x?	Anna entered the room/Anna started brushing her teeth
8	Who participated in this event?	(What person brushes her teeth right now?	Anna
9	What was the previous sequence of actions of this person?	What did Anna do before brushing her teeth?	She entered the room
10	Where does this person live?	In what city does Anna live?	In Berlin
11	What are the planned events of this person during a certain time?	What are the calendar entries of Anna for today?	At 10 she has a meeting, then she has free time
12	What are the capabilities of these devices?	What are the capabilities of this screen?	Show content of type x, with a resolution of y
13	What are the available devices in the current location for presenting this content?	(Where can this weather information be displayed?	On the mirror
14	How should this content best be presented on a particular device?	How should this weather information be presented on the mirror?	With resolution x, font size y
15	What is the presentation of this content in a certain modality?	What is the presentation of this weather information on a screen/as audio/…?	<A realization of the content in that modality?>
16	What are the activity types/event types that are preferred by this person?	What are the free-time activity types preferred by Anna?	Sports and cinema
17	What are the content types that are preferred by this user in this particular situa-	What does Anna want to see on a Thursday	Weather information and free-time activity

	tion?	morning when she has free time in her calendar and she is currently brushing her teeth?	proposals
18	What are the preferred system actions when this particular person is performing this particular action?	What should the system show when Anna is brushing her teeth?	Weather information
19	What type of content is this?	What type of content is this?	Weather information
20	What location is this content related to?	What city is this weather forecast for?	Berlin
21	What topic does this content item have?	What is the topic of this movie?	It is about historical events in Rome
22	What are the most recent content items of this category?	What is the most up-to-date weather forecast?	<Return reference to a content item>
23	What is the most recently updated content of this particular type connected to this location?	What is the current weather forecast for this particular location?	<Return reference to a content item>
24	What are appropriate activity types for this person given the current situation?	What are the free-time activity types suitable at the current time?	Cinema
25	What are the available activities/events for this person given the current situation?	What free-time activities of a particular type is available in this city at this time?	A movie show tonight at 8pm
26	Where does a specific activity/event take place?	Where is this movie shown?	At CinemaOne
27	At what time does a specific activity/event take place?	When is this movie shown?	At 8pm tonight
28	What was the previous sequence of content items presented to this person?	What was presented before this movie proposal?	Weather information for Berlin
29	What are the available services for this content type?	What are the available weather services?	www.yr.no
30	What are the available services for this activity type?	What are the available cinema booking services?	www.sf.se
31	What user actions does a certain service offer?	What are the actions a user can perform on the cinema service?	View trailers, book tickets
32	What content is related to this service action?	What content is related to this movie-ticket booking service for this particular movie?	A trailer of the movie, the seat plan of the theatre

Subsequently, contextual statements as well as reasoning requirements were identified for the competency questions.
The contextual part of the KR can be seen as the knowledge base (KB) of facts that provides the basis for describing both the concrete bathroom environment and the content that is handled in the bathroom's interactions with the users. Hence, the context representation needs to on one hand provide means to store and record information about the physical environment in the bathroom, including the user's actions and preferences, and on the other hand the content that is handled by the system. As illustrated above, the situation representation is later used to connect parts of the context representation into typical "situations" that the system should handle. In practice, ontology networks are aligned that represent the context and situation KRs respectively.

Conceptually, the context representation is designed as an ontology network, consisting of loosely coupled ontology modules that we call AmI ODPs. An ontology network is an ontology that is composed of a set of (interconnected) ontology modules that are in turn ontologies themselves. An ODP is a modelling solution that solves a recurrent ontology design problem, as mentioned in Section 2.3, and ideally it encodes some best practice in the field. ODPs exist in many different forms (for an overview see (Gangemi & Presutti, 2009)), where one type is Content ODPs (CPs). CPs encode *conceptual*, rather than *logical* design patterns. In other words, while logical ODPs solve design problems independently of a particular conceptualization, CPs propose patterns for solving design problems for the domain classes and properties that populate an ontology, therefore addressing *content* problems (Gangemi, 2005). CPs are instantiations of logical ODPs, featuring a non-empty signature. In principle, CPs do not depend on any specific language, however in order to reuse them as *building blocks*, they have to be implemented in some way. In the context of this project, we deal with CPs for the Semantic Web. Hence, we have used OWL as the formalism for representation. CPs can exist on several levels of abstraction, and although most of the highly reusable ODPs that are available online, e.g. at the ODP Portal[13], are quite abstract and general, they can quite easily be specialized into domain specific ODPs, e.g. the AmI ODPs. The context representation for the AmI case can be seen as an ontology network composed of domain-specific specializations of general CPs.

The resulting ontology network, i.e. the context ontology network, currently (in its initial version) consists of 47 distinct ontology modules, i.e. the network is composed of 47 loosely coupled modules. Each module is identified by its own URI, since the modules are OWL ontologies themselves. Most of the modules contain fewer than ten locally defined classes and properties, but overall some modules are quite large due to the import and reuse of other modules and general ODPs. The complete network is not easily visualized, but to illustrate the structure, in Fig. 10 we show the structure of one of the modules, called *Context*, which is in itself a small ontology network. The *Context* module is one of the core modules of the context ontology network, and it models the user-specific physical context in the bathroom, i.e. the user's location and interaction with different devices at a certain point in time.

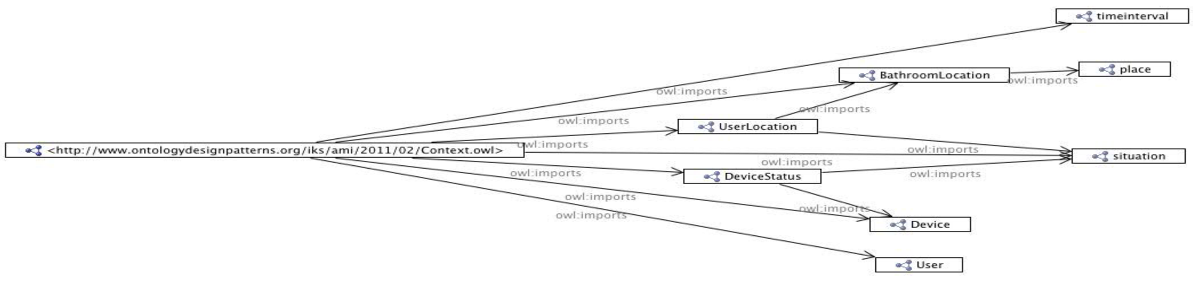


Fig. 10: The ontology network representing one of the modules.

As can be seen in Fig. 10, the network is based on three general ODPs; *timeinterval*, *place*, and *situation* (boxes in the diagram represent ontologies, and arrows the owl:imports property). It also reuses the two general AmI ODPs about modelling devices (*Device*) and users (*User*). The *place* ODP has been specialized in an AmI ODP called *BathroomLocation*, where the very general *place* pattern is extended with notions of indoor locations in a room and the orientation of something, e.g. facing an object or being turned away from an object. Next, the *UserLocation* AmI ODP uses the general ODP *situation*, which is a CP for modelling n-ary relations, and the *BathroomLocation* module to express facts about the location of a user within the bathroom. In the overall *Context* module, this is then extended with a time-index, letting us express the user's location at a certain point in time. The *DeviceStatus* mod-

[13] http://www.ontologydesignpatterns.org

ule also reuses the *situation* ODP to express the states of different devices in the bathroom, which is in the *Context* module integrated with the user model, i.e. expressing that a user can affect those states and they belong to the user's physical context. For instance, a user may turn on an electric toothbrush, and subsequently a turned-on electric toothbrush is part of that user's physical environment.

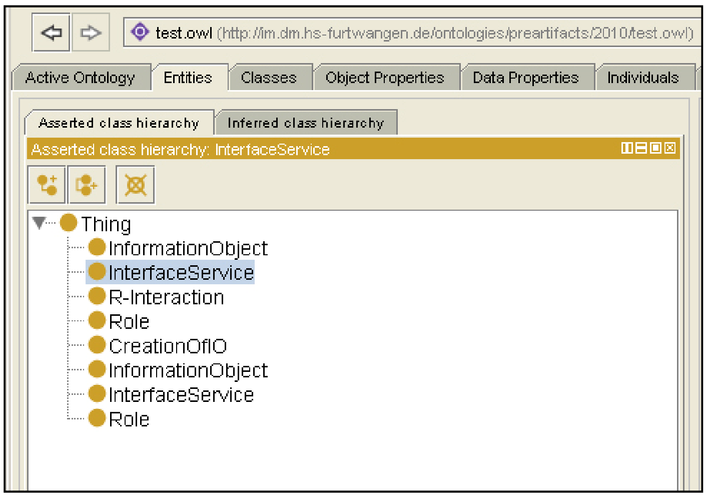
5 Development of Knowledge Representation for the AmI Use Case

Next, we formalized the contextual and situational part of the knowledge representation for the AmI use case based on the conceptual design of the knowledge representation introduced in Section 4. In the following sub sections, we will present the procedure of formalizing both KR parts that means Pre-Artifacts (cf. Section 5.1) and the Context Ontology Network (cf. Section 5.2). Last, we will introduce the concept of linking both parts of the knowledge representation within the AmI case system (cf. Section 5.3).

5.1 Formalization of Situational Part

After modeling Pre-Artifacts based on the narrative CMs by means of the aforementioned "5 steps" (cf. Section 4.1.2), the Pre-Artifacts were translated into the third type of CM of our methodological approach - the propositional CMs (cf. Fig. 4). The objective of this translation is the creation of specifications for later system designs (Purao et al., 2003) as well as machine-processable CMs that can be verified (Bera et al., 2010) (Maass et al., 2011b). Considering the method of formalization, there are several opportunities, for instance Unified Modeling Language (UML), entity-relationship model (ER) or a formalization based on ontologies by means of RDF or OWL. Bera et al. (2010) identified some unique features of OWL that are not available in ER model and in UML, e.g., OWL is implementable that means OWL ontologies are machine-readable, and thus computational. Furthermore, OWL constructs are independent, i.e. classes can exist independent of instances or properties and properties are independent of classes. Beside these advantageous features of OWL, there are also difficulties in using OWL for the formalization of Pre-Artifacts. Bera et al. (2010) determine that there are no clear rules how to map from domain information as represented by Pre-Artifacts to OWL constructs similar to the intended propositional CMs (Maass & Janzen, 2011). We also faced these problems within T4.1. Therefore, we tested three different approaches of translating Pre-Artifacts into propositional CMs by means of OWL constructs by modeling three exemplary Pre-Artifact patterns (Role Interaction, Role creates Information Object, Service Interaction) that were used for generating the propositional CM of the exemplary Pre-Artifact in Fig. 9.

Approach 1. This option was realized by representing each Pre-Artifact pattern by a unique formal propositional model. The approach leads to redundant concepts when integrating these propositional models into a complex propositional CM, e.g., the exemplary Pre-Artifact. A modeling person had to specify equivalences to resolve these redundancies; e.g., in our example (Fig. 9) the

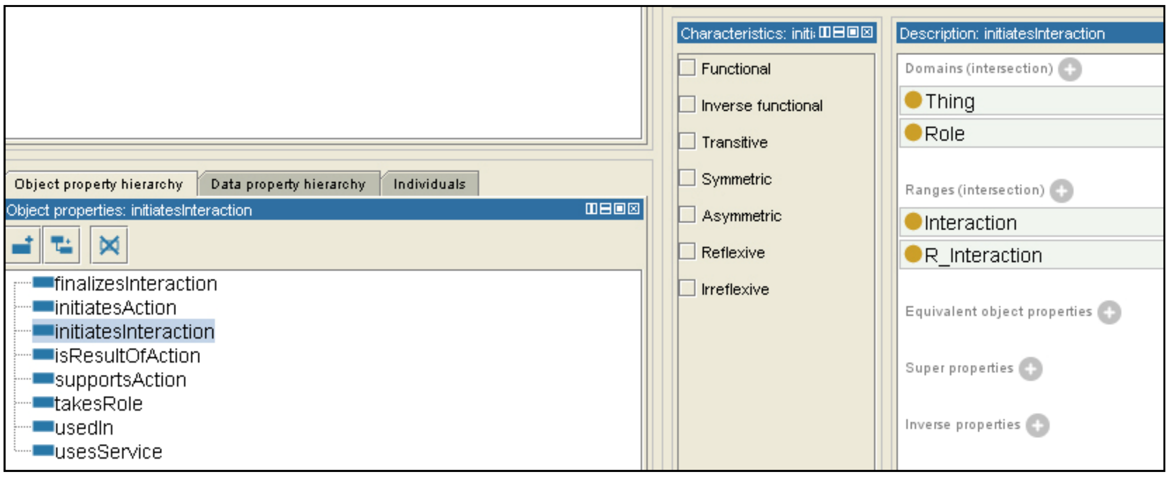


Thing
- InformationObject
- InterfaceService
- R-Interaction
- Role
- CreationOfIO
- InformationObject
- InterfaceService
- Role

Fig. 11: Redundant concepts (screenshot of Protégé tool)

concept type “Role” occurred several times because two of the imported patterns contain this concept type (cf. Fig. 11). This procedure demonstrates the aforementioned lack of modeling guidelines and constraints (Bera et al., 2010).

Approach 2. Following the approach by Bera et al. (2010) of using a “philosophical ontology” to derive guidelines on how OWL constructs can be applied in the modeling of propositional conceptual models, an Abstract Information System (AIS) Ontology based on AISM was created that represents a “vocabulary” and generic object properties of Pre-Artifact patterns. The AIS Ontology consists of 12 concept types and eight generic object properties. It represents basic entities of AISM: *InformationObject, Role* and *Service* with sub-classes *Interface* and *Internal Service*. Furthermore, a super-class *Action* is defined that contains further sub-classes that specify diverse types of pattern actions: *Creation, Receiving* and *Interaction* with sub classes *R_Interaction* and *S_Interaction*.



Object properties: initiatesInteraction

- finalizesInteraction
- initiatesAction
- initiatesInteraction
- isResultOfAction
- supportsAction
- takesRole
- usedIn
- usesService

Characteristics: ☐ Functional ☐ Inverse functional ☐ Transitive ☐ Symmetric ☐ Asymmetric ☐ Reflexive ☐ Irreflexive

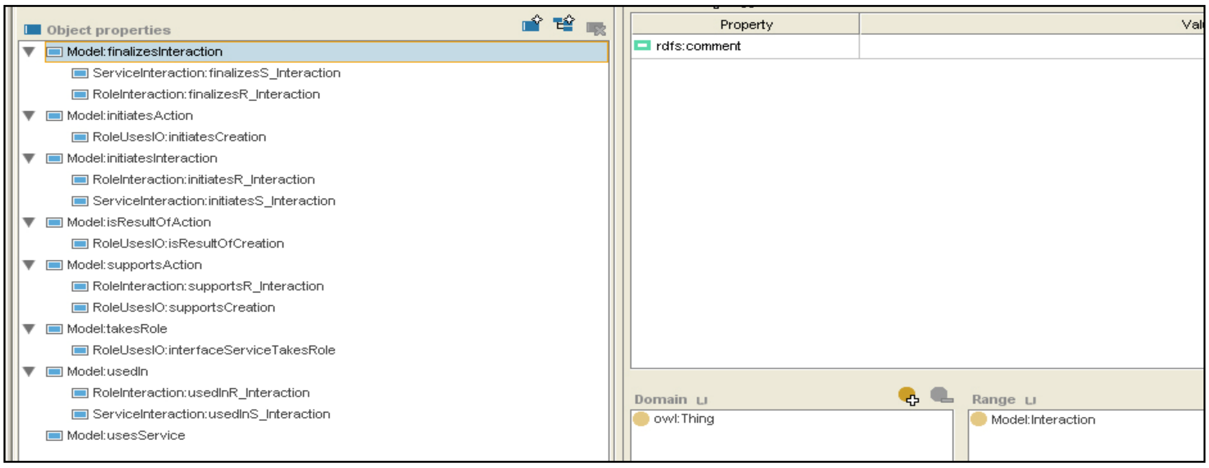
Domains (intersection)
● Thing
● Role

Ranges (intersection)
● Interaction
● R_Interaction

Fig. 12: Ambiguous statements (screenshot of Protégé tool)

The decision to model most of the pattern relations by means of additive concepts was due to the fact that these relations represent three-way connections. A second opportunity would be to use property chains in OWL 2 that support transitive relationships between objects (Motik et al., 2010). The advantage of the former opportunity lies in integrating actions as specific concepts in the social system. This allows a differentiated consideration and extensibility by further properties. Furthermore, the model consists of 8 generic object properties: *initiatesInteraction, finalizesInteraction, initiatesAction, isResultOfAction, supportsAction, takesRole, usedIn* and *usesService*. Each pattern ontology imports the AIS Ontology. Afterwards, the pattern ontologies specify relevant generic object properties with additional concepts. When integrating the pattern ontologies in the propositional CM, ambiguous assignments of object properties to specific patterns occurred (cf. Fig. 12). Because of lack of clear results and statements, the second approach is not a proper solution to handle the lack of modeling guidelines.

Approach 3. In this approach, the notion of the AIS Ontology as well as the integration of this “vocabulary” into pattern ontologies was adopted. But, for the specification of pattern-specific object properties based on the generic properties of the model, inheritance structures of object properties were used. That means each pattern defines sub properties of the relevant object properties imported from the model. Therefore, super-properties and concepts of the AIS Ontology remain unchanged. In this context, the OWL feature is used, that OWL constructs are independent, i.e. properties can exist independent of classes (Bera et al., 2010). Based on this approach, clear assignments of specified object properties to specific patterns are realized. Conceptual modelers will be supported by modeling guidelines because of a canalization of modeling options. The propositional conceptual model can be modeled in an incremental way by importing patterns step by step according to the requirements of the Pre-Artifact.

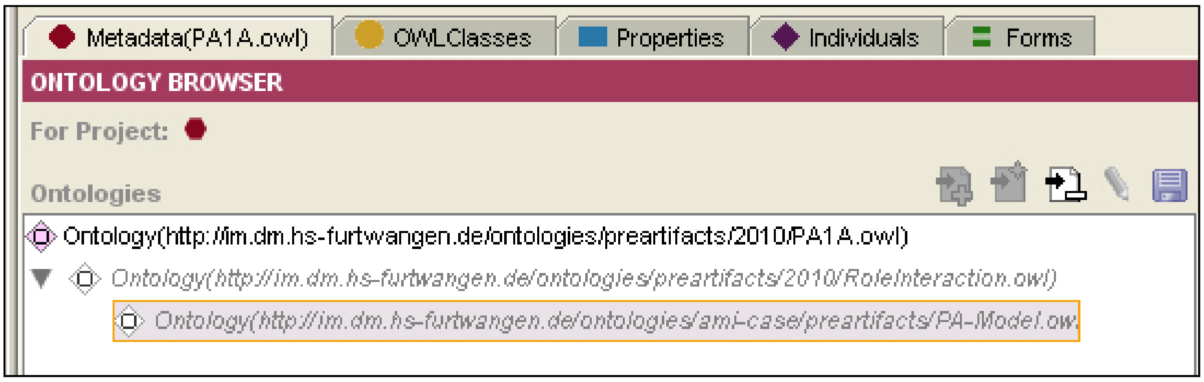

Object properties
- Model:finalizesInteraction
  - ServiceInteraction:finalizesS_Interaction
  - RoleInteraction:finalizesR_Interaction
- Model:initiatesAction
  - RoleUsesIO:initiatesCreation
- Model:initiatesInteraction
  - RoleInteraction:initiatesR_Interaction
  - ServiceInteraction:initiatesS_Interaction
- Model:isResultOfAction
  - RoleUsesIO:isResultOfCreation
- Model:supportsAction
  - RoleInteraction:supportsR_Interaction
  - RoleUsesIO:supportsCreation
- Model:takesRole
  - RoleUsesIO:interfaceServiceTakesRole
- Model:usedIn
  - RoleInteraction:usedInR_Interaction
  - ServiceInteraction:usedInS_Interaction
- Model:usesService

Property	Val
rdfs:comment	

Domain: owl:Thing
Range: Model:Interaction

Fig. 13: Inheritance structure of object properties when importing 3 patterns (screenshot of Protégé tool)

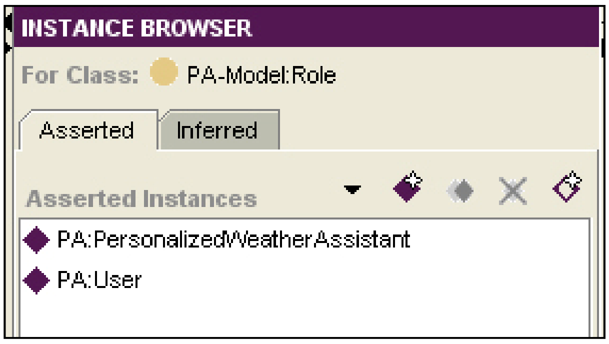
Within T4.1, the aforementioned third approach was applied. To exemplify the modeling the propositional CMs based on the specified Pre-Artifacts, we will translate our exemplary Pre-Artifact (cf. Section 4.1.2, Fig. 9) into a propositional CM (D2.1 - Janzen et al., 2011).
After generating an empty OWL file in an ontology development tool, e.g., Protégé, we use the expressiveness of Pre-Artifact patterns represented as single ontologies by importing them by their URL. Similar to the procedure of defining Pre-Artifacts in five steps, we start with applying the *RoleInteraction* pattern and import its formalized model (cf. Fig. 14).



Ontology(http://im.dm.hs-furtwangen.de/ontologies/preartifacts/2010/PA1A.owl)
- Ontology(http://im.dm.hs-furtwangen.de/ontologies/preartifacts/2010/RoleInteraction.owl)
  - Ontology(http://im.dm.hs-furtwangen.de/ontologies/ami-case/preartifacts/PA-Model.ow

Fig. 14: Import of Pre-Artifact pattern RoleInteraction (screenshot of Protégé tool)

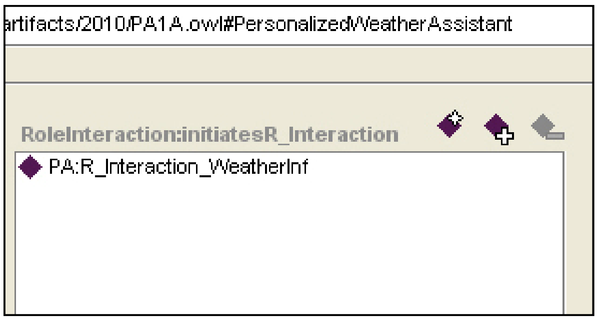
Then, we instantiate the relevant concepts of the pattern, i.e., we create instances of the concept "Role" named "User" and "PersonalizedWeatherAssistant" (cf. Fig. 15). To represent the interaction between User and PersonalizedWeatherAssistant, an instance of the concept "R-Interaction" has to be created.
For linking both roles with the instance of R-Interaction (cf. Fig. 17), the formalized pattern offers the specified object properties "in-



PA:PersonalizedWeatherAssistant
PA:User

Fig. 15: Instantiation of roles - User and PersonalizedWeatherAssistant (screenshot of Protégé tool)

itatesR_Interaction" and "finalizesR_Interaction" that inherit from the super-properties "initiatesInteraction" and "finalizesInteraction" (cf. Fig. 16).

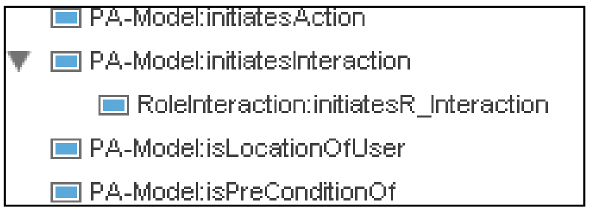


artifacts/2010/PA1A.owl#PersonalizedWeatherAssistant

RoleInteraction:initiatesR_Interaction

◆ PA:R_Interaction_WeatherInf

Fig. 17: Instantiation of R-Interaction that is initiated by PersonalizedWeatherAssistant (screenshot of Protégé tool)



PA-Model:initiatesAction
PA-Model:initiatesInteraction
RoleInteraction:initiatesR_Interaction
PA-Model:isLocationOfUser
PA-Model:isPreConditionOf

Fig. 16: Role Interaction pattern specifies object property (*PA-Model*=prefix of namespace of AIS ontology) (screenshot of Protégé tool)

The super properties of the AIS Ontology are filled automatically without touching them explicitly when working with the pattern-specific relations. Within the proceeding formalization, the formalized patterns *Role creates IO* and *Service Interaction* are imported. This leads to further specifications of the super-properties of the AIS Ontology and thereby to an enhancement of the expressiveness without "getting lost". The following OWL snippet shows the representation of the interface service “PersonalizedWeatherService" modeled by using three Pre-Artifact patterns.

```
<Model:InterfaceService rdf:about="#PersonalizedWeatherService">
      <rdf:type rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
      <RoleCreatesIO:supportsCreation rdf:resource="#Creation_1"/>
      <RoleCreatesIO:interfaceServiceTakesRole
      rdf:resource="#PersonalizedWeatherAssistant"/>
      <RoleInteraction:supportsR_Interaction
      rdf:resource="#R_Interaction1"/>
      <ServiceInteraction:finalizesS_Interaction
      rdf:resource="#S_Interaction_3"/>
      <ServiceInteraction:finalizesS_Interaction
      rdf:resource="#S_Interaction_4"/>
</Model:InterfaceService>
```

The result of the formalization is an OWL description that represents the exemplary Pre-Artifact as well as the original narrative CM in a formal and computational way. According to the fact, that the steps of the formalization are on par with the steps of the definition of the diagrammatic CM, there should be no challenge to execute the formalization in an automatic way, e.g., with a graphical modeling tool based on semantic technology infrastructures. For the intelligent bathroom use case, we translated the three best-performing narrative CMs into Pre-Artifacts and afterwards into propositional CMs. So, the situational part of the knowledge representation consists of seven propositional CMs formalized in OWL.

5.2 Formalization of Contextual Part

In order to develop the ontology network described in Section 4.2 we applied the XD methodology (Presutti et al., 2009), as mentioned in Section 4. XD is a methodology specifically tailored towards rapid prototyping of highly modularized ontology networks, by means of ODP reuse. The methodology consists of seven activities as illustrated in Fig. 18. Although illustrated as a sequence in the figure, it is important to note that the sequence in the figure mainly illustrates the flow of information, i.e. arrows represent an input feed, and since some activities are performed by different groups they may very well be performed in parallel or continuously throughout the project. For instance, the integration activity (6) is performed continuously throughout the project, starting from when the first module is ready until the final result is delivered. In the meantime, the design pairs are in parallel producing modules solving specific sets of requirements (4-5). The following paragraphs introduce the methodology more in depth.

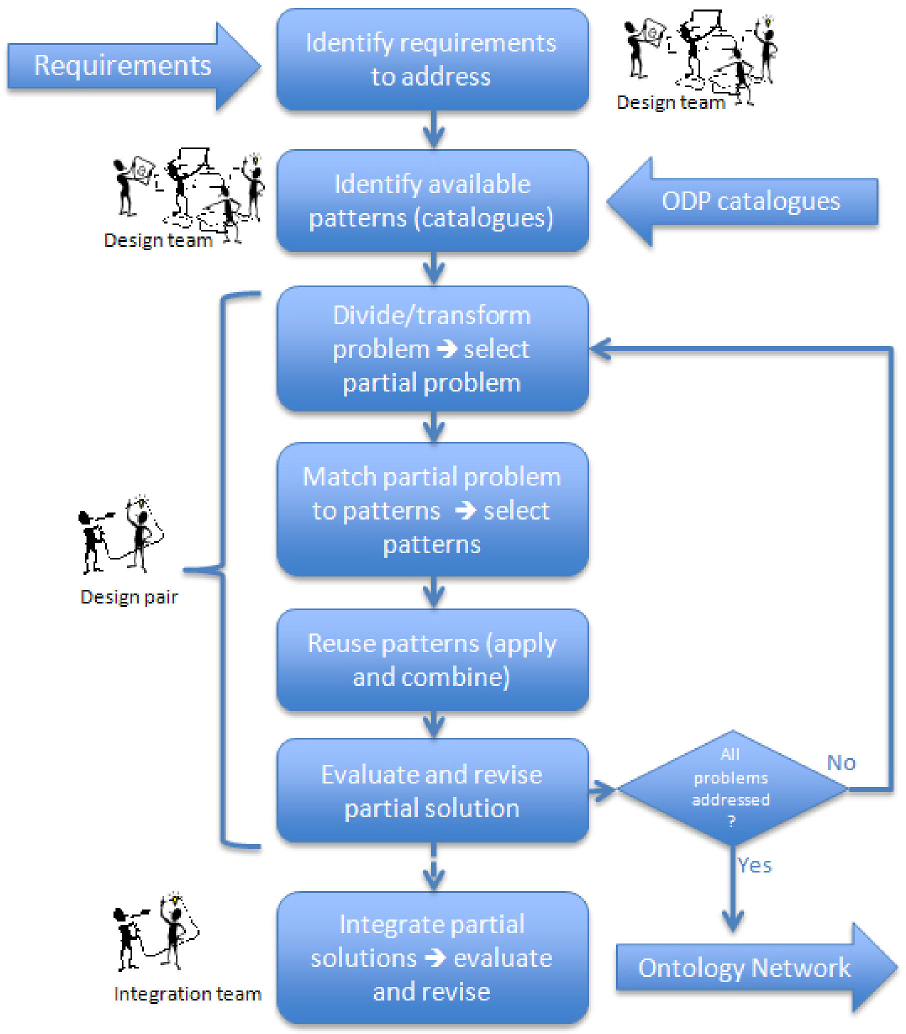

Requirements
Identify requirements to address
Design team
Design team
Identify available patterns (catalogues)
ODP catalogues
Divide/transform problem → select partial problem
Match partial problem to patterns → select patterns
Design pair
Reuse patterns (apply and combine)
Evaluate and revise partial solution
All problems addressed ?
No
Yes
Integration team
Integrate partial solutions → evaluate and revise
Ontology Network

Fig. 18: The XD methodology for CP reuse (Presutti et al., 2009)

The XD methodology for CP reuse (Presutti et al., 2009) provides an iterative method for incrementally building an ontology network from CPs. The first three activities concern the pro-

ject setup and requirements engineering, i.e. how the ontological requirements are collected from user stories. User stories are short scenarios expressed in natural language; describing the expected usage of the ontology network from a user perspective (a “user” can be either an end-user or a software system making use of ontology-based services). In our case these stories were created by breaking down and detailing the narratives used for developing the situation KR, so that the stories more clearly and concretely exemplify the information storage and usage needs. As explained previously, the Pre-Artifacts and the software requirements were also used to guide the requirements elicitation process.
The following two activities are ideally performed by pairs of ontology designers in parallel, who will iterate through these activities until they have ran out of requirements, i.e. solved the complete problem. However, in our particular case the development was done by one individual (periodically interacting with the developers of the situation representation), whereby the “pair design” principle of the methodology had to be relaxed.
The requirements are first divided into small coherent sets, i.e. grouping the requirements that concern the same concept or aspect, where each set is treated separately to derive one of the modules for the ontology network. Each module is then realized through finding appropriate CP modules on the Semantic Web, e.g. in repositories like the ODP Portal[14], specializing and extending them with classes, properties, and other axioms specific to the intelligent bathroom, and composing them to form a module solving that specific set of requirements. An important part of XD is ontology testing. Testing is performed on each individual module, before its release. Each type of requirement needs to be tested separately, e.g. the CQs are transformed into SPARQL queries and used together with test data to check retrieval capabilities, while contextual statements and reasoning requirements are tested through creating specific test data for allowing certain inferences or provoking inconsistencies.

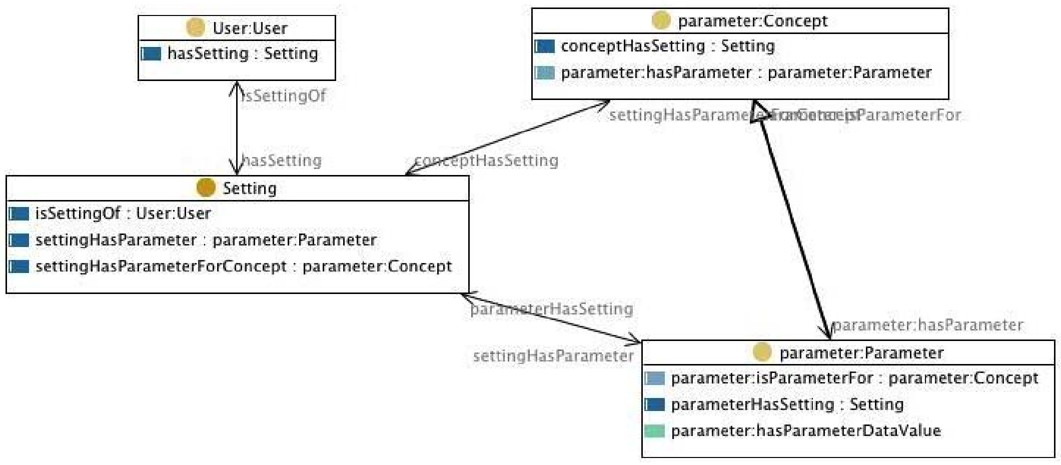

User:User
hasSetting : Setting
parameter:Concept
conceptHasSetting : Setting
parameter:hasParameter : parameter:Parameter
isSettingOf
hasSetting
conceptHasSetting
settingHasParameterForConcept
parameter:isParameterFor
Setting
isSettingOf : User:User
settingHasParameter : parameter:Parameter
settingHasParameterForConcept : parameter:Concept
parameterHasSetting
settingHasParameter
parameter:hasParameter
parameter:Parameter
parameter:isParameterFor : parameter:Concept
parameterHasSetting : Setting
parameter:hasParameterDataValue

Fig. 19: Illustration of the main parts of the *UserSettings* module. [15]

Finally, the last two activities are concerned with the continuous integration of modules, and subsequent releases of new versions of the resulting ontology network. Preferably, there is a dedicated integration team (or pair) for performing this task; however, in our case the developer performed the integration. Integration may be challenging, and require both alignments between modules to be added, or even some refactoring of previous modules. For each new module that is added, tests are run to confirm its correct integration. By applying XD the resulting ontology network becomes highly modular in its architecture, which both supports future changes and updates, as well as to support efficient handling of the KB when it is

[14] http://www.ontologydesignpatterns.org
[15] The diagram has been generated through the UML-notation used in TopBraid Composer

aligned to the situation representation, i.e. for a certain situation usually only a subset of all the modules need to be loaded and reasoned upon.

XD specifies reuse as one of the main objectives; this can be both reuse of CPs as well as reuse of other existing Semantic Web resources. A general principle of the Semantic Web, and in particular the Linked Data community, is to reuse existing vocabularies as far as possible. The benefits of reusing existing ontologies have to be weighted against the overhead that possibly irrelevant parts of the reused components will add. In the first version of the context ontology network we have focused on the reuse of small modules representing CPs from the ODP Portal, in order to minimize the overhead for the first demo system. In later versions, we will broaden the reuse to common Semantic Web ontologies, and some alignments will be added, or modules may even be replaced.
As examples of the modules constituting the context ontology network, i.e. the contextual KR expressed in OWL, we describe three of them in detail. First, the *UserSettings* module, modelling the system settings of particular users, as illustrated in Fig. 19. This module specializes a general CP called *Parameter*, which expresses the parameterization of some concept[16]. The module also relies on another, more general, module, i.e. the *User* module, which defines the notion of a user of the intelligent bathroom. In this module the class *Setting* is defined, where a setting can apply to some concept, e.g. the notion of music listening, and hold a certain value for each parameter of that concept for a certain user, e.g. a certain preferred volume level for a certain user when listening to music. This module realizes CQs such as: “What is the value of this parameter, concerning a certain concept, for a particular user?”

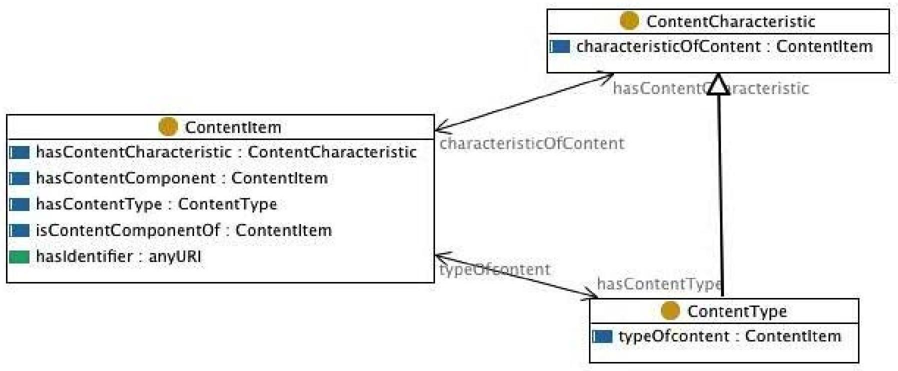

ContentCharacteristic
characteristicOfContent : ContentItem
hasContentCharacteristic
ContentItem
hasContentCharacteristic : ContentCharacteristic
hasContentComponent : ContentItem
hasContentType : ContentType
isContentComponentOf : ContentItem
hasIdentifier : anyURI
characteristicOfContent
typeOfcontent
hasContentType
ContentType
typeOfcontent : ContentItem

Fig. 20: Illustration of the main parts of the *Content* module.

Second, there is the *Content* module, modelling content items of a CMS, their relations to each other (e.g. containment – one content item being contained within another), and their characteristics. A graphical illustration of the module can be seen in Fig. 19 (some elements have been omitted for readability). This very simple module defines the classes *ContentItem* and *ContentCharacteristic*, where a particular content characteristic is the type of the content, which can contain any application specific types that apply for this particular application. Content items are related to each other, through containment, i.e. one content item can be contained within another one. Content items are related to characteristics through an object property (and its corresponding inverse), which in turn has a sub-property for expressing the type of the content item. Since we are operating in a Semantic Web context, we can also assume that every content item can be uniquely identified through a URI, which is modelled through the *hasIdentifier* datatype property. This very general model of content items, is then

[16] http://ontologydesignpatterns.org/wiki/Submissions:Parameter

further specialized in several other modules, e.g. for expressing user preferences for certain types of content, and to represent more detailed characteristics of content.
The *UserSettings* module shows a part of the context KR that can store actual sensor data about the bathroom, e.g. what is the current volume setting for a specific user, or together with other modules store the preferred settings for different situations. There we are dealing with concrete information about the bathroom environment. The second example, on the other hand, deals very concretely with representing content from the web, to be delivered to a user of the bathroom. The third example, that we are about to describe, is a more complex but also more concrete and application-specific module, called *WeatherForecast*. As can be seen in the (partial) illustration in Fig. 20, this module relies on the more general *Content* module, to describe a specific type of content, i.e. weather forecasts. The module also relies on a general CP, which is called Information Realization[17]. That CP models the distinction between abstract information and its realization as concrete objects, e.g. files or pages. This distinction is used to separate a *WeatherForecast*, as a *ContentItem*, i.e. a concrete *InformationRealization* as an HTML file or piece of text, from the information that the weather forecast contains, i.e. *WeatherInformation* in terms of the *WeatherType* that is valid for a certain *Place* during a certain *TimeInterval*. The module is used for describing external content, i.e. weather forecasts retrieved through web services, and to semantically annotate some parts of their content, i.e. the weather type, place, and time of the forecast.

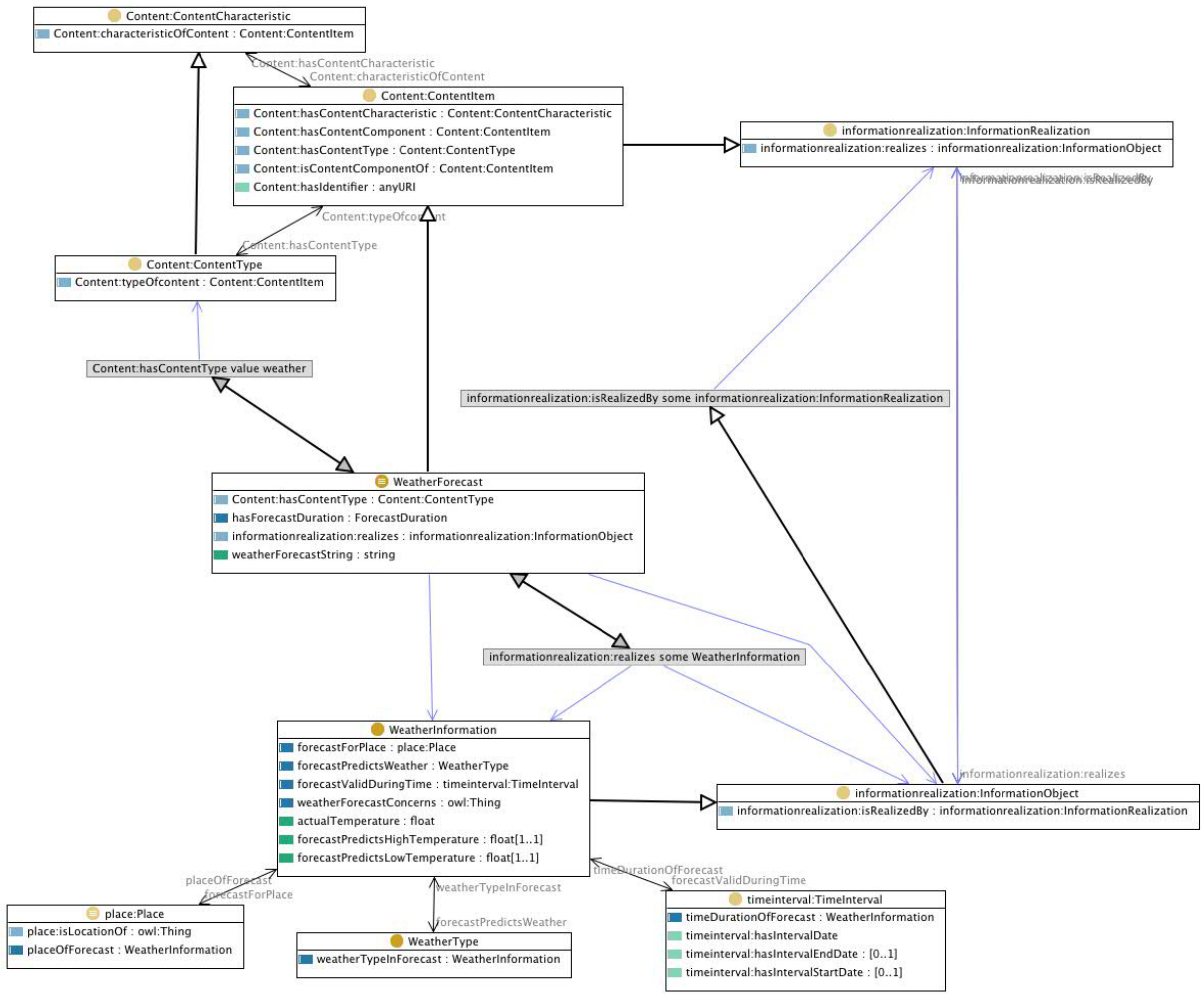

Content:ContentCharacteristic
Content:characteristicOfContent : Content:ContentItem
Content:hasContentCharacteristic
Content:characteristicOfContent
Content:ContentItem
Content:hasContentCharacteristic : Content:ContentCharacteristic
Content:hasContentComponent : Content:ContentItem
Content:hasContentType : Content:ContentType
Content:isContentComponentOf : Content:ContentItem
Content:hasIdentifier : anyURI
informationrealization:InformationRealization
informationrealization:realizes : informationrealization:InformationObject
informationrealization:isRealizedBy
Content:typeOfcontent
Content:hasContentType
Content:ContentType
Content:typeOfcontent : Content:ContentItem
Content:hasContentType value weather
informationrealization:isRealizedBy some informationrealization:InformationRealization
WeatherForecast
Content:hasContentType : Content:ContentType
hasForecastDuration : ForecastDuration
informationrealization:realizes : informationrealization:InformationObject
weatherForecastString : string
informationrealization:realizes some WeatherInformation
WeatherInformation
forecastForPlace : place:Place
forecastPredictsWeather : WeatherType
forecastValidDuringTime : timeinterval:TimeInterval
weatherForecastConcerns : owl:Thing
actualTemperature : float
forecastPredictsHighTemperature : float[1..1]
forecastPredictsLowTemperature : float[1..1]
informationrealization:realizes
informationrealization:InformationObject
informationrealization:isRealizedBy : informationrealization:InformationRealization
placeOfForecast
forecastForPlace
weatherTypeInForecast
forecastPredictsWeather
timeDurationOfForecast
forecastValidDuringTime
place:Place
place:isLocationOf : owl:Thing
placeOfForecast : WeatherInformation
WeatherType
weatherTypeInForecast : WeatherInformation
timeinterval:TimeInterval
timeDurationOfForecast : WeatherInformation
timeinterval:hasIntervalDate
timeinterval:hasIntervalEndDate : [0..1]
timeinterval:hasIntervalStartDate : [0..1]

Fig. 21: A (partial) view of the *WeatherForecast* module.

[17] http://ontologydesignpatterns.org/wiki/Submissions:Information_realization

5.3 Concept of Linkage of Situational and Contextual Part

Both parts of the knowledge representation, i.e., contextual and situational part (cf. Fig. 4), are linked dynamically based on rules (cf. Fig. 22). Form a conceptual point of view, the situational part consists of the *propositional CMs* and a representation of the *Current Situation* in the bathroom also framed by the AIS Ontology. Each propositional CM consists of the aforementioned ontological structures classifying the identified situations and a related rule set. The *Current Situation* represents the real situation in the physical bathroom. It is updated according to events taken place in the environment, e.g., a user enters the room. By permanently aligning the *Current Situation* with the existing propositional CMs, the UIS is able to determine which propositional CM represents an appropriate solution to handle the events taken place in the bathroom. The process of determining the appropriate propositional CM will be elaborated in the following sections. The contextual part consisting of the context ontology network covering ODPs is served by IKS enabled CMS(s) and external content services that are requested when required by specific situations, e.g., weather or event services. A detailed description of the relevant IKS modules and their usage will be given in Section 6 and 7. When processing a propositional CM, specific Content ODPs of the context ontology network are triggered by rules delivered by the propositional CM, e.g., to request the location of the system. The results that mean references to specific concepts of Content ODPs are then stored within the *Current Situation* instance. Hence, the *Current Situation* instance grows and is adjusted to the propositional CM that is currently processed. That means the UIS reacts according to the situational structure that is growing within the *Current Situation* instance and tries to fulfill the situation that is pretended by the propositional CM. A detailed, technical description of the linkage process will be part of Section 6 and 7.

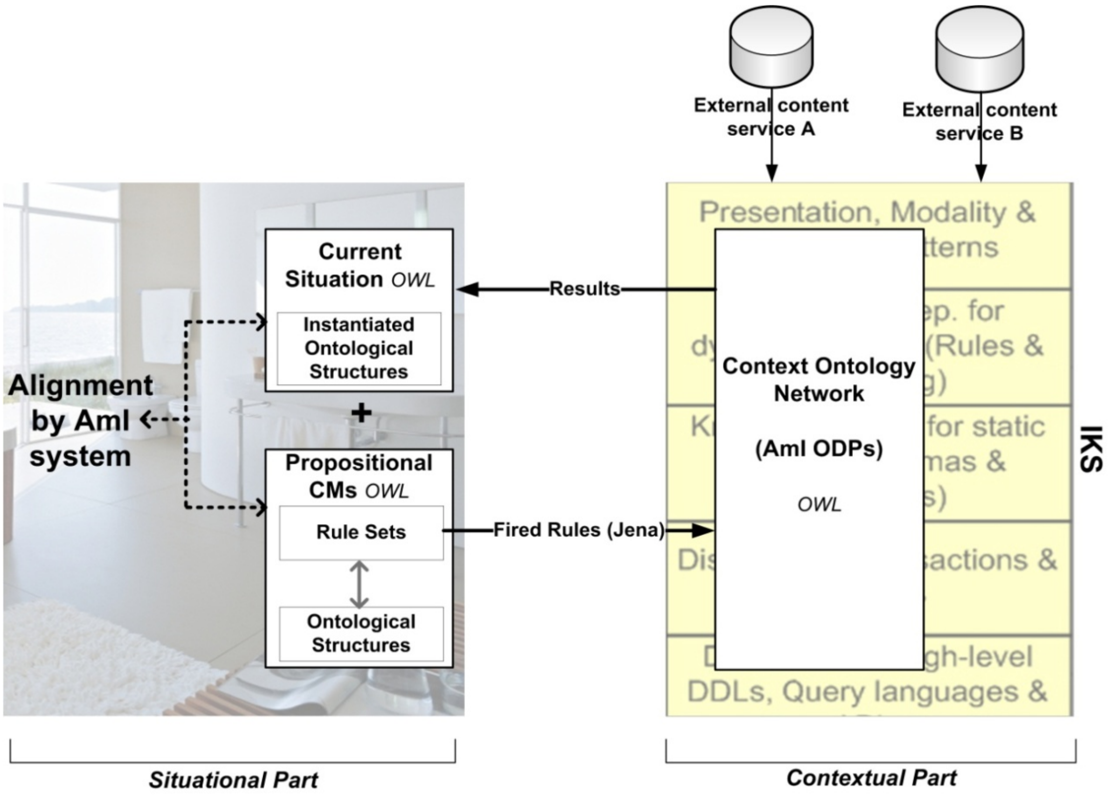


Fig. 22: Concept of linkage of contextual and situational part of the knowledge representation

6 Design of System Architecture for the AmI Use Case

In the following sub sections, we will present the development of a logical architecture for the AmI case based on IKS Reference Architecture. Note that caused by the later finalization of IKS beta components, the team of T4.1 decided to develop the logical architecture before the finalization of the IKS beta. Within the re-iteration of the AmI system, the finalized IKS components were integrated later (cf. Section 10). Fig. 23 shows the high-level logical architecture for the AmI Case system that consists of 13 modules on 5 logical architecture layers based on the IKS reference architecture (Christ & Nagel, 2011).

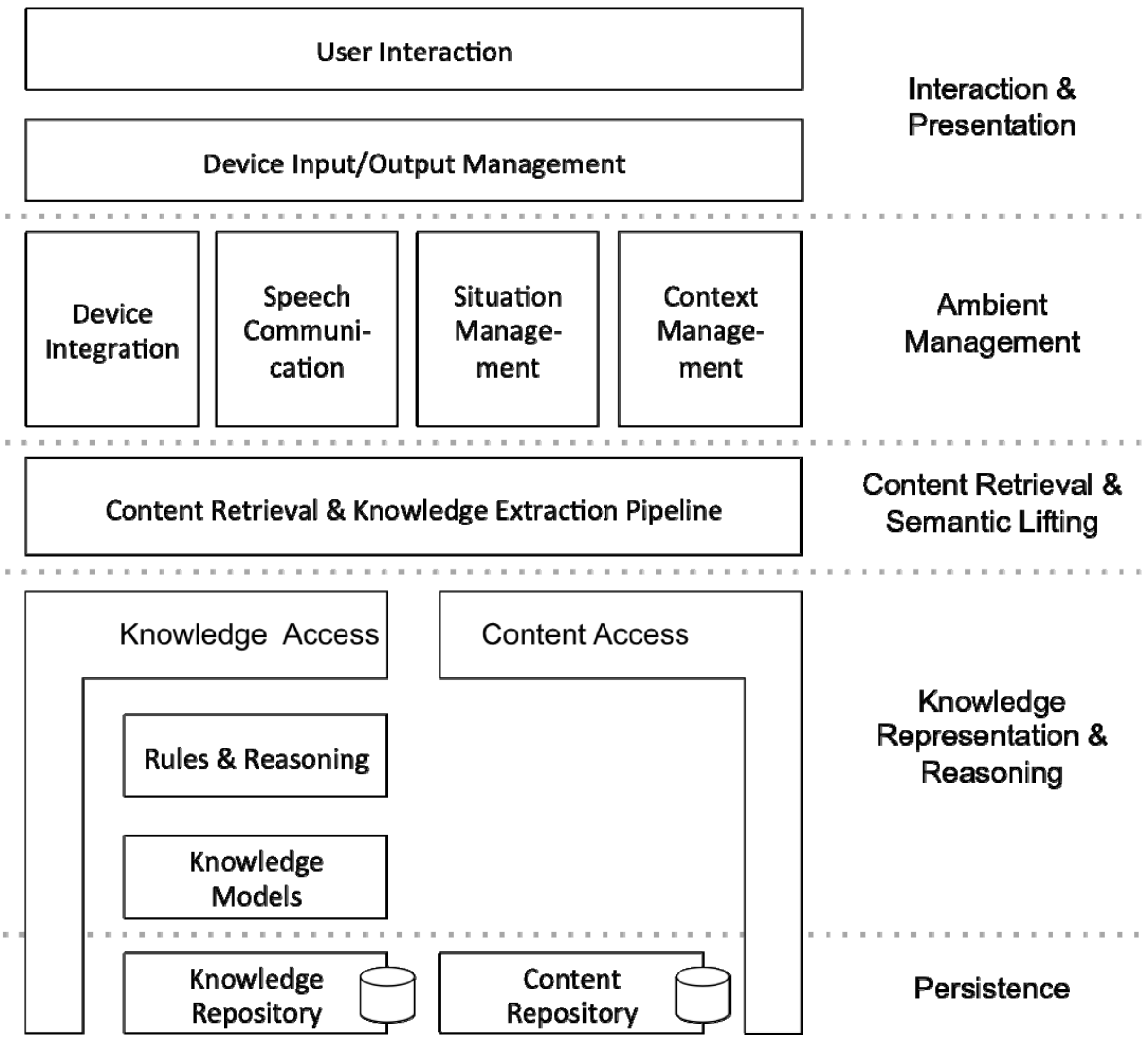


Fig. 23: Logical Architecture of AmI System

In the next sub sections we will introduce the following modules: *Device Input/Output Management, Device Integration, Speech Communication, Situation Management, Context Management* and the *Content Retrieval & Knowledge Extraction Pipeline*., i.e. all modules on the Content Retrieval & Semantic Lifting layer and above, because these mainly lie in the focus of the AmI case. The other components will be addressed in the detailed elaboration of the architecture and the system workflows (cf. Section 6.7 and 6.8).

6.1 Device Input/Output Management

The *Device Input/Output Management* manages all technical operations regarding devices inside the AmI Case bathroom environment after these have been integrated into the technical infrastructure (cf. 6.2). On the one hand the module cares for the playback of different forms of contents (e.g. audio, video, images, speech) on appropriate devices that were evaluated and selected by the Context module. On the other hand it cares for the the interpretation of sensor inputs (e.g. distance or person recognition sensors) as well. In the following, the functional behaviour of the module as well as relevant data structures and its input-output behaviour are elaborated.

Functional Behaviour: The functional behaviour of the Device Input/Output Management module covers two main aspects: (1) device orchestration and (2) device feedback. After the device detection and integration process is performed by the Device Integration module, the Device Input/Output Management module is responsible for the management of the device. This management encompasses the following steps: After the Context Management module has determined the best device to present a specific content in the current context (device selection) the Device Input/Output Management module presents the content on this selected device (content presentation).

Data structures: The Device Input/Output Management module works on the device representation (stored within AmI ODPs) that contains descriptions and specifications of the device itself and its services/functionalities. Furthermore, the module processes device service libraries containing standardized interfaces and software libraries to communicate with the devices. To broadcast interaction events between modules an interaction communication protocol is used.

Input-Output Behaviour: The Device Input/Output Management module provides a subscription service that allows other modules to get informed about interaction events in the bathroom. Furthermore, the module redirects more complex input data (e.g. voice input) to the appropriate interpretation module (e.g. the Speech Communication module) for further interpretation. On the other hand, the Device Input/Output Management module offers an interface to commit content objects for presentation on the device that was selected by the Context module.

6.2 Device Integration

The *Device Integration module* manages all technical operations regarding devices inside the AmI Case bathroom environment, which are necessary that the devices can be used by the Device Input/Output Management module (cf. 6.1). This includes on the one hand the discovery of in- and output devices inside the environment as well as on the other hand their integration in the technical infrastructure. In the following, the functional behaviour of the module as well as relevant data structures and its input-output behaviour are elaborated.

Functional Behaviour: The functional behaviour of the Device Integration module covers two main aspects: (1) device detection and (2) device integration. The Device Integration module has to detect, which devices are available within the bathroom (device detection). Next, the module has to integrate devices into the AmI environment based on software parts that are necessary to communicate with these devices (device integration). Furthermore, services or functionalities offered by the devices have to be detected. When a device has been removed from the environment, the module also has to care for the reverse process of the device removal.

Data structures: The Device Integration module works on the device representation (stored within AmI ODPs) that contains descriptions and specifications of the device itself and its services/functionalities. Furthermore, the module processes device service libraries containing standardized interfaces and software libraries to communicate with the devices. To broadcast interaction events between modules an interaction communication protocol is used.

Input-Output Behaviour: The Device Integration module autonomously works on the device representation and does not offer any external interfaces.

6.3 Context Management

The context inside the bathroom environment is managed inside the *Context Management* module. This encompasses the management of the current physical situation in the bathroom including the devices and users that are currently available. The current context, devices and user representations are therefore managed inside the knowledge repository using the modules on the knowledge representation & reasoning and persistence layer (cf. contextual part of knowledge representation). Regarding the devices, the Context Management module is also responsible for the determination of the most appropriate presentation device for presenting content objects of a specific form. The Context Management module represents a bridge between the physical devices handled by the Device Input/Output Management and the situational knowledge managed by the Situation Management module. In the following, the functional behaviour of the Context Management module as well as relevant data structures and its input-output behaviour are elaborated.

Functional Behaviour: The functional behaviour of the Context Management module covers two main aspects: (1) context-based device section for playback and (2) detection and communication of context changes inside the contextual part of the knowledge representation.
Within the context-based device selection, the Context Management module has to identify the most appropriate presentation device to present a content object of specific form in the current physical situation. Therefore, the Context Management module has to detect, which devices are available in the current context. Second, services or functionalities offered by the devices have to be detected. Last, the Context Management module selects the best presentation device to present the specific content in the current context. A further main task of the Context Management module is the listening on modifications of specific parts of the contextual part of the knowledge representation (current context within bathroom), e.g. devices are broken. When such modifications are detected, the Context Management module informs the Situation Management module that the situational part of the knowledge representation (cf. current situation in bathroom) has to be adjusted according the modifications.

Data structures: The Context Management module works on the contextual part of the knowledge representation.

Input-Output Behaviour: The Context Management module continuously checks for context changes, which require an adaption of the current situation. If such a change occurs it informs the Situation Management module about these changes. Such changes can activate adjustments of the current situation within the Situation Management module. On the other hand, the Context Management module reacts on interaction messages that will be sent by the Situation Management and Speech Communication module. The latter has the possibility to suggest a device for presentation of a specific content object. This suggestion will be used as impact for context-based device selection. All data modifications are communicated to the Knowledge Repository via the context-specific part of the Knowledge Access module.

6.4 Situation Management

The counterpart of the Context Management module is the Situation Management module that manages the situational part of the knowledge representation that describes all situations the system can react on. This part is managed inside the Knowledge Repository as situational knowledge part. The Situation Management module can determine situations stored within the Knowledge Repository as propositional conceptual models (propositional CMs) that fit to the current context in the bathroom. Furthermore, it identifies continuative situations as well as next appropriate system activities. This leads to fulfilment of the identified situation. The results of processing of situations are sent to the Context Management module that adjusts the current context in the bathroom, e.g., switching on the light. In the following, the functional behaviour of the Situation Management module as well as relevant data structures and its input-output behaviour are elaborated.

Functional Behaviour: The functional behaviour of the Situation Management module covers five main aspects: (1) identification of relevant propositional CM, (2) processing of propositional CM, (3) alignment of propositional CM and current situation, (4) generation of propositional CMs and (5) management of situation adjustment resulting from interaction messages. In general, the Situation Management module has to manage the propositional CM and current situation representations, i.e., reading, modifying, comparing, saving, and creating. Concerning the first aspect – the identification of the relevant propositional CM, the Situation Management module has to match the current bathroom situation with the available propositional CM representations by means of the context module. Afterwards, it has to select an appropriate propositional CM. In case that no appropriate propositional CM can be determined, determination methods like fuzzy search are used to identify propositional CMs that are similar to the current bathroom situation or later on similar to the current situation as "copy" of the bathroom situation. Within the alignment of propositional CMs and Instances, the Situation Management module compares both structures and tries to fill missing concepts in the current situation. There are two opportunities for filling up the current situation: (1) filling of (empty) current situation based on interaction messages, e.g., Anna is recognized in the bathroom; an instance of UserRole will be created within the current situation, or (2) execution of system activities presented in propositional CM that lead to integration of adequate concepts in the current situation. In contrast, the latter opportunity shows that the execution of system activities or steps described in the propositional CM leads also to a fulfilment of the propositional CM, because the current situation (and therefore the real bathroom) is adjusted accordingly. Certainly, the results of these processes are communicated to the Context Management module.
When a propositional CM was successfully fulfilled, the Situation Management module proceeds with the subsequent propositional CM if available. In case, the Situation Management module is not able to identify an appropriate or similar PA, a new propositional CM is created. This also takes place, if parts of the real bathroom situation differ from the selected propositional CM. Then, a new propositional CM is generated based on suggestions by Speech Communication module regarding the discourse history etc., for instance Anna wants to hear music instead of getting the weather information.

Data Structures: The Situation Management module works on situational part of the knowledge models. This part can be divided into propositional CMs and the current situation. Propositional CMs represent all AmI bathroom situations that can be handled by the system. During run-time the system creates new propositional CMs according to events taking place in the bathroom. The real current situation in the bathroom is represented by an instantiation of propositional CMs. It bases on propositional CMs and is filled regarding the current context of the physical environment, e.g., UserRole is instantiated by Anna.

Input-Output Behaviour: The Situation Management module communicates with the Context Management module to identify appropriate propositional CMs according the current bathroom situation. All data modifications are communicated to the Knowledge Repository via the Situation Management module-specific part of the Knowledge Access module.

6.5 Speech Communication

The *Speech Communication* module cares for the management of the communication between the system and the user. Therefore it manages the NLP communication and discourse in communication with the Device Input/Output Management module. The communication knowledge is stored inside the Knowledge Repository and can be managed using the Knowledge Access module. In the following, the functional behaviour of the Speech Communication module as well as relevant data structures and its input-output behaviour are elaborated.

Functional Behaviour: The Speech Communication module is responsible for the management of the speech-based direct interaction with the user. This management is split in three main tasks:

- Speech Input Interpretation
- Discourse management
- Dialog management

The task of the speech input interpretation step is the determination of so called “user intentions” (semantic result of interpretation) to the interpretation “hypothesis” resulting from each recognition process. In the second step, the discourse management checks current user intention interpretation against the discourse context (discourse interpretation). During this process, interpretations are compared with the discourse history in order to enrich interpretation by resolving typical discourse phenomena (e.a. ellipses resolution, reference resolution, etc.) and disambiguate interpretations if needed and possible. This happens by filling empty slots values with possible candidates from the discourse history or substituting slot values with more plausible ones. The dialog management task consists in determining next steps in interaction with the user by: managing turn taking, planning reaction, retrieving needed knowledge from system. Finally, after having interpreted user intentions and retrieved the needed requested information the presentation has to be performed. This last step is orchestrated in the Device Input/Output Management module.

Data structures:

- User Intentions – a set of possible user intentions to be used as results of interpretations
- Domain Knowledge – access to the given domain knowledge sources has to be given in order to determine user intentions and pertinence to give context
- Language Models (e.g., Language Grammars)

Input-Output Behaviour:
The Speech Communication module receives speech input from the Device Input/Output Management module, retrieves needed content from the Knowledge Repository and passes results to the Context Management module in form of presentation recommendations. All data modifications are communicated to the Knowledge Repository via the Knowledge Access module.

6.6 Content Retrieval & Knowledge Interaction Pipeline

The *Content Retrieval & Knowledge Interaction Pipeline* module retrieves content objects from external (unstructured) sources (e.g. proprietary websites, IKS-capable systems) and prepares them for the usage inside the AmI case environment. In the following, the functional behaviour of the module as well as relevant data structures and its input-output behaviour are elaborated.

Functional Behaviour / Input-Output Behaviour: The Content Retrieval & Knowledge Interaction Pipeline module of the AmI Case is responsible for the integration of the external contents with the AmI case environment. The module performs four main tasks:

- Content aggregation
- Content reengineering
- Content refactoring
- Content filtering

Content aggregation is the task, which includes identifying and registering external services. How the module communicates with the external services is described in this task. Furthermore retrieval of external content through defined protocols is also one of the main functionalities of this task. The output of the content aggregation step is the retrieved contents from external sources. The retrieved content may appear in various formats. The aim of content reengineering step is to transform the retrieved content into an ontological representation. This task can also be called syntax alignment. In other words, content retrieved from an external source may include semantic information in different formats. Contents retrieved in this task, are transformed to a common ontological representation. Once external content is transformed to a common ontological representation, there still needs an alignment to map external content concepts to AmI concepts from the AmI ODPs. Content refactoring task is responsible for this semantic alignment. Fig. 24 depicts the logical architecture details of the Content Retrieval & Knowledge Interaction Pipeline module. Note that the content aggregator may directly send the retrieved content to the content refactorer when the external service already produces content in the desired ontological representation. Finally, content refactorer makes the transformed content items available in the AmI Case environment by storing them into the Knowledge Repository as external knowledge using the context-specific interface of the Knowledge Access module. The binary parts of some content items (e.g. video or audio content) are stored in the Content Repository using the Content Access module.
Data structures: Content Retrieval & Knowledge Interaction Pipeline module will be able to retrieve content in different formats. In order to integrate the retrieved content there is a need for a common representation. Resource Description Framework (RDF) provides rich framework information. Hence, the Content Retrieval & Knowledge Interaction Pipeline module aims to transform non-RDF content to RDF. Another data that needs to be described and represented in the scope of the module is the transformation rules. Rules need to be defined to describe transformations between non-RDF and RDF content. Furthermore, extracted RDF need to be aligned with the ontology of the AmI Case. Therefore, rules are also needed to RDF-to-RDF conversions.

Input-Output Behaviour: Triggering content extraction can be performed through the following options:

- If the external service is a push service, then content extractor processes the received content and feeds the Knowledge Repository.
- If the external service is a pull service (such as web service): (a) The Content Retrieval & Knowledge Interaction Pipeline module enables defining time intervals to retrieve content from the external systems periodically, or (b) when Knowledge Access

does not find the requested data from the Knowledge Repository, it triggers content extractor and waits for a notification. Then content extractor retrieves the content from the external service and feeds the Knowledge Repository.

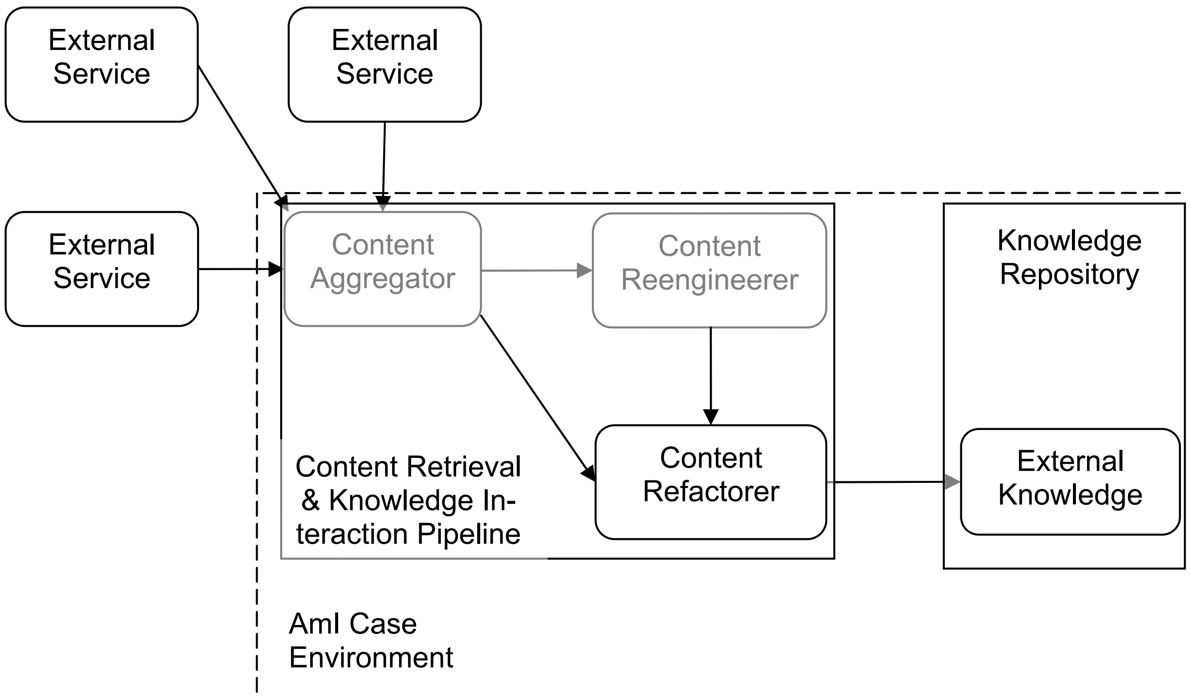


Fig. 24: Details of Content Retrieval & Knowledge Interaction Pipeline Module

6.7 Elaboration of Logical Architecture

Based on the aforementioned high-level logical architecture of AmI system, we further elaborated the single modules. Fig. 25 shows the elaborated logical architecture for the AmI use case. The modules were split into several components that intercommunicate directly or via broadcast messages. The modules, their components as well as data flows and dependencies between these components are described in the following sub sections.

6.7.1 Components of Device Input/Output Management

Device Input Interpretation Component. The *Device Input Interpretation Component* cares for the interpretation of inputs sent from devices within the bathroom environment. After interpretation, the result is broadcasted to all other major modules. The functionality of the Device Input Interpretation Component encompasses three major tasks: (1) the component has to care for the interpretation of device inputs; especially from activity recognition devices; (2) the component has to broadcast the interpreted device events to other modules. In case that the input of a device is too complex (e.g., audio streams), (3) the Device Input Interpretation Component has to redirect these inputs to the Context Management Module for the evaluation of inputs. Furthermore, the component provides an interface to enable devices to communicate sensor events.

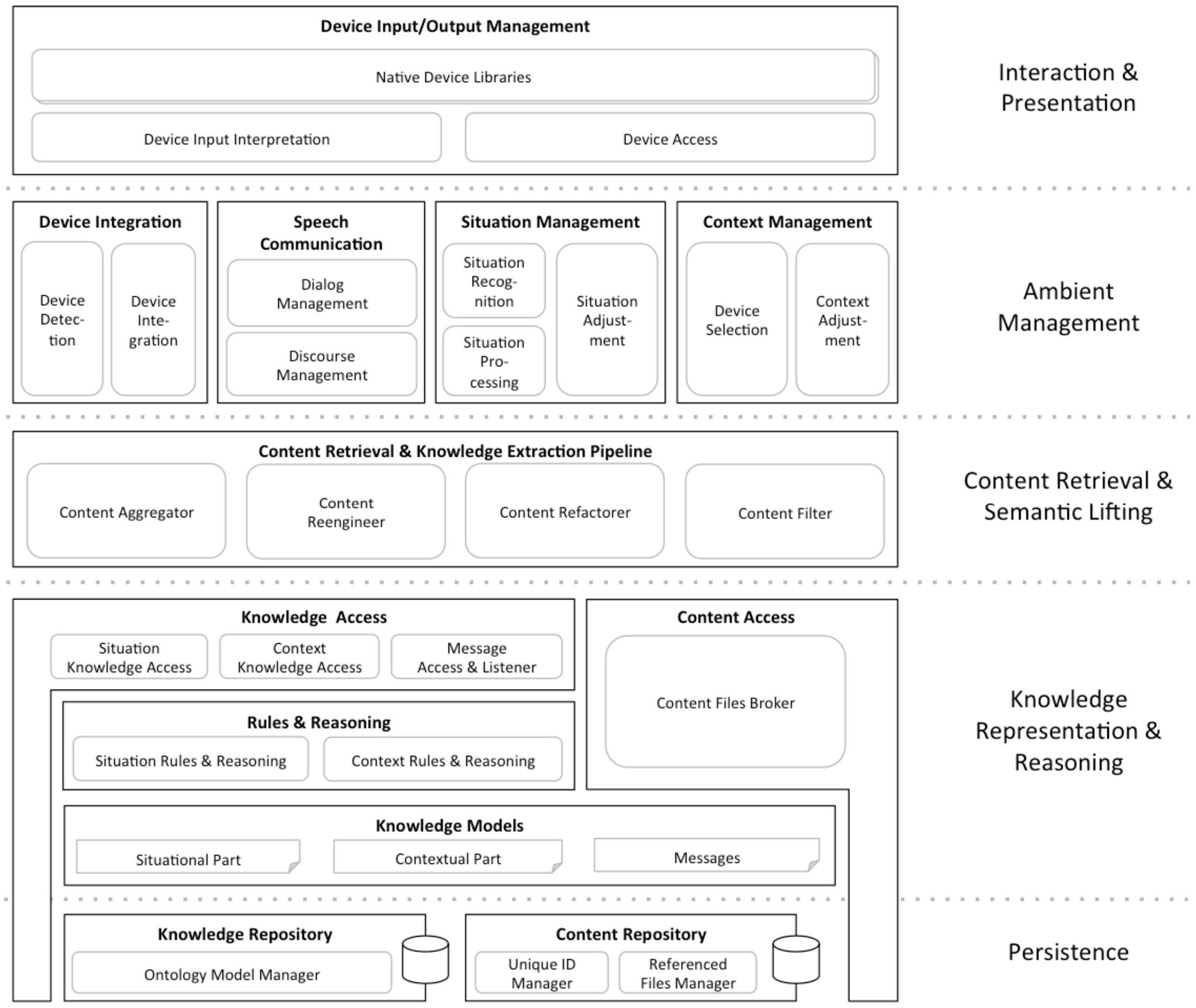


Fig. 25: Component-specific elaboration of logical architecture of AmI system

Device Access Component. The *Device Access Component* is responsible for the status management of devices within the bathroom environment. Furthermore, it provides the possibility to send content items to specific devices for presentation. The main task of this component is the orchestration of the device status *busy* to avoid a simultaneous presentation of multiple contents on one device. Furthermore, the component cares for the presentation of content items on devices that are currently integrated in the environment (device has the status *available*). To realize this, the module provides an interface to present content items on a specific preselected device. The Device Access Component has to adjust the device representations in AmI ODPs via the appropriate Context Knowledge Access Component within the Knowledge Access module. This mainly affects the management of status attributes of the current device. After a successful content presentation the Device Access Component informs the Device Input Interpretation Component, which will broadcast this information as an event.

6.7.2 Components of Device Integration

Device Detection Component. The Device Detection Component is responsible for the detection of devices within the bathroom environment (equivalent to *system initiated identification*). Furthermore, it continuously checks whether a device is already integrated into the environment. To complete the device detection cycle the Device Detection Component also detects if devices left the bathroom environment. The component further provides an interface to explicitly inform the Device Integration Component about devices that are currently available in the bathroom (equivalent to *device initiated identification*).

Device Integration Component. The second component is the Device Integration Component that is responsible for the integration of detected devices into the technical environment. One of the major functionalities of the Device Integration Component is the retrieval of the semantic representation of the device that has to be integrated. Afterwards, this semantic device representation is integrated into the Knowledge Repository. In case of removal of a device from the bathroom, this representation has to be removed from the Knowledge Repository as well. The second major functionality encompasses the aforementioned steps regarding the OSGi based native device library. The component cares for the retrieval of the OSGi based native device library, the integration of this library into the technical I/O environment as well as the removal of the same when the device is removed from the bathroom.
Regarding the integration of devices from a knowledge management perspective, the adaption of AmI ODPs in the Knowledge Repository is necessary. Each device that is currently available in the bathroom environment has to be represented by an instance of the semantic device representation integrated in the Knowledge Repository.

6.7.3 Components of Context Management

Device Selection Component. The Device Selection Component cares for the selection of appropriate presentation devices for the presentation of specific content items. Informed about all devices currently available in the bathroom by looking into the contextual part of the knowledge model, the component is able to filter these devices regarding those that fit to the current content item (e.g., type of content item is supported by the output device) as well as the current physical context (e.g., privacy of a device). The survey of references on semantic device descriptions is retrieved from the Situation Processing Component of Situation Management module that decided to present a specific content item. After evaluating a presentation device the result and the content item is communicated to the Device Access Component of the Device Input/Output Management module for presentation.

Context Adjustment Component. The Context Adjustment Component retrieves events broadcasted by the Device Input Interpretation Component of Device Input/Output Management module. This leads to context adjustments (e.g., user movement). If such an event is received by the component the context representation is updated in the Knowledge Repository using the Context Knowledge Access Component of Knowledge Access module. In case this event led to a context change, this change is communicated to the Situation Adjustment Component of Situation Management module for further adjustments of the situation. In parallel, the Context Adjustment Component cares for management of the contextual part (AmI ODPs) in the *Knowledge Repository*. This encompasses for instance the occurrence of a new content item that could be needed in the current situation.

6.7.4 Components of Situation Management

Situation Processing Component. The Situation Processing Component continuously checks whether the current situation can be fulfilled based on the appropriate situation (propositional CM) by a pro-active system action. This is possible if all requirements are fulfilled that are necessary to perform the next step within the current situation. The action that will be performed by communicating the decision to the Device Selection Component of Context Management module is the only step that is left over to fulfil the whole situation description of the current situation. An exemplary situation in this case could be if the user is at the correct position in the bathroom, there are appropriate presentation devices available for the content item, the devices are available for presentation as well and the component determines that the presentation of the content item would fulfil the situation. In this case communicating the decision to present the content item to the Communication Module will perform the action.

Situation Recognition Component. The Situation Recognition Component continuously compares the current situation state with all available propositional CMs to determine if a change of situation type should be performed. In case the change was conducted this decision is broadcasted so that all components can adapt on the modification.

Situation Adjustment Component. The analysis of the current context regarding changes of the current situation representation that means additions, modifications and removals of objects and relations within the situation representation is processed in the Situation Adjustment Component. This evaluation is triggered by general context changes that are communicated by the Context Adjustment Component of Context Management module to the Situation Adjustment Component. If a relevant change is evaluated, the current situation and the physical bathroom situation are adapted to the change recognized in the Context Adjustment Component. This could be for instance a user entering the bathroom or a content item required in the current situation that becomes available in the contextual part of the knowledge representation.

6.7.5 Components of Speech Communication

Discourse Management Component. The Discourse Management Component is responsible for the interpretation of the context of discourse (i.e., information coming from already loaded interaction). The component receives an interpretation from the *native device libraries* of Device Input/Output Management module and the speech recogniser which already performed a speech interpretation and checks it against the discourse history in order to enrich and/or disambiguate interpretation. Once the discourse management task has been performed, the interpretation of user intention is passed to the Dialog Management Component.

Dialog Management Component. The Dialog Management Component is responsible for the management of dialogical interaction. After the interpretation of user intentions has been passed from the Discourse Management Component, the dialog management component decides on the next steps needed to achieve user requests or perform user commands. If the intention is clearly specified, the component broadcasts the command or the content item requested by the user to the Situation Processing Component of Situation Management module or in some simple cases directly to the Device Access Component of Device Input/Output Management module.

6.7.6 Components of Content Retrieval & Knowledge Extraction Pipeline

Content Aggregator Component. The Content Aggregator Component is responsible for registering and accessing external services. These external services can be any services including web services, content repository services or content management systems from which information can be retrieved using various protocols. The output of content aggregation is the retrieved contents from external sources. The retrieved content may appear in various formats.

Content Reengineer Component. After content is retrieved from external sources, an alignment with the contextual part of the knowledge representation (AmI ODPs) is needed. In order to achieve such an alignment, the first step is to transform the retrieved content into a common ontological representation. The Content Reengineerer Component is responsible for this transformation process. Since RDF[18] has become the de facto method for conceptual description or modelling of information, it is used for representing knowledge in the AmI Case system. It should be noted that if the retrieved content is already in RDF format, then there is no need for a reengineering step.

Content Refactorer Component. The second step in the alignment process is performed by the Content Refactorer Component. Once external content is transformed to a common ontological representation, there is still a need for alignment to map external content concepts to AmI ODP concepts. To perform such a mapping, the content refactorer requires access to the contextual part of the knowledge representation using the Context Knowledge Access Component of Knowledge Access Module. For example, if weather information is represented through the weather ontology[19]. Any external content representing weather information should be aligned with the weather information concepts provided by the AmI ODPs. The refactoring is performed by means of refactoring patterns. A refactoring pattern is a description of an alignment between two ontologies that can be expressed in RDF (e.g. using the SWRL vocabulary). Once the Content Refactorer Component generates the content alignments/annotations it interacts with the Context Knowledge Access Component of Knowlegde Access module to persist and make available the generated knowledge to the other modules of the system.

Content Filter Component. The Content Filter allows integrating the content aligned/reengineered by the Content Refactorer Component/Content Reengineerer Component according to the user preferences. This integration is done by reasoning over the content items through the relations defined in the ontologies realizing these relations between the user and content items. These preference ontologies are also a part of the AmI ODPs which are composed by the ontologies that formalize the representation of knowledge in the AmI Case. A preference is expressed considering some characteristic owned by the content item, that satisfies the user. There are specific ontologies in the AmI ODPs For items that have different preference characteristics. For example, UserContentPreferences[20] ontology expresses the relation between AmI users and content items in terms of user preferences about specific content item features and UserEventPreferences[21] ontology expresses the relation between AmI uses and external event items. How these preference ontologies are used to integrate the user information with the knowledge gathered content will be explained in Section 6.8.2 Integrating External Content Services.

6.8 Elaboration of System Workflows

After elaborating the detailed logical system architecture and the system components, we focussed on the elaboration of the system workflows between the presented modules and their components. The communication between the components is realized by a semantic “Notification and Message Format”. All events, i.e. messages about changes in the physical environment, and modification commands, i.e. messages to change or exchange individuals in the Knowledge Repository that are communicated between components of the AmI case system conform to two major principles: (1) the message itself is reduced to a minimum by communicating just a simple URI, and (2) the message itself that is referenced by the URI is expressed as one or several semantic statements. To realize this approach, the following mechanisms need to be implemented:

[18] Resource Description Framework (RDF) http://www.w3.org/RDF/
[19] Weather ontology in AmI Case http://ontologydesignpatterns.org/iks/ami/Weather.owl
[20] http://ontologydesignpatterns.org/iks/ami/2011/02/UserContentPreferences.owl
[21] http://ontologydesignpatterns.org/iks/ami/2011/02/UserEventPreferences.owl

(1) A standardized set of statements that can be used for generating messages needs to be determined; this set can be used by all involved modules
(2) An interface must be available that offers the functionality to store message statements and to return an unique identifier that refers on the message; furthermore, the reverse workflow has to be implemented that means a module can retrieve the statements of a corresponding identifier

Since the whole AmI system is based on semantic knowledge models, it is reasonable to re-use these knowledge models also as format for the messages. To evaluate the appropriate ontologies for this task, we have to distinguish between the different forms of events and modification commands that are communicated within the AmI case system. To realize the proposed identifier-based approach, the whole message will be sent to the *Message Access & Listener component* of Knowledge Access module. Then, the message interface returns a unique identifier that refers to the message transmitted before. Afterwards, this identifier can be communicated to other modules. Furthermore, it can be used to retrieve again the event message from the same component interface. This semantic notification and message format is used by all interfaces for communication between the modules. The internal communication of components inside the modules can be realized based on other proprietary forms of communication to reduce implementation overhead in this case.

6.8.1 Semantic Knowledge Management

The modules involved in the AmI case architecture require functionalities to create, read, update and delete different forms of contents: (1) ordinary content items (e.g. audio or video files), and (2) different forms of semantic knowledge models that are independent from each other and/or interconnected in one repository. Further elaborated, several functionalities to manage the aforementioned resources are required:
(1) Storage and retrieval of content items and knowledge models
(2) Semantic querying on knowledge models
(3) Semantic rule-based reasoning on knowledge models
(4) Consistency checks on knowledge models
(5) Listeners on specific modifications within the knowledge repository
(6) URI based referencing on content items, semantic representations and individuals inside the knowledge models

All these features for the aforementioned forms of contents are realized by the modules on the *knowledge representation & reasoning* and the *persistence* layer of the presented logical architecture (cf. Section 6.7). The functionalities can be used by all other modules using the *situation* and the *context knowledge access* modules*.*

A specific interface for the access of the knowledge models is provided by the *message access & listener* component. As mentioned before, the functionality to exchange messages by circulating just IDs is required for the message communication between the modules and components. This functionality is provided by the just mentioned interface:
(1) A message can be sent to the module and will be stored inside the knowledge repository; an ID is given back to the sending module.
(2) An ID can be sent to the module and the message related to this ID will be retrieved from the Knowledge Repository to send it back to the requesting module.

A detailed description of the message types can be found in Appendix C.

6.8.2 Integrating External Content Services

The *content retrieval & knowledge extraction pipeline* module in the AmI case architecture aims to integrate external contents into the AmI case environment. The interaction of this module with other modules and its internal components are depicted in Fig. 26.

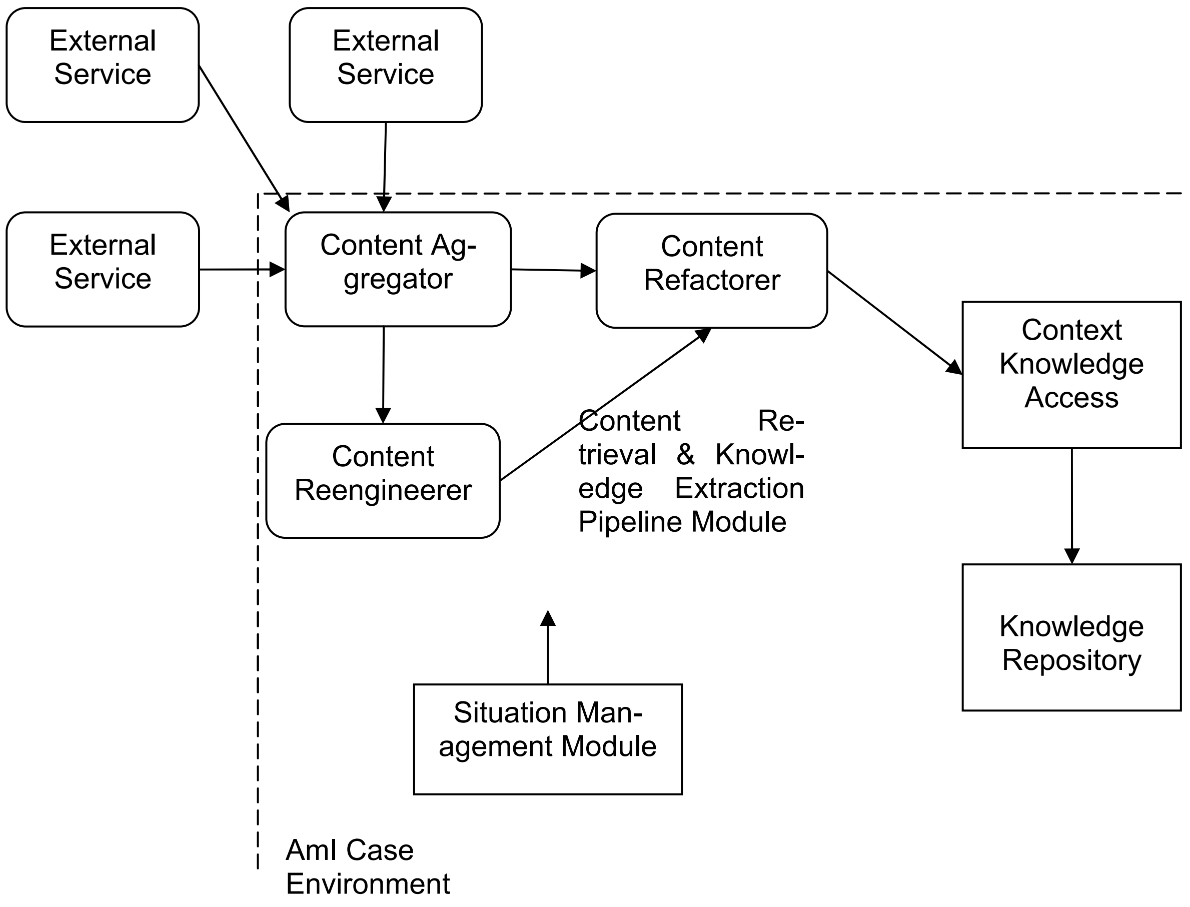


Fig. 26: Logical Architecture details of the *Content Retrieval & Knowledge Extraction Pipeline*

The content aggregator component of the *content retrieval & knowledge extraction pipeline* module, responsible for registering and accessing external services, *content retrieval & semantic lifting* layer. The external services can be any services including web services, content repository services or content management systems from which information can be retrieved using various protocols. Once the content aggregator retrieved external content, the alignment process starts. The first step is performed by the content reengineer component, which transforms the retrieved content into common ontological representation. Since RDF has become the de facto method for conceptual description or modelling of information, it will also be used for representing knowledge in the AmI case system. It should be noted that if the retrieved content is already in RDF format, then there is no need to reengineering step. The second step in the alignment process is performed by the content refactorer. Once external content transformed to a common ontological representation, there is still a need for the alignment, to map external content concepts to AmI concepts. To perform such a mapping the content refactorer requires access the context knowledge using the context knowledge access component. For example, in the AmI case weather information is represented through the ontology described at http://ontologydesignpatterns.org/iks/ami/Weather.owl. Any external content representing weather information should be aligned with the AmI Case weather ontology. This process is an example of interaction between the *content retrieval & semantic lifting layer* and the *knowledge representation & reasoning layer.* Once the content refactorer generates the content alignments/annotations it interacts with the context knowledge access module to persist and make available the generated knowledge to the other modules of the system.

The reengineer and the refactorer components of the *content retrieval & knowledge extraction pipeline* module allow to transform content from non-RDF sources to RDF and to align RDF content expressed with various vocabularies/ontologies to the set of the AmI Case vocabularies/ontologies[22]. The refactoring is performed by means of refactoring patterns. A refactoring pattern is a description of an alignment between two ontologies that can be expressed in RDF (e.g. using the SWRL vocabulary). Once the module retrieves external content and aligns it to the AmI ontology network, it should be defined how to integrate the user information with the knowledge gathered. But the integration is formally defined in the ontology network itself. In fact some ontologies from the network realize the relations between users and content objects. For example, the integration of the knowledge represented according to the Content ontology[23] and the knowledge about User represented according the User ontology[24] is realized by the `UserContentPreferences` ontology[25]. This ontology expresses the relation between AmI users and content items in terms of user preferences about specific content item features. The relation between a user and a content item is realized by the object property *preferenceOfUser* whose domain and range are User and Preference respectively. In this context the preference is about a content item, and this specialization is realized by the ContentPreference class, which in turn is a subclass of Preference. A preference is expressed considering some characteristic owned by the content item, that satisfies the user, and this is expressed via the object properties *preferenceRegardingCharacteristic and preferredCharacteristi*c. Of course, this model also allows for content features which, once combined, lead to an exact match with existing content items. Now considering that the *content retrieval & knowledge extraction pipeline* module extracts some knowledge from an external service and aligns it to the context ontology network the preferences of a user for particular new gathered content items can be found by applying reasoning on some defined rules.
Clearly, the coverage of content items that explicitly belong to the set of user favorites due to a given characteristic does not exhaust the rationale by which user-based content recommendations should be performed. Relying on user preferences for content characteristics assume great user knowledge of content features and implies that users should be able to project their mental model for content of interest as thoroughly as possible, thus limiting the potential for long-tail selection and retrieval.
For this reason, other types of user preference are formalized in the knowledge model, such as the characteristics of events as modeled in the `UserEventPreferences` ontology. Specific features and attributes of events, modeled as instances of the *EventCharacteristic* class (which can be related to user preferences via property-value pairs expressed by instantiation of the *eventCharacteristicInPreference* and *characteristicValueInPreferences* properties), may indeed be part of stated user preferences that do not explicitly match a feature of a generic content item, yet still be highly significant due to the relationships holding between content items and the factual knowledge addressed by them. With this potential in mind, rules analogous to those defined for content preferences can be defined and fed to reasoning services.

[22] The ontologies are available at http://ontologydesignpatterns.org/iks/ami/
[23] http://www.ontologydesignpatterns.org/iks/ami/Content.owl
[24] http://www.ontologydesignpatterns.org/iks/ami/User.owl
[25] http://www.ontologydesignpatterns.org/iks/ami/UserContentPreferences.owl

6.8.3 Mapping of Contents

The content integration consists of two main tasks, which are content aggregation and content alignment. Content aggregation is the task to identify and register external services whereas the content alignment task is responsible for mapping retrieved contents to the logical architecture concepts. The following figure (cf. Fig. 27) depicts the logical architecture of content integration part of the AmI case.

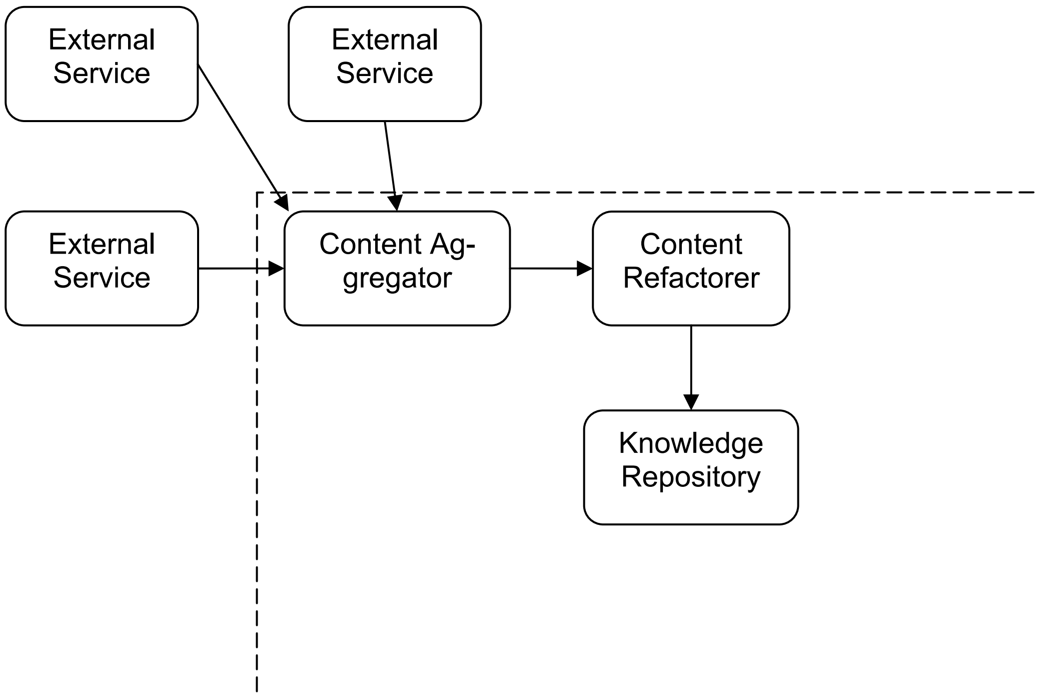


Fig. 27: Concept of Content Integration in AmI Use Case

In the above conceptual model, the content aggregator interacts with external services/systems to obtain content items, which may have different formats. The content aggregator enables registering external services or systems through its configuration. It accesses these systems based on the defined configuration parameters and retrieves the relevant content again based on the defined criteria stored in its configuration. Once the content aggregator obtains content from external services, it delegates them to the content refactorer. The *content retrieval & knowledge extraction pipeline* module performs transformations and creates corresponding instances of the content items to make them processable in the AmI Case environment. In order to achieve such transformations, the content refactorer should know how to map the external content to the concepts of AmI case. To do this, the content refactorer retrieves the concept definitions from the knowledge repository and provides a user interface to enable user to perform mappings between the structure of retrieved content and the concepts of AmI case. Finally, the content refactorer makes the transformed content items available in the AmI Case environment by storing them into the knowledge repository.

Therefore, we use *Semion* that is one of the components of the KReS framework which provides the following functionalities: (1) transformation of legacy, non-RDF data sources to RDF, and (2) the refactoring of RDF datasets by the means of recipes that can represent alignment patterns. The method implemented in Semion is based on an approach that substantially divides the reengineering process from the modeling one. The reengineering process performs the semantic lifting just extracting RDF triples driven by the OWL description of the structure of the datasource provided as input. On the other hand, the modeling process allows to introduce semantics in the extracted data set. It includes and is composed by the following components:

- Manager: is the top level component that interact with the other components of the IKS (and of KReS of course). It provides service for starting a reengineering session or a refactoring session, for storing and fetching generated or transformed data sets (via the ONM, see Section 3.1). It also hides all the Semion processes to other services or components and is able to manage several reengineers that implement the transformation to RDF from various data source type (XML, RDBMS, etc.) and several refactorers that implement different transformation behaviors.
- Semion Reengineerer: each reengineer is able to transform a specific type of legacy structured data source. KReS alpha supports XML and Relation database source types (iCal and RSS formats are currently under development). The reengineering process performs the transformation from the original data source to RDF making no semantic assumption at the domain level, but just extracting RDF triples driven by the provided OWL semantic description of the structure.
- Semion Refactorer: is responsible for the ontology mapping in KReS. It is particularly relevant considering that all the RDF data sets that are not expressed by adopting ontologies from the ontology network of KReS need to be refactored and so mapped to the ontology network itself in order to perform reasoning and eventually infer new knowledge from the provided data sets. The refactorer applies recipes provided by the RMI and execute them over - for instance - a reengineered dataset. Refactored data set are stored by using a KReS session of the Reengineering scope negotiated with the ONM.

IKS aims to support CMS providers in the exploitation of semantic technologies. To do that legacy systems must be integrated with the knowledge engineering system. Non-RDF data sources need to be trasformed in RDF to be exploitated by semantic technologies. Along with that, several external sources that can be related to the CMS content are not defined as RDF (microformats like iCal, for instance). Semion supports content transformation as basic pre-requisites for other high level functionalities. To efficiently publish content as linked data one of the major constraint is the use of well-known and shard vocabularies. Those vocabularies are in general not specific and cannot overlap with the models managed by the CMS (and by the IKS services that supports it for internal tasks), thus they need to be refactored exploiting shared vocabularies. An example of Semion is given in Appendix D.
Furthermore, *CMS Adapter* is used that involves functionalities of both content aggregator and content refactorer. The aim of CMS adapter[26] is to extract the already available semantics in the content model of a content repository as ontology. This component enables developing semantic bridges between a content repository and a knowledge-base where the content repository semantics is maintained. Bridges are defined through Content Repository API for Java (JCR) and Content Management Interoperability Services (CMIS). Concepts and their instances are transformed as ontology constructs through these bridges. An example of CMS Adapter is given in Appendix E.

6.8.4 Device Management

According to related work (cf. Section 3), we identified several process steps for the management of devices in ubiquitous information system environments. These are the following:

[26] IKS Alpha Stack Report for Semantic Data Access and Persistence Components

1) **Device integration:** Integration of devices into the AmI environment based on software parts that are necessary to communicate with the device
2) **Device detection:** Which devices are available in a current context?
3) **Service detection:** Which services or functionalities do the currently available devices provide?
4) **Device selection:** Which device represents the best presentation device to present a specific content in the current context?
5) **Content presentation:** Presentation of the content by using the provided services

Fuchs et al. (2006) introduces an approach to automatically identify the best presentation device available in the current context by using semantic reasoning, user profiles and simple rights constraints. Hofer et al. (2003) describes a Java-based framework that provides information of specific devices according to the current context. A rather hardware-based approach is presented by Eichhold et al. (2008). Eichhold introduces small Java-based sensor devices consisting of hardware and software components that can be used as sensor devices in AmI environments. Within this section, we will present the approach for the device management for the AmI case according to the aforementioned steps 2-5. Device integration of step 1 goes beyond the scope of the AmI case and will therefore not be considered in this work.

Device Detection: Within device detection, the system has to analyse which devices are available in the specific current context. In prior Sections (cf. Section 3) we already identified standards for device detection, for instance SSDP and UPnP, HAVi and Bonjour. Assumed that process step 1 (device integration) is ensured, every device within the AmI case environment can be addressed by using a unique identifier (e.g. an IP address). After this step there are two possible ways to detect devices, which are currently available in the environment:

1) **System initiated:** The Device Integration module queries for devices available in the technical environment. Each device sends back a discovery message containing the name of the device, the device type (e.g. audio device) and an URI linking to the detailed device description.
2) **Device initiated:** The devices initiates the communication by directly broadcasting the discovery message mentioned in possibility 1. The device integration module will receive this message.

Assumed SSDP and UPnP would be used within the AmI case the workflow regarding the discovery of a device would be the following:

1) The device is registered within the local network and has a unique IP address. (E.g. derived from a DHCP service inside the network)
2) The I/O management module is listening for UDP multicast messages.
3) The device broadcasts a SSDP message via UDP to the multicast address 239.255.255.250:1900 as a simple HTTP request. An exemplary SSDP message would be:

```
NOTIFY * HTTP/1.1
SERVER: Apache/1.5.5 (Linux) PHP/5.1.1
CACHE-CONTROL: max-age=1800
LOCATION: http://192.168.0.25:8080/description.xml
NTS: ssdp:alive
NT: urn:schemas-upnp-org:device:Basic:1.0
```

```
USN: uuid:550e8400-e29b-11d4-a716-446655440000::urn:schemas-upnp-
org:service:AmI-Case-Device:1
HOST: 239.255.255.250:1900
```

Service Detection: In this step, the system has to detect, which services or functionalities are provided by the currently available devices. In the AmI case, we intend to describe the services or more precisely functionalities that are offered by devices semantically. Therefore, we identified several device features that have to be represented to enable a substantiated device selection. The identification of the features relies on requirements for the AmI case. In the following, the identified features for the representation of devices are listed in Tab. 2, each with description, example and source. The semantic representation based on the Smart Product Description Object (SPDO) (Janzen et al., 2010b; Janzen & Maass, 2008) of the devices is part of the contextual part of the AmI KR.

Tab. 2: Survey of Features of Device Representation

Feature	Description	Example	Source
Function	Functionality that is offered by the device concerning input and output of specific types of contents	Loudspeaker enables output of content of type audio	Device module of AmI context know-legde based on SPDO
ContentType	Type of the content that have to be pre-sented by a device: audio, video, image and text	Type of content item “News” is audio	Device module of AmI context know-legde based on MPEG7 (alignment to semantic concept of Stack KR)
PresentationArea	Specification of de-vices that enable the presentation of video, image and/or text concerning the size of the presentation area, e.g. DIN A4	Beamer can present content in size 2x3 meters	Device module of AmI context know-legde based on SPDO
AudioChannels	Specification of de-vices that enable the presentation of video and/or audio con-cerning the audio channels, e.g. mono, stereo, Dolby ProLogic	Loudspeaker enables output of content of type audio (stereo)	Device module of AmI context know-legde based on SPDO
ContentEncoding	Encoding of content items and functional-ity of devices regard-ing supported codecs, e.g., xvid, divx, mov	Loudspeaker enables output of content of type audio (stereo) and encoding WMA, MP3	Device module of AmI context know-legde based on SPDO (alignment to semantic concept of Stack KR)
DeviceStatus	Representation of status of device that means available (for	Status of loud-speaker is out of or-der	Device module of AmI context know-legde based on

	content presentation), busy, out of order		SPDO
IndoorLocation	Representation of location of device inside the bathroom, e.g., near sink, inside shower	Loudspeaker is located near sink	Location module of AmI context knowlegde
Visibility	Representation of access rights of devices, e.g., device enables content presentation for all users or just for Anna (public, private); equivalent to access rights of contents	Device iPhone exclusively presents content if current user is Anna	Device module of AmI context knowlegde based on SPDO

Assuming a device is detected as described in the prior step the semantic devices description including the feature descriptions can be retrieved in two simple steps: (1) the device description is retrieved from the device, and (2) the semantic device description resource that is mentioned in the device description will be retrieved from the device or from an external service, which is addressed in the URI linking to the semantic device description.
After retrieving the semantic device description and the referenced dependencies inside these will be loaded inside the device integration module to proceed with the next steps. Based on the UPnP standard for basic devices[27] the minimal device description, which can – regarding the example from the former step - be found at http://192.168.0.25:8080/description.xml would look like the following:

```
<?xml version="1.0"?> <root xmlns="urn:schemas-upnp-org:device-1-0">
<specVersion><major>1</major><minor>0</minor></specVersion> <URL-
Base>http://</URLBase>
<device>
<deviceType>urn:schemas-upnp-org:device:Basic:1</deviceType>
<friendlyName>XY Audio Playback Device</friendlyName>
<manufacturer>XY Company</manufacturer>
<UDN> uuid:550e8400-e29b-11d4-a716-446655440000::urn:schemas-upnp-
org:service:AmI-Case-Device:1</UDN>
 [...]
<presentationURL>http://192.168.0.25:8080/sem-device-description.owl
</presentationURL>
</device>
</root>
```

This device description contains the reference on an OWL-based semantic device description.

Device Selection: When the Context Management or the Speech Communication module decides to present a content item, the task of the Device Selection component of the Context Management module is to select an appropriate presentation device to present a specific content according to a current context. This is done by the combination of different forms of reasoning on the semantic device description, the current context representation and the

[27] http://upnp.org/specs/basic/UPnP-basic-Basic-v1-Device.pdf

semantic annotations of the content item itself. These steps result in a prioritized list of appropriate presentation devices, which will afterwards be resorted by the suggestions coming from the presentation module. This process is described in detail in the following:

1) Semantic rules determine all devices, which fit for playback based on the factual knowledge (e.g. encoding of content item must be supported by playback device).
2) Semantic rules filter this list again based on other boolean values (e.g. devices status is ready for playback or not) to ensure playback is in general possible on all devices remaining in the results list.
3) In the last step the dialog management component has the possibility to influence, which of the top rated appropriate devices should be used regarding the maintenance of the consistency of the communication history. (E.g. if a user in the bathroom tells the system that he wants to see a movie on the cell phone screen instead of presenting it at a wall although this would be a better option regarding the size of the presentation area.)

Content Presentation: When the prior steps evaluate the appropriate presentation device, the last step is to playback the content item on the evaluated device. The Device Input/Output Management module also does this last step by retrieving the content item from the Content Repository and sending it to the appropriate device. This is done by a standardized software interface, which will not be further discussed in this document. Beside this the contextual part of the knowledge representation is adjusted to synchronize the current status of presentation devices with their real behaviour.

6.8.5 Representation of Devices

Based on the prior section we will introduce the concept for device management in T4.1. This encompasses procedural steps of device management as well as identified parameters for a device representation. At first we will present and exemplify our semantic representation of devices. Afterwards, the concept of semantic device management will be introduced. Next, this concept will be elaborated according to relevant components in the Device Input/Output Management module of the logical architecture.

Semantic Representation of Devices: As mentioned earlier concerning the device management, the system has to detect, which services or functionalities are provided by the currently available devices in the bathroom. In the AmI case, we will describe services or more precisely functionalities that are offered by devices semantically. To realize our approach of device management, we need to represent all devices within the bathroom. Also furniture can belong to this category, for instance bathtub, shower screen or washbasin. Within the AmI case, these will partly become a type of "device" as well, for instance when displaying news onto the shower screen. Beside these objects, we have pure technological devices that means input-output-devices, e.g., screens, projectors, touch screens, speaker etc. It is necessary to represent functions; capabilities etc. that are offered by the devices, for instance the ability to present audio signals. Otherwise, information about interfaces between device and system, device and device as well as device and content type has to be represented. Imagine, the user wants to extend his bathroom with more devices or out-dated/broken-down devices have to be replaced. Then, we need to know how the devices are characterized.
According to these aspects, we decided to use Smart Product Description Object (SPDO) as foundational ontology for describing generic information about devices that represent sub types of products. SPDO is a semantic product information that describes physical products especially within intelligent environments (Janzen et al., 2010; Janzen and Maass, 2008). For the description of functions, capabilities (IO) etc. of devices, we extended SPDO by a device-specific module. Before, we identified several device features that have to be represented to

realize a substantiated device selection in the physical environment (cf. Appendix F). The device-specific SPDO plugin[28] extends the domain-independent information of SPDO Core according to the features of the device domain. In conjunction, both parts present the representation of devices for the AmI case. Based on this, each device in the bathroom is described by one instance of device representation. Therefore, we get a “pool of devices” represented semantically (cf. Fig. 28) that enables rule-based generation and integration of statements about matchings between types of contents and available devices. Furthermore, underspecified relations or concepts can be completed automatically.

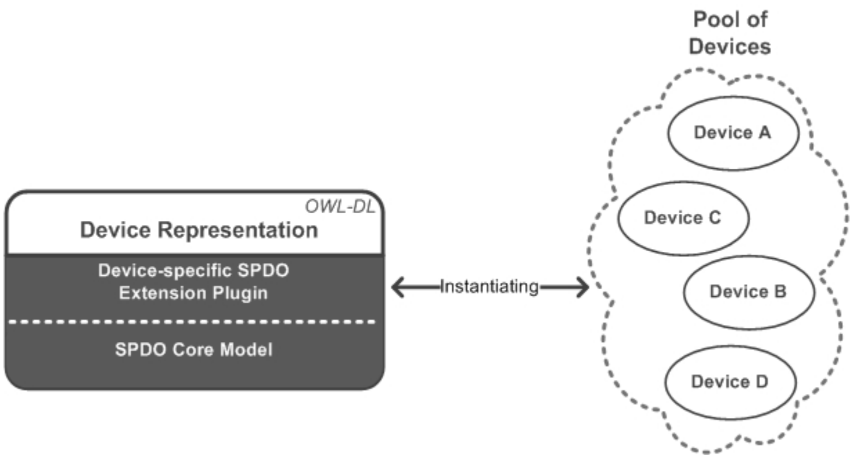


Fig. 28: Representation of devices in IKS Task 4.1

Appendix F shows the essential features of the device representation giving a short description as well as extracts of the representation of the features in the Turtle[29] language. To specify the generic concept of *Product* within SPDO Core, we generated the concept *Device* that is an equivalent class of *Product* and represented as follows:

```
@prefix owl: <http://www.w3.org/2002/07/owl#> .
@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> .
@prefix : <http://im.dm.hs-furtwangen.de/ontologies/spdo/2010b/
SPDO_AmICaseDevice.owl#> .

:Device     a owl:Class;      rdfs:comment     "AmI case: Device inte-
grated in bathroom"@en; rdfs:label     "Device"@en;
owl:equivalentClass     :Product .

:upnp-id     rdfs:domain     :Device .
:hasIndoorLocation     rdfs:domain     :Device .
:hasVisibility     rdfs:domain     :Device .
:fitsTo     rdfs:domain     :Device .
:fitsTo     rdfs:range     :Device .
:isVisibilityOf     rdfs:range     :Device .
```

Concerning the matching of device functions with specific content types, we used the concept *MultimediaContent* offered by the MPEG7 standard. Later, this part could be filled by an ontological representation of content types given by the Stack. In this case, we have to assure that the representation can also handle AmI-specific types of input. The concept *Multi-*

[28] http://im.dm.hs-furtwangen.de/ontologies/spdo/2010b/SPDO_AmICaseDevice.owl
[29] http://www.w3.org/TeamSubmission/turtle/

mediaContent covers several sub classes, for instance Audio, Video, and Image. But in our physical bathroom environment we have to handle further specific types of input:

- Speech input (specification of audio content)
- Sensor-based input of implicit user activity (e.g., taking a towel, going to the sink, brushing teeth)
- Sensor-based input of explicit user activity (e.g., pressing a button on touch screen)

Therefore, the concept of *ContentType/MultimediaContent* has to be extended according to these AmI-specific input types. As speech input is handled via the aforementioned audio content type, we integrated a concept *UserActivity* that presents sensor based input of implicit and explicit user activities within the bathroom.

Semantic Device Management: The semantic representation of devices will be managed within the Device Input/Output Management and the Device Integration module. The handling of a device inside the AmI case environment requires the integration of the semantic representation of this specific device into the contextual part of the knowledge representation. On the one hand, the steps of *device detection, service detection* and *device integration* ensure the availability of the device representation in the current context. On the other hand, these steps care for the correct technical integration of the device into the technical environment. That means that visibility and location of the device are adjusted according to the circumstances, e.g., a mobile phone that has a private visibility as well as flexible indoor location. Later, these service descriptions are required for the *device selection* step when the system has to present specific types of contents within the bathroom. From a technical perspective, the device handling steps require further additional steps until the device can be used within the bathroom environment:

1. Functionalities (individuals of class *Function*) of devices can be found in the device representation with the predicate *hasFunction*. With the predicate *enablesInputOf* and *enablesOutputOf* devices can be connected to specific content types like audio, video, image, user activity etc.
2. These functionalities are implemented as services within the OSGi service Java Archive (JAR) that is also referenced in the UPnP device description. After integrating the OSGi service within the device I/O management module, the appropriate services can be called as described in the semantic representation.
3. Standardized interfaces for all supported content types are provided by the AmI case environment, in particular the device I/O management module. These need to be implemented for the specific devices. This ensures compatibility independent from the device-specific APIs.

All devices represented as individuals within the contextual part of the knowledge representation can be used if they possess a *hasStatus* relation to an individual of the type *DeviceStatus*. Depending on the current usage of the device this status can switch between available, busy or out of order when a device is integrated into the bathroom environment. The Device Integration module adjusts this statement automatically. If a device is removed from the environment, the *hasStatus* relation will also be removed. This kind of representation of device status has the advantage that available, busy or broken devices in the bathroom can be easily determined. An exemplary device handling process is given in Appendix H.

6.8.6 Natural Language Processing in the AmI Use Case

This section describes the conceptual overview of the Language Processing infrastructure for the support of Speech I/O, language interpretation and generation for the IKS AmI case environment. The fundamental tasks by the **d**ialog-based interaction **m**anagement are the following:

- Speech Input Interpretation
- Discourse Interpretation
- Dialog Management
- Presentation

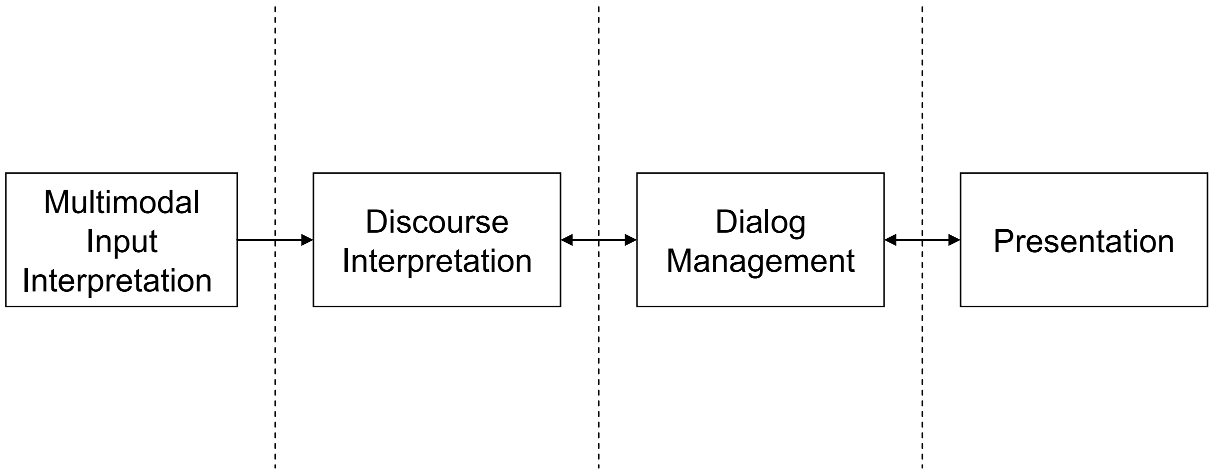


Fig. 29: Dialog-based Interaction **M**anagement Basic Components Blocks

Fig. 29 gives an overview on the basic components blocks in a **d**ialog-based interaction **m**anagement system grouped by general task affiliation. For the sake of simplicity the picture depicts just basic data flow between the four blocks.

Speech Input Interpretation: The basic objective of the Multi-modal Input Interpretation (MMII) Block is to deliver a linguistic interpretation of the information coming from the different modality (e.g., speech, gesture, sensors data) recognizers. For this purpose 2 steps are foreseen:

1. Interpretation: recognition hypothesis form each single modality recogniser is associated to a set of **underspecified** semantic objects ***I(USO)*** representing the possible intentions of the user by the given modality input.
2. Fusion: different interpretations are compared and put together in a set of **underspecified** semantic objects ***F(USO)*** (level of underspecification ***U-Level*** is at this stage ***U-Level(F(USO)) <= U-Level(I(USO))***), in order to:
    a. Resolve cross modal references
    b. Reduce the set of USO by cross modal disambiguation.

In Fig. 30 we give an overview of the basic logic components involved in the MMII task.

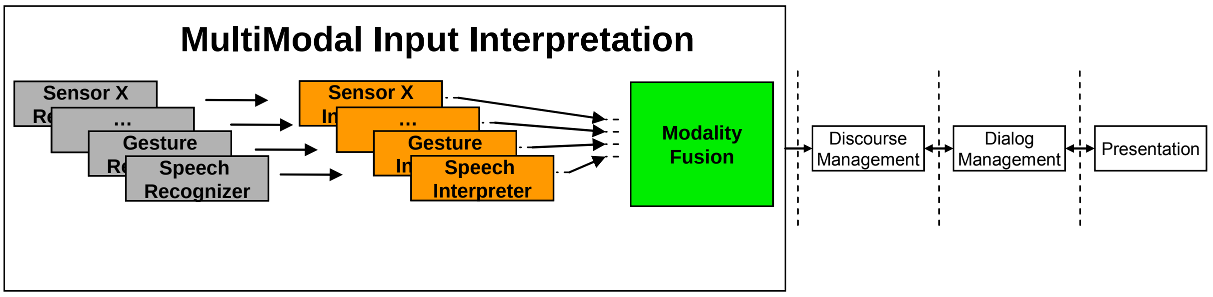


Fig. 30: Components in the MMII Block

The first group of components are so called recognisers. In a first shot they get the rough information and convert it to recognition hypothesis e.g., the speech recogniser converts speech to strings representing one or more recognition hypothesis. Such hypotheses can be represented in terms of lattices and can be expressed using standards like the Multi-modal Extensible Markup Language EMMA[30].

Integration of Language Interpretation with AmI Case User Representation: The task of Speech Recognition can be performed adopting a statistic analysis approach (where probabilities for word chains realisation are extracted from corpora) or Grammar Based Language Models where grammar modules are matched to the corresponding dialogue situations. The latter has been adopted for the AmI case as they are more suitable in typical dialog systems applications and we already have well specified situations we can refer to in order to define grammars. For an overview of the grammar models see Fig. 31.

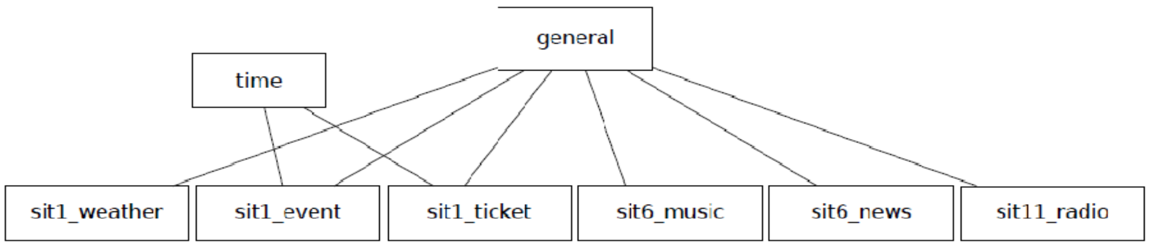


Fig. 31: Modular Structure of the Speech Grammars for the AmI Case Situations 1,6 and 11

First grammars for the AMI Case have been specified for the situations 1, 6 and 11, selected for the first implementation. For the specification of the grammars the W3C recommendations have been followed and the Nuance recogniser has been used for evaluation.

Speech Interpretation is then performed by generating set of instances of the specific semantic model containing interaction types and domain information. The results of this task have to be packed in a container language as for the case of the recognizers. Language interpretation needs a set of interpretation roles in order to be performed.

Internally, a dedicated NLU syntax using extended type feature structures (eTFS) (SemVox, 2009) is used that simplify reasoning operations in the forthcoming stages of processing (e.g., discourse management, dialog management). For an example of such interpretations see the following snippet:

[30] EMMA: Extensible MultiModal Annotation markup language, http://www.w3.org/TR/emma/

```
<utterance name="S_SHOW_WEATHER_LOCATION">
   <phrases>
      <phrase>VP_WOULD_LIKE see [WEEKDAY
         N_LOCATION]</phrase>
      <phrase>VP_WOULD_LIKE see N_LOCATION
         PP_WEEKDAY</phrase>
      <phrase>VP_WOULD_LIKE see N_LOCATION PP_WEEKDAY
         TIME_OF_DAY</phrase>
   </phrases>
   <semantic-interpretation>
      <object type="WeatherRequest">
        <slot name="hasInitiator">
          <object type="User"/>
        </slot>
        <slot name="hasContent">
          <object type="location">
              <slot name="hasContent">
                  <variable name="N_LOCATION"/>
              </slot>
          </object>
          <object type="time">
            <slot name="hasContent">
                  <variable name="PP_WEEKDAY"/>
            </slot>
          </object>
        </slot>
      </object>
   </semantic-interpretation>
</utterance>
```

Interpretation components are responsible for the distillation of user intentions from recognition hypothesis. The result of this task is a set of **instances** of the specific semantic model containing interaction types and domain information expressed in a semantic language (e.g., owl). Depending on the intention (e.g., questioning, commanding, etc) by the user and on the type of input (e.g., multi-modal, only speech, etc.), the instances can be more or less underspecified. The results of this task have to be packed in a container language as for the case of the recognisers. For sake of simplicity, well practices and coherence to the IKS Alpha the container language should be used over the all of language processing analyses. Language interpretation needs a set of interpretation roles in order to be performed.

After every interpreter has produced distinct sets of user intentions, in the multimodal fusion task interpretation are put together in order to coordinate input from different modalities and disambiguate interpretation. This is done by means of unification of pairs of user intentions.
The result of the fusion activity will be passed to the next component. **Note**: there is in general interplay between speech grammars, tasks and domain models so that additional effort is needed for the harmonisation of the analysis with related components.

Discourse Management: Discourse management is the task of disambiguating user intentions by means of comparing them with the context of discourse i.e., with the knowledge acquired during past interaction. The discourse manager is usually composed of a stack in which all past interaction objects are collected and a set of operations based on the comparison of the user intentions against the objects in the stack in order to extract missing information and disambiguate utterances. The result of a discourse disambiguation is a defined user intention that can be passed to the Dialog management.

Dialog Management: Once the hypothesis from recognition has been interpreted to user intentions and disambiguated to the most appropriate user intention, the dialog management decide on the next steps of the interaction basically following interaction patterns. The dialog management achieve following tasks:

- turn taking management;
- reaction planning;
- communication with internal knowledge retrieving services.

The dialog management task can be achieved only with direct access and negotiation of communication protocols with the knowledge retrieving services.

Presentation:
The presentation task is articulated in three basic sub tasks: modality fission, multimodal generation and multimodal rendering. The presentation task can be seen as sort of specular with respect to the interpretation task.

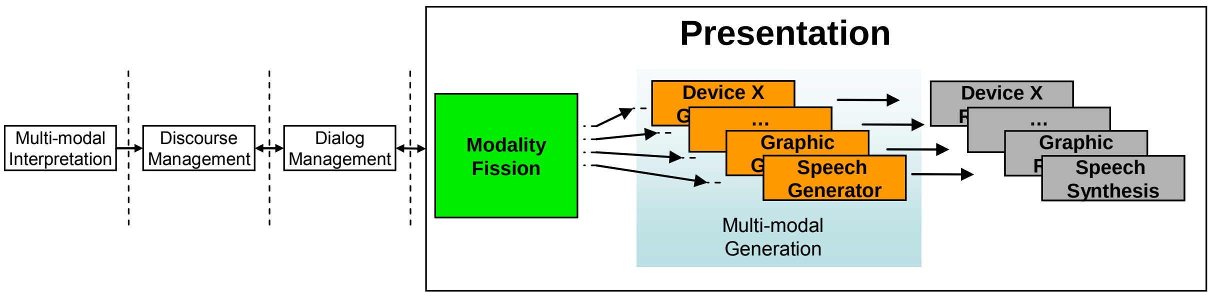


Fig. 32: Components in the Presentation Block

After the dialog management has retrieved needed information and passed it to in form of a semantic instance to the Presentation block, presentation has to be planned and performed. The modality fission task distributes presentation priorities and competences over the different modality generators and passes to each generator a “generation recommendation” in form of an XML coded presentation object. The generators transform the generation recommendation in a form suitable for the use in the addressed renderer e.g., the speech generator transform the string coded utterance received from the fission component in an instance of the Speech Synthesis Mark-up (SSML) Language suitable for the text-to-speech component. The rendering task is the one of render the XML coded input in the desired modality e.g, text-to-speech in the Speech Synthesis component.

Needed logical components have been defined with respect to the tasks evidenced in the AmI case description. Speech Grammars have been specified for the 3 selected use cases for the use in the Nuance Framework.

AmI Case Peculiar: for the AmI case a more powerful discourse and dialog management is needed so that extra solutions will be taken into consideration.

6.8.7 Natural Language Processing Infrastructure

Summarising, the AmI Case Speech Communication module aims at providing natural language and in general multimodal interaction functionality to the system. Interaction of this module with other modules and its internal components are depicted in Fig. 33.

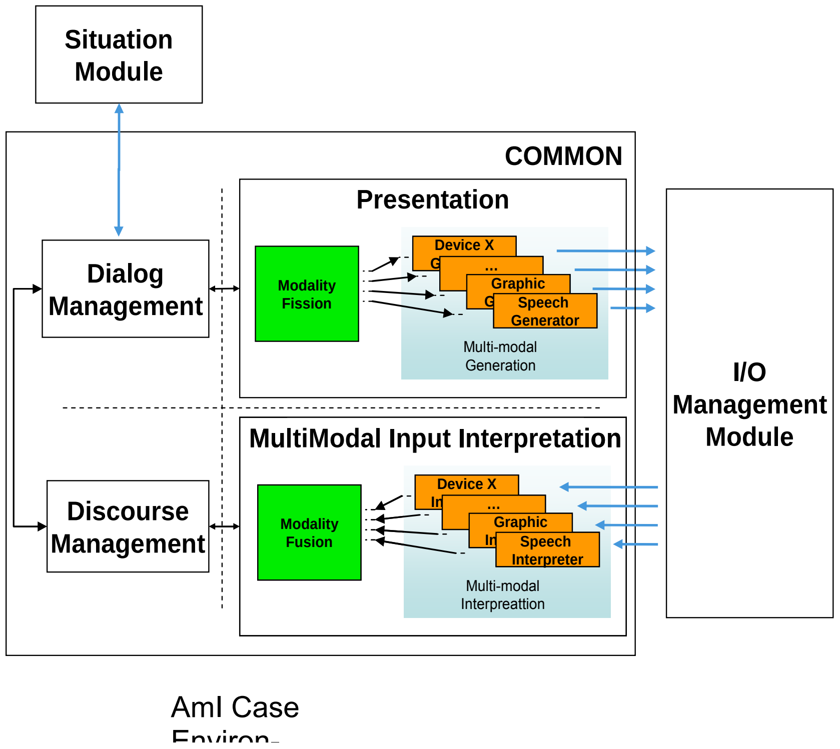


Fig. 33: Logical Architecture details of the Speech Communication Module COMMON

6.9 Software Architecture of AmI system

To go the step from the detailed logical architecture developed in the former Sections (cf. mainly Section 6.7) to the implementation of the AmI case system as described in the following Section (cf. Section 7) one further major step is required: The elaboration of the software architecture. While some modules and components in the logical architecture can be developed based on already existing frameworks, other modules and components need to be implemented from scratch. In this Section we will focus on the reuse of existing software solutions and their adaption for the AmI case system, while the modules, which were especially implemented for the AmI case, will be addressed in Section 7.

The following figure (See Fig. 34) shows the software modules and components of the AmI case system. The green marked modules and components were implemented from scratch in exactly the same way as they appear in the logical architecture, while the blue marked modules were implemented in a slightly different way as in the logical architecture and have therefore also slightly different names. These modifications are mostly based on the reuse of already existing frameworks (bold boxes), which required a slightly different implementation perspective than the conceptual perspective we had during the conceptual phase. Nevertheless the functionality provided by the implemented modules is adequate to the functionality described in the prior Sections. In detail the modifications are the following:

- The Device Input/Output Management Module and the Device Integration Module are combined and realized as Device Management Module
- The Content Retrieval & Knowledege Extraction Pipeline is realized as Semantic Content Extractor Module (SCEM) mainly based on the KReS framework
- The components of the Knowledge Access, Content Access, Rules & Reasoning as well as Knowledge and Content Repository Modules are realized under the name Knowledge Repository Module because of the intense reuse of the functionalities provided by TnT2

When looking at this figure you should always keep in mind that some frameworks only support some of the tasks expected from the different modules and are extended by own source code. These are the Device Integration, the Knowledge Access Module, Rules & Reasoning and Knowledge Repository Modules. Also some frameworks are attached using wrapper classes for the AmI case environment, these are Nuance Dragon, Cling, and JENA inside Tip ‘n Tell 2 (TnT2).

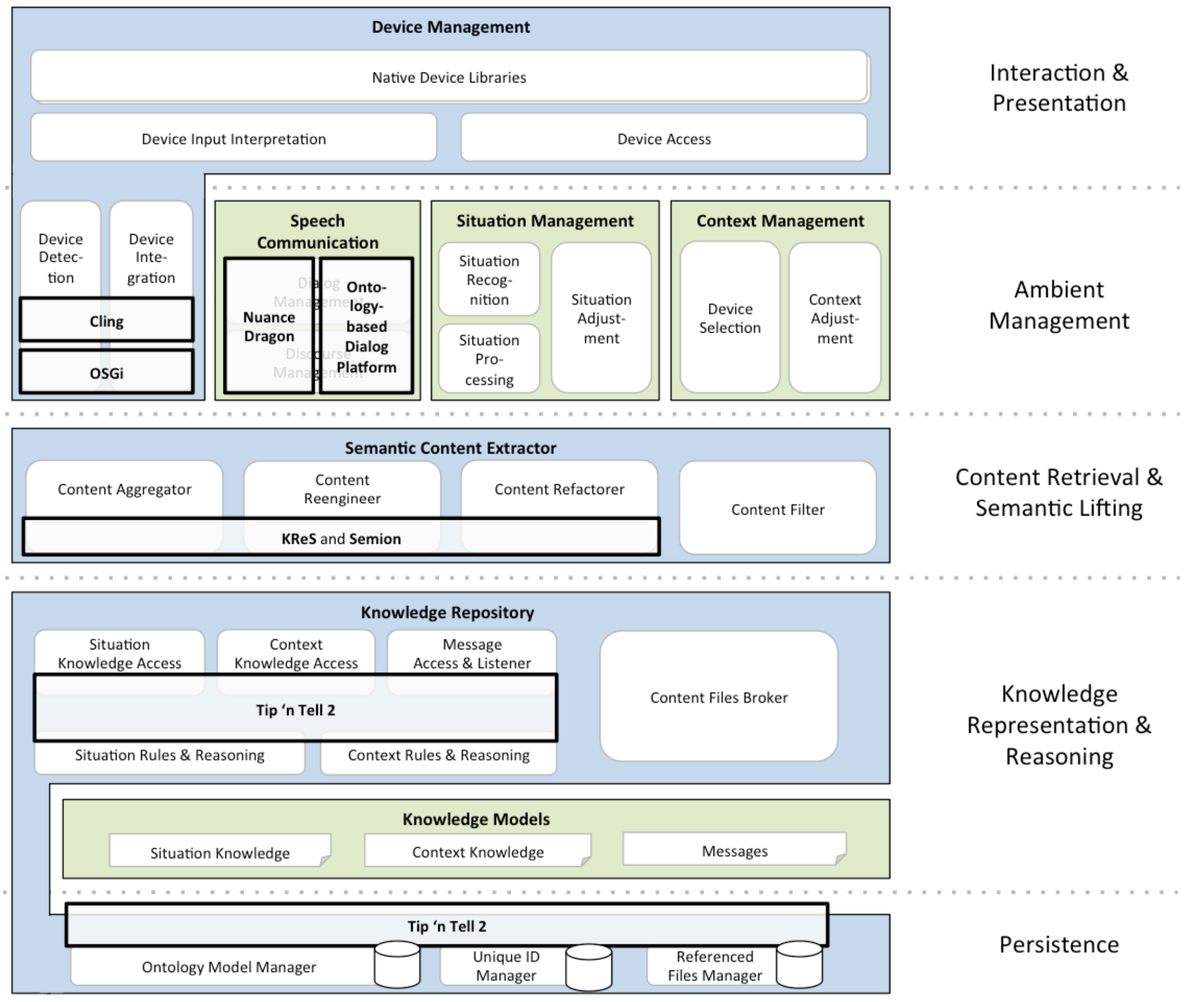


Fig. 34: Software architecture of AmI use case

After describing the software architecture, in the following, we will describe the aforementioned selected frameworks.

OSGi[31]: To enable the development of a loosely coupled, dynamic, component-based system infrastructure, OSGi will be used for the modularization of tasks and the user of service concepts within the AmI case system. To be independent from a specific OSGi implementation like Apache Felix[32] or Eclipse Equinox[33], all bundle and service descriptions will follow the OSGi standard without implementation of specific aspects.

JENA[34]: The JENA Semantic Web Framework is a Java framework, which provides all necessary functionality to work with semantic information representations in Java. This encompasses ontology management in OWL and RDF as well as reading and writing capabilities in several formats. A SPARQL query engine and support for several semantic reasoners is integrated as well.

Cling[35]: The Cling framework is a Java library to create UPnP software stacks in Java. In detail, it enables the dynamic discovery and service identification of devices, which broadcast their availability and provided services using the UPnP protocol.

Tip 'n Tell 2 (TnT2): The semantic middleware TnT2 will be used in the AmI case system to provide most of the functionalities required in the *knowledge representation & reasoning* as well as on the *persistence* layer. TnT2 is able to manage semantic knowledge sources and to combine them to a single knowledge base. This enables interlinking of different knowledge sources as well as the execution of rules or consistency checks on the knowledge base via extension modules. Furthermore, TnT2 can be extended dynamically by own modules that can access the knowledge base and register listeners on specific forms of content changes inside the knowledge base. Some parts of TnT2 are based on JENA.

KReS: KReS has been designed to support a pattern-based approach with respect to the definition and composition of ontologies and tasks in the AmI case system. This approach is also reflected in the way KReS manages rules. Rules can be applied for different kinds of tasks, sometimes independently of the ontologies the rules are related to. Even if the rules - at the core level - are managed as ontology fragments (by the means of the SWRL language), KReS manages them in a separate environment, making them available on demand for any subset of the AmI case ontology network.
The following concepts have been introduced:

- *Ontology scope*. Ontologies are provided within so-called Scopes, which are cotrolled sub-sets of the ontology network related to a particular task or domain within the CMS.
- *Ontology space*: an access-restricted container for synchronized access to ontologies within a scope.
- *Session*: in order to safely apply changes to ontologies at runtime, for a single specific task, client applications use KReS sessions. The rationale is to avoid potential inconsistency originating from multiple concurrent tasks, and to control the propagation of changes to the TBox (for security reasons).
- *Recipe*: represents a set of rules, aggregated into a collection in order to be executed for a specific purpose. Recipes can be defined at configuration time in an OWL ontology, and can then be added, modified or deleted at runtime. The relation between recipes and ontology scopes is orthogonal, in that they can be executed on any active scope within KReS.

[31] http://www.osgi.org/Main/HomePage
[32] http://felix.apache.org/site/index.html
[33] http://www.eclipse.org/equinox/
[34] http://jena.sourceforge.net/
[35] http://teleal.org/projects/cling/

Semion: Semion is a set of components that support the following tasks: (1) the transformation of legacy data, from non-RDF sources, to RDF, and (2) the refactoring of RDF datasets by the means of recipes (as they are described in KReS) representing alignment patterns, refactoring patterns, or a mix of the two. The method implemented by Semion is based on an approach that differentiates the reengineering process from the modeling process, i.e., simple transformation to pure RDF triples from the refactoring of RDF triples themselves. The aim of the reengineering is to transform the legacy (non-RDF) data source provided as input without fixing any assumption on the domain semantics, that is, instead, introduced modeling the triples through the refactoring that allows to use good practices of knowledge engineering and focusing the target to the AmI case ontology network.
The AmI system supports the exploitation of semantic technologies into non-semantic (and legacy/external) data sources. To do that, legacy systems must be integrated with the knowledge base of the AmI system. It needs to be possible to transform non-RDF data sources into RDF, if they are to be exploited by semantic technologies, e.g., queried or reasoned with.

Ontology-based Dialog Platform (ODP): The multimodal dialog platform ODP (Pfalzgraf et al., 2008) will provide functionalities for the Communication module in the AmI case system. ODP is able to provide speech interpretation, fusion of multimodal input and interpretation in context of the dialogical context (Sonntag et al., 2010). The ODP platform also provides access to speech recognition and synthesis engines (for the AmI case we will base this on a Nuance speech server). The platform offers also capabilities for the realisation and orchestration of presentation and dialogical interaction management.

Nuance Dragon: For enabling speech recognition we will use the Nuance Dragon NaturallySpeaking 10[36]. The server-based processing supports both trained and untrained speech-profiles and can be controlled in an intuitive way. The platform is well integrated with the ODP and can handle basic W3C speech standards.

7 Implementation of the AmI Use Case

The Eclipse IDE has been used as the development tool by the developers of the AmI use case. This IDE comes with the built-in Equinox environment, which is an implementation of OSGI framework. This provided flexible development for an OSGI based system. The OSGI framework will be elaborated in the paragraph below. Code maintenance has been obtained by SVN version control system. Servers of Salzburg Research have been used to provide a common repository. Eclipse ID has also support for keeping the developed code synchronized with the SVN through its graphical interface. To track any software issues regarding the AmI System, the Trac[37] system of Salzburg Research has been used. For instance, this system provides a ticket system to inform about bugs or enhancements related to specific components of the system. The documentation of the implementation procedure has been also put to the Trac system.

First of all AmI System developed in this use case is an OSGI based one. OSGI technology is a dynamic module system for Java language. It provides reusable, collaborative components constructing the application and these components can be plugged to the system without requiring a restart. OSGI provides a service oriented architecture. This provides dynamic reciprocal discovery of services provided by components for collaboration. In the AmI System

[36] http://www.nuance.com/for-business/by-solution/dragon-enterprise/index.htm
[37] https://svn.salzburgresearch.at/trac/iks-project

each component were implemented as an OSGI bundle, which is the pluggable unit to the system. As being an OSGI based system, Java programming language was used in implementation of AmI Use Case. OSGI framework meets the requirement that we identified as implementation guidelines (cf. Section 3.1) stating that integration of modules at runtime should be managed by semantic service orchestration to allow the creation of highly flexible AmI environment and the implementation of the resulting architecture should be based on open standards. As it provides an easy way to plug new components and services to the running system, it also meets the guideline for light-weight protocols that enable the simple creation of new modules and services as well as their integration at runtime. Furthermore, the services within the system are easily traceable and obtainable thanks to OSGI framework.

The components of AmI system were implemented based on the design of principles that are already explained in Section 6 and Section 3.1. Each component serves its services to other components through interfaces independent from the underlying implementation. Through this modality, the components realize the concept of encapsulation by clearly defined service interfaces in order to ensure high flexibility and fewer dependencies. The components are developed to realize subparts of main modules of the system as elaborated in Section 6.7. Components implemented will elaborated in detail in the following subsections.

7.1 Device Management Module

The whole device management process is within the AmI case system managed by four components: (1) *Device Detection Component* (2) *Device Integration Component* (3) *Device Input Interpretation Component* and (4) *Device Access Component*. While *Device Detection Component* and *Device Integration Component* are responsible for the communication with devices the *Device Input Interpretation Component* and *Device Access Component* care for interaction with the rest of the system. The *Device Detection Component* cares for the monitoring and notification of devices status. Each time a device enters or left the environment the system is notified. The *Device Integration Component* is responsible for the integration of devices in the system. This is done by integrating of removing the semantic representation of a given device in the knowledge base. The component also offers interfaces that define methods for passing commands to devices e.g., presenting content on the screen. The *Device Input Interpretation Component* is responsible for the interpretation of sensory devices input. In particular the component is responsible for the identification of the user, her activity and position in the environment. The interpretation is then broadcasted to the system in form of dedicated messages. The *Device Access Component* cares for the orchestration of content presentation on devices. The component provides an interface for other components to pass content for presentation to the I/O Module. After a generation step a presentation is passed to the appropriate integrated devices over dedicated interfaces, which have been preselected in the *Device Selection Component* depending on the type of content that has to be presented.

7.2 Context Module

The Context Module consists of two components: (1) *Context Adjustment* and (2) *Device Selection*. While the *Context Management* cares for the adjustment of the context representation (i.e. AmI ODPs), the *Device Selection* offers a service to retrieve a rated listing of devices that are currently available and enable the presentation of a given content item to the user in the bathroom. The aforementioned *Context Management* encompasses:

- detection and broadcasting of new content items in the context representation
- detection, broadcasting and removal of devices in the bathroom environment

- adjustment of the representation of the current position of the user in the bathroom in the context representation
- adjustment of the light scenario in the bathroom based on the context representation

7.3 Situation Module

The situational part of the KR (cf. Section 5.1) is managed by the two components of the Situation Module: (1) *Situation Recognition & Processing* and (2) *Situation Adjustment*. The first component combines the tasks of the conceptual *Situation Recognition* and *Situation Processing* component and cares for the recognition, processing and broadcasting of situations concerning situational changes. Since situation recognition and processing tasks are highly interconnected, these two conceptual issues were realized in one component. The process of linking the context and the situation knowledge representation is described in detail in Section 7.6. The second component - *Situation Adjustment* - cares for the adjustment of the situation based on contextual changes in the bathroom environment, e.g., in case the user moves to another location.

7.4 Speech Communication Module

The communication Module consists of a single component, the *Communication Adapter Component* and the following additional non IKS-specific multimodal dialog tools: the Nuance Dragon Naturally Speaking[38] has been used for speech recognition, the SVOX TTS[39] has been used for text-to-speech generation and the SemVox ODP Server as dialog server[40]. The component is responsible for the speech interpretation, discourse/dialogue management and generation of speech output. Once the user performs a speech request to the system this is recognised by the speech recogniser, which generates a recognition hypothesis. The component checks the given hypothesis again the situational context and identifies the expected task, which is then broadcasted to the system in order to produce the expected presentation. The multimodal presentation is produced by broadcasting content presentations to the *Device Access Component* and speech presentation directly to the speech synthesis system.

7.5 Semantic Content Extractor Module

The Semantic Content extractor module realizes the functionalities of the Content Retrieval & Knowledge Extraction Pipeline as described in Section 6, the Semantic Content Extractor Module is responsible for gathering external content from different sources and aligning them into AmI System. It consists of four components: (1) *Content Aggregator*, (2) *Content Reengineer*, (3) *Content Refactorer*, and (4) *Content Filterer*.

Content Aggregator is responsible from gathering different types of contents from external sources. In the scope of AmI Use Case it collects 5 types of data namely weather condition information, external events, music collection, news and calendar information. All these data are collected from different services. News, external events and weather condition information are obtained through RESTful services of different services. Music content is obtained from a Jackrabbit content management system by using standard JCR API. Lastly, the calendar information used is obtained from Google Calendar services via its specific API. As

[38] http://www.nuance.com/for-business/by-solution/dragon-enterprise/index.htm
[39] http://www.svox.com/Technology.aspx
[40] http://www.semvox.de/en/menu-services/menu-services-products.html

soon as the data arrives, an XML representation of data is sent to Content Reengineer Component so that it would create an ontology on which rules are executed to form actual content items in the backend knowledge base. One key issue to be dealt with when processing data from external resources beyond our control is the heterogeneity of the possible ways these resources can be authored or presented. Because the content retrieved from these resources will ultimately have to conform to the AmI Context model, we need a way to transform non-RDF data into RDF. However, knowing that the original resource is a relational database or an XML or JSON document is only part of the solution of this problem: even with this known, we cannot predict which relational schema or XSD or vocabulary is used. Even RSS only specifies a vocabulary for the header and metadata of a feed, whereas the data themselves can be described according to an arbitrary XSD. Therefore, a blind general-purpose approach does not apply and the transformation into RDF by a certain model has to be at least a 2-step process.

The **Content Reengineer** implements the first step of this process, where heterogeneous data are transformed into RDF graphs whose metamodel is that of the *original representation language*. The reengineering of non-RDF data is a crucial aspect when bridging legacy and semantic-enabled systems, which is why an implementation of this functionality was prioritized for the IKS Alpha implementation. The Content Reengineer provides an OSGi service that wraps the Semion Reengineer functionality available from IKS KReS Alpha. This functionality is used by invoking its POJO (Plain Old Java Objects) API, as opposed to the RESTful one, which would have implied a greater overhead against no sensible advantage, if applied internally within the demonstrator. The Semion Reengineer is a pluggable system that can process as many formats as are the reengineering engines registered with it. This step is a black-box process, in that it is not parametric and requires no configuration, other than by specifying the content format. As far as the requirements of this use case were concerned, we were able to identify data sources and services that could deliver output in some XML formalism, therefore we only registered the KReS XML Reengineer and omitted other engines such as the DB Reengineer.
The reengineer output is an RDF graph which describes semantically the exact same knowledge as the original XML content item, according to a formal model of the core XML language structures such as nodes, elements, attributes and their values[41]. Such reengineer content is then ready to be fed to the SCEM Refactorer. Having transformed the original external content into an RDF graph that expresses the same knowledge in semantic form, a second step extracts all and only the knowledge that is relevant to the target knowledge representation (i.e. the AmI Case Context model) and transforms it accordingly. This step is called *refactoring* and is carried out with the execution of *inference rules*. These are provided in the *KReS Rule Language* syntax[42], which maps seamlessly to SWRL rules and SPARQL construct queries. Sets of such rules were authored for all the external service vocabularies considered for the domains at hand (i.e. weather forecasts, event feeds, event booking, calendar management and music playlists).

These rule sets, called *Recipes*, have been made available as resources within the **Content Refactorer** component, which in turn wraps the *Semion Refactorer* module available in IKS Alpha. Refactoring is a parametric operation that can be configured with the location of the ontology containing the desired recipes, as well as the URIs that identify the recipes themselves. A default configuration is provided as part of the AmI Case Knowledge Repository component. By employing different recipes for each call to the refactoring service, we can ensure that a single reengineered content item is processed so that only the rules that pertain to a given domain are applied at one time. This way, we can prevent the generation of

[41] http://www.ontologydesignpatterns.org/ont/iks/oxml.owl
[42] http://stlab.istc.cnr.it/stlab/KReS/KReSRuleLanguage

spurious triples deriving from the concurrent application of multiple rules to the same RDF graph.

Content Filterer reacts according to situation change in the system. In case of a situation change, it retrieves the necessary information types and delegates the necessary requests to the Content Aggregator Component. It also provides services to other components to filter already available content items according to current user and desired content type. Content Filter Component considers user's preset preferences. If there are sufficient content items satisfying the user preferences, they are returned as filtered items; otherwise the filtering service returns content items only considering the desired type.

7.6 Linking "Situations" with "Context"

In order to make use of the contextual and situational part of the KR in the AmI Use Case system for storing and delivering information to the user, the parts need to be aligned, which is currently done at design time by the Situation Module (cf. Section 7.3) and will be explained in detail in this section. Before this alignment can be used to provide content to the users of the UIS, several other steps are required, e.g. the collection of context data from the bathroom environment (cf. Section 7.1 and 7.2) as well as the retrieval and semantic representation of content items (cf. Section 7.5).
To explain the whole process of bringing the context and situation part of the KR together in our bathroom environment, at first we need to have a look at the management of the context ontology network. When the context ontology network is up-to-date, i.e. with respect to the current physical situation in the bathroom - containing users, their locations and all physical devices available - the network forms the basis for the determination of the appropriate propositional CM (cf Section 5.3). An appropriate propositional CM in this case means that its semantic situation description fits to the current "real" situation in the bathroom and therefore the AmI Use Case system "executes" specific steps described by the propositional CM. This task is performed by the situation module, which creates a situational representation of the current situation in the bathroom. Afterwards this representation of the current situation is compared with all propositional CMs available in the situational KR, using a rule set that is delivered with the propositional CMs in addition to their ontological structure in combination with Java logic. The execution of the semantic rules as well as the SPARQL queries used within the whole process is performed using Tip 'n Tell 2 (TnT2, cf. Section 6.9) and the JENA framework, which is already integrated in TnT2. Before the AmI Use Case system tries to execute the steps described in the appropriate propositional CMs several other constraints have to be checked. On the one hand, these constraints are provided by the propositional CM itself, e.g., preconditions regarding prior situations. On the other hand, there are specific UIS issues, e.g., the availability of a presentation device at the location of the user, which is capable to display the content item selected for presentation. If a propositional CM fits to the current context information available in the context ontology network as well as to the current situation representation, and all mentioned constraints are fulfilled, the AmI Use Case system performs actions related to that particular situation as specified in the propositional CM, e.g., the presentation of a specific content item or the request for a specific interaction performed by the user. The described determination of an appropriate propositional CM is performed continuously, i.e. when parts of the KR that are relevant for this determination change, the determination algorithm is triggered to check whether the current situation is fulfilled and/or a new situation occurred in the bathroom. If more than one propositional CM would fit to the current situation they are all performed one after another if no user interactions are required to proceed to the next fitting CM.
The semantic content extractor module retrieves the content items required in a specific propositional CM, i.e. a specific situation in the bathroom, from several semantic and non-semantic information sources, and reengineers and refactors the information so that it fits the

concepts provided in the context ontology network. In principle, the AmI Use Case system is not a CMS in itself and hence does not store content items, however, in order to select and reason over content, some information about the content needs to be stored, e.g. the topic of a news article, or the extent and location of a weather forecast. Such information is not always available in a semantically rich representation, but commonly implicitly expressed inside the content item itself or available as metadata in some specific metadata vocabulary, which makes reengineering and refactoring essential tasks. When a content item is selected for presentation, i.e. based on the current situation, the presentation is orchestrated by the device I/O management module (cf. Section 7.1). Involving the speech communication module (cf. 7.4) also enables speech-based user interactions with content items when they have been presented to the user, e.g., when a user selects an event recommendation to start the booking of tickets. This interaction can in turn lead to situation switches, which are again managed by the software module for handling situations. Some user interactions, can also lead to an explicit situation change forced by the user, e.g. a voice command.

7.7 Helper Components

During the implementation of the AmI case system some helper components were added to ease the implementation work for all developers; the *AmI Case Tester* and the *AmI Case Status Monitor*. The *AmI Case Tester* is an optional OSGi component that enables developers to write and execute test cases. These test cases can be executed cascaded based on their internally described dependencies and can exchange test results with the subsequent test cases. The component was developed to enable a developer to execute test cases at runtime in the JVM of the OSGi environment. Similar to the *AmI Case Tester*, the *AmI Case Status Monitor* is an optional OSGi bundle and can directly be started with the AmI case system. When started the developer can see all messages broadcasted in the system as well as statistics of the concepts inside the knowledge representation, e.g., details about the users in the bathroom, content items, devices etc.

7.8 Installation

To ensure a consistent runtime environment as well as testing beneath all developers and the bathroom installation in Furtwangen, some specific features provided by Eclipse in combination with SVN were used:

- All components by all partners were shared using SVN
- The runtime configuration of the OSGi environment was also stored persistently and exchanged using SVN
- All external OSGi and non-OSGi dependencies were provided as target runtime environment and shared using SVN as well
- All configuration files of all components were stored at one place in one package to only have one location where a developer can adjust the configuration for his/her development machine or the real bathroom environment

8 Merging Software with Hardware

After designing and implementing the system architecture (cf. Section 6 & 7), the next step in T4.1 was merging the resulting software with the hardware. In the following, the selection of required devices (cf. Section 8.1), their integration and interconnection within the bathroom environment (cf. Section 8.2) and the connection of devices with the software layer (cf. Section 8.3) will be described. The result of all steps described in the following subsections – the bathroom prototype covering *Interactive Knowledge (IK)* points is shown in Fig. 35. We specified three IK points, each enabling (1) the recognition of users and (2) of user location; (3) the presentation of contents and (4) the interaction with content. The first IK point is the mirror (cf. Fig. 35; *IK point Mirror*) whereas the second IK-Point is realized by the room divider that represents a screen in the middle of the room (cf. Fig. 35; *IK point eScreen*). The third IK-Point is placed in the shower on the right hand side (cf. Fig. 35; *IK point Shower*).

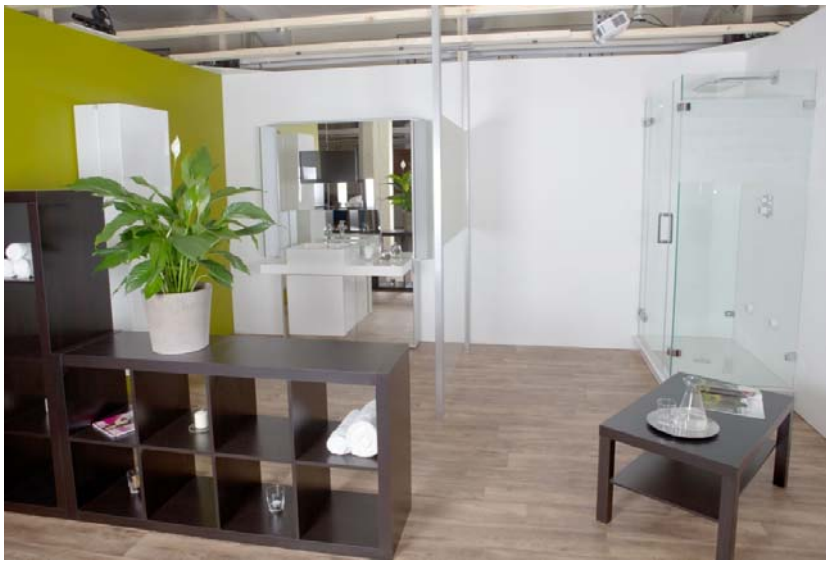


Fig. 35: Bathroom prototype with embedded devices realizing *Interactive Knowledge* points

8.1 Selection of Required Devices

Before selecting specific devices for the bathroom prototype, we specified eight issues in which we need devices in the intelligent environment: (1) Enabling speech communication, (2) enabling gestural communication, (3) detecting user position, (4) detecting situation in bathroom, (5) identifying user, (6) presenting visual content, (7) presenting audio content, and (8) giving ambient feedback. A detailed description of the issues concerning the current situation, context of interaction and user intentions as well as the related device-specific solution is shown in Tab. 3.

Tab. 3: Application of devices in the bathroom environment

Issue in the bathroom environment	Description of device-specific solution
(1) Enabling speech communication	The user can query the system by means of speech and the system can react with spoken feedback. For this purpose three array microphones were integrated positioned at IK points.
(2) Enabling gestural communication	The user can interact by gestures with the IK point Mirror. This is made up of a touch screen and a camera positioned in place of the bathroom mirror. The camera captures the scene in front of the mirror and projects it in the screen as background. The system can open presentation windows into the screen based on an AIR[43] software component.

[43] http://www.adobe.com/products/air/

(3) Detecting user position	The AmI system is able of detecting the position of the user in the bathroom. This is achieved by a mixture of sensor devices i.e., distance sensors positioned at IK points and a Kinect[44] device.
(4) Detecting situation in bathroom	The same devices used for user position detection will be useful for the situation detection.
(5) Identifying user	The AmI system will be able to identify the actual user. User identification will be performed by means of a Kinect device. This will be able to identify the user e.g. by his body height.
(6) Presenting visual content	The AmI system is able to present visual content. This is displayed depending on the situation and the position of the user on the IK points by means of a touchscreen and two projectors positioned in different locations.
(7) Presenting audio content	The AmI system is able to output audio content by means of an audio system with at least four loud speakers distributed in the room (two locations with stereo audio).
(8) Giving ambient feedback	The system is able to adapt the presentation of contents to the position of the user. This happens by adapting the lightening and the audio streams to the position of the user e.g., if the user is at IK point Mirror the lightening at this location will be more intensive and loudspeakers louder. The lightning is controlled by computer controlled electrical sockets.

Furthermore, the list of required devices for the specified issues was elaborated and assigned according to the input/output functionality of devices (cf. Tab. 4).

Tab. 4: Required devices for the specified issues assigned to input/output functionality

Devices	Input					Output				
	Enabling speech communication	*Enabling gestural communication*	*Detecting user position*	*Detecting situation in bathroom*	*Identifying user*	*Enabling speech communication*	*Enabling textual communication*	*Presenting visual content*	*Presenting audio content*	*Giving ambient feedback*
Microphones	**X**									
Audio system (Loudspeakers)						**X**			**X**	**X**
Projection screens at IK							**X**	**X**		**X**

[44] http://www.xbox.com/kinect

Devices	Input					Output				
	Enabling speech communication	Enabling gestural communication	Detecting user position	Detecting situation in bathroom	Identifying user	Enabling speech communication	Enabling textual communication	Presenting visual content	Presenting audio content	Giving ambient feedback
points Shower and eScreen (projection foil + projector)										
Touchscreen + camera at IK point Mirror		X					X	X	X	X
Touch sensors at IK points Shower and eScreen		X								
Kinect sensor					X					
Distance sensors			X	X						
Lightening										X

Based on Tab. 4, specific hardware was selected and ordered (cf. Tab. 5).

Tab. 5: Selected and integrated hardware

Device Type	Amount	Hardware
Microphones	3	AcousticMagic Voice Tracker II[45] array microphone
Audio system (Loudspeakers)	2	Logitech Z323 2.1[46] audio system
Projection screens at IK points Shower and eScreen (projection foil + projector)	2	Projection foil
	2	BenQ MP515ST[47] short distance projector
Touchscreen + camera at IK point Mirror	1	Acer T231Hbmid[48] 58cm 23" touchscreen display
	1	Logitech HD Pro Webcam C910[49] camera

[45] http://www.acousticmagic.com/voice-tracker-ii-array-microphone-product-details.html
[46] http://www.logitech.com/speakers-audio/home-pc-speakers/devices/5862
[47] http://benq.eu/products/Projector/index.cfm/product/1103
[48] http://www.acer.co.uk/ac/en/GB/content/model/ET.VT1HE.005
[49] http://www.logitech.com/en-us/webcam-communications/webcams/devices/6816

Touch sensors at IK points Shower and eScreen	4	Phidget touch sensors for PhidgetInterfaceKit
Kinect Sensor	1	Microsoft Kinect[50] sensor device
Distance Sensors	2	PhidgetInterfaceKit 8/8/8 (V2)[51] sensor board
	3	Phidgets distance sensors for PhidgetInterface-Kit
Lightening	1	Gembird Silvershield PMS[52] USB-controllable power supply
	4	Regular lights with standard 220V power supply

8.2 Integration of Devices in the Bathroom Environment

The integration of the aforementioned devices in the bathroom environment encompasses the positioning of the input and output devices in the bathroom as well as the setup of devices, i.e. the connections between the devices on hardware level.

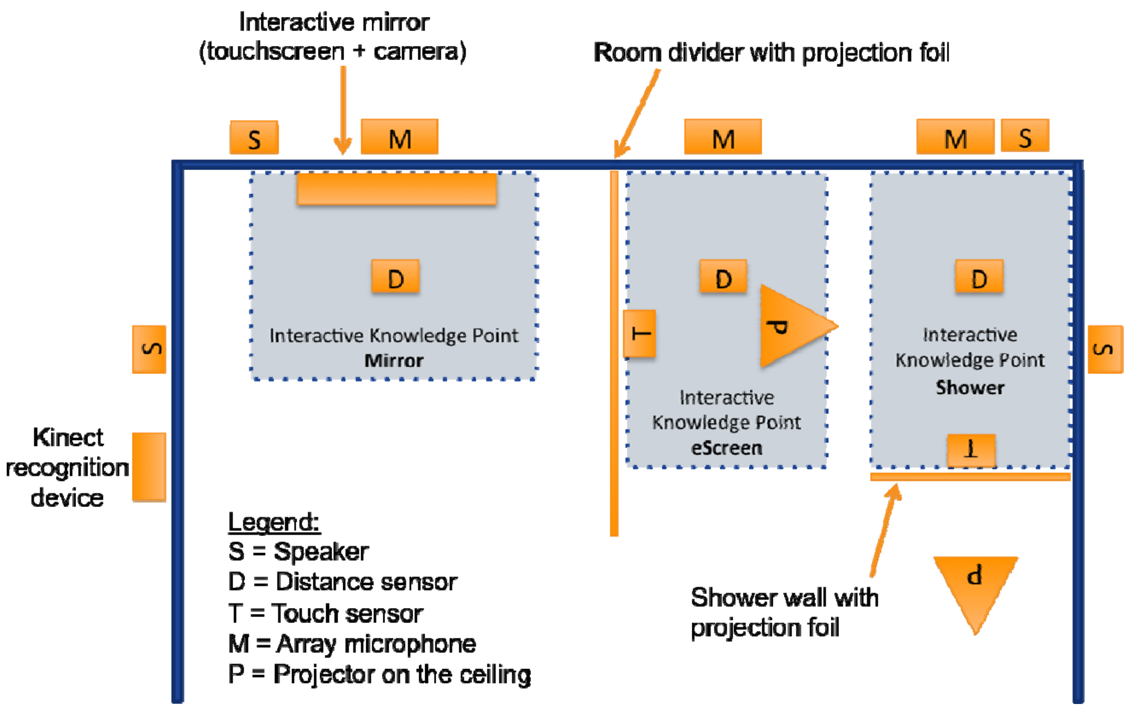

Interactive mirror (touchscreen + camera)
Room divider with projection foil
S
M
M
M
S
D
Interactive Knowledge Point
Mirror
D
P
T
Interactive Knowledge Point
eScreen
D
Interactive Knowledge Point
Shower
T
S
S
Kinect recognition device
Legend:
S = Speaker
D = Distance sensor
T = Touch sensor
M = Array microphone
P = Projector on the ceiling
Shower wall with projection foil
P

Fig. 36: Positions of Devices in the Bathroom Environment

The position of devices in the bathroom is visualized in Fig. 36. After integrating the devices in the bathroom, it was necessary to connect the devices in order to exchange data, e.g., sensor data, user input or content items presentations (cf. Fig. 37).

[50] http://www.xbox.com/kinect
[51] http://www.phidgets.com/products.php?category=0&product_id=1018
[52] http://www.gmb.nl/default.aspx?op=products&op2=item&id=3234

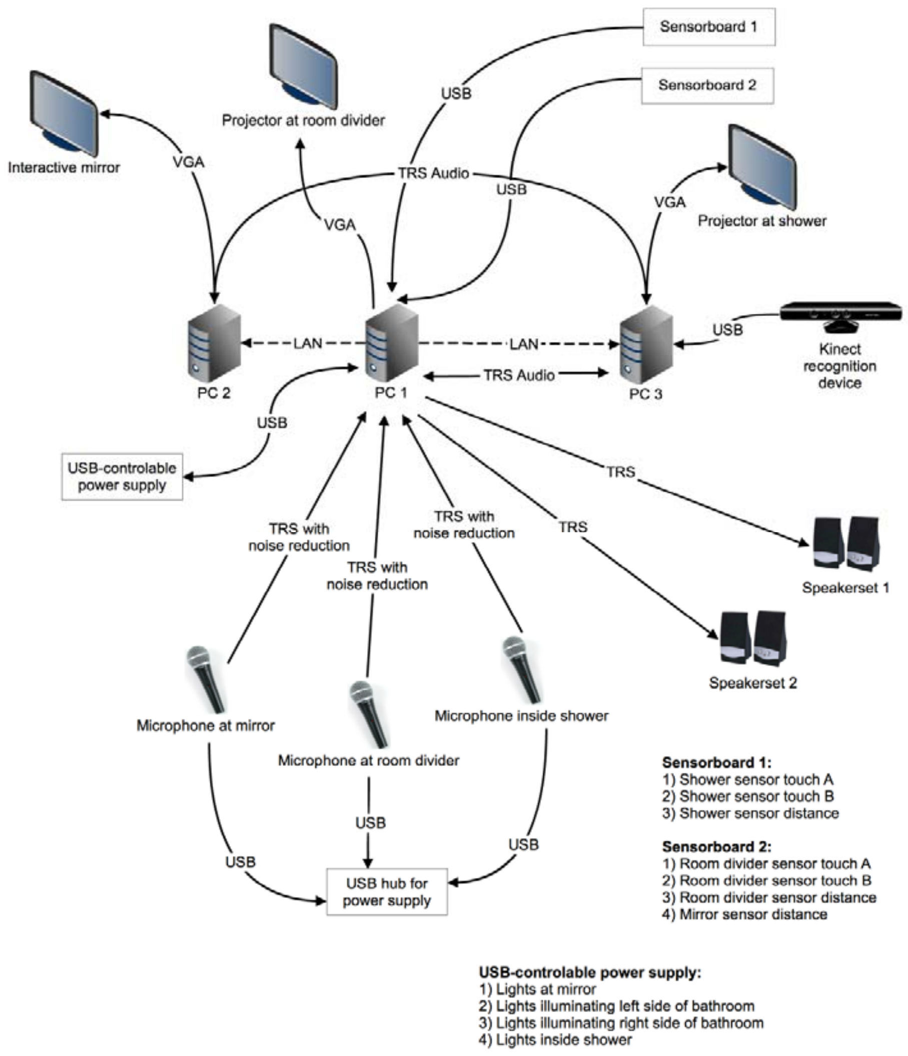


Fig. 37: Device connections within the bathroom environment

8.3 Intercommunication of Devices on the Software Layer

The step to merge the software with the hardware part was the addressing of devices by means of their drivers as well as the intercommunication of the software parts for each device. As described in Section 6 and 7, each device possesses an own OSGi module that is automatically loaded and integrated into the system when a device is broadcasted as being available in the bathroom or being removed. The OSGi service represents the interface to access more complex software parts in the background. The communication with additional software parts, which were developed in addition to the AmI case system itself, and the functions they achieve in the system will be described in the following. The AmI case system itself is running on *PC 1*.

Enabling speech communication. To interact with the system using natural language the Nuance Dragon Naturally Speaking[53] software is used in the AmI system, as well as the SVOX TTS[54] sofware for text-to-speech-based answers. These software components are installed on *PC 3* and are encapsulated in an additional Java application. For the communica-

[53] http://www.nuance.com/for-business/by-solution/dragon-enterprise/index.htm
[54] http://www.svox.com/Technology.aspx

tion between this Java application and the AmI system the MRCP[55] protocol and the SGRS[56] standard are used. This is performed by controlling the ASR and TTS components with a dedicated dialog server (SemVox ODP) which itself communicate with the rest of the system by means of the *Device Communication Adapter Component*.

Enabling gestural communication. Gestural communication in the bathroom environment is possible in two ways:

a. Using an AIR-based application installed on PC 2 (running the interactive mirror) the user can select presented content items on the touch screen. The unique URI of the selected content item is communicated to the AmI system using an HTTP request.
b. By touching hidden touch sensors directly below the presentation areas in IK point eScreen and Shower, the user can tell the system to swap content items. These touch sensors are attached to one of the sensorboards. Knowing the position of the sensors and by accessing the sensors using the SDK provided by the sensorboard, the AmI system can be informed at which interactive knowledge point the user wants to switch the contents.

Detecting user position. The user position is evaluated by several distance sensors, which are positioned at the ceiling or the walls of the bathroom within the IK points. Using the SDK provided with the sensorboard attached via USB, the values from the distance sensors are evaluated. To reduce the error rate, i.e. a user is recognized although he/she is not in front of the sensor, the sensor values are filtered to only inform the AmI system about a position change when the user really moved there.

Identifying user. A Kinect sensor performs the identification of the user within the bathroom. When a user enters the bathroom environment the person is recognized by the Kinect using the OpenNI[57] based drivers, which are encapsulated in an own .NET based application. This application uses a proprietary socket connection to communicate with the AmI system. The body height of the user in the bathroom and its position is continuously evaluated and communicated.

Presenting audiovisual content. All contents, except for the speech output already mentioned, are presented to the user using an AIR-based presentation client. This client is installed on all three PCs to enable the presentation of video, audio, text and other structured contents (e.g. weather information, events) on the interactive mirror as well as the room divider and the shower wall projection areas. Each AIR component can be accessed by its appropriate OSGi bundle using HTTP requests to display or remove content items.

Giving ambient feedback. The lighting scenario in the bathroom automatically adjusts to the users position. To switch on and off specific lights in the bathroom an USB-controllable power supply is used. Using system command calls performed by the device's OSGi bundle the power supply of the lights inside the bathroom environment can be switched on and off.

Summarising, communication between devices and the system is performed by means of their representation as OSGI bundles. In addition, while sensor devices are addressed via sensor specific SDKs, the presentation of audiovisual contents and gesture recognition in the context of IK point mirror are addressed via HTTP requests. Additionally, speech is controlled over MRCP protocols.

[55] http://tools.ietf.org/html/rfc4463
[56] http://www.w3.org/TR/speech-grammar/
[57] http://www.openni.org/

9 Evaluation of AmI Use Case

Consistent with SiDIS task 8 (cf. SiDIS design methodology in Fig. 1), we conducted a lab experiment, in which the Intelligent Bathroom environment was evaluated from an end-user perspective. In particular, the three situations 1, 6 and 11 as listed in Section 2 and its six information and communication services, i.e. (1) weather information service, (2) event recommendation service, (3) ticket order service, (4) personalized music service, (5) personalized news collage service, and (6) adaptive news service were tested. Hereby, Technology acceptance research as theoretical foundation was adopted. In the following sections, the method, results and discussion of the lab experiment are provided.

9.1 Method

The evaluation procedure consisted of four steps. First, subjects were given an introduction to the project and were briefly introduced in the use of the interactive bathroom environment with regard to a particular situation. Second, the subjects played through that situation twice to account for learning effects and to get more reliable evaluations afterwards. Third, each subject had to evaluate the situation and its embedded information and communication services by a questionnaire. The objective of this questionnaire was to get quantitative evaluations. Finally, interviews were conducted to get qualitative feedback, too. In general, the subjects were allowed to ask questions throughout the experiment. In addition, the supervisor supported the subjects in case a service did not work as expected (e.g., a pre-defined speech command was not recognized).

Due to the unbalanced situation-service distribution (e.g., Situation 1 was comprised of three services whereas Situation 3 just employed one service) and the corresponding unbalanced evaluation time, we decided that each subject had to evaluate three services. Subjects were therefore assigned to either Group A (Situation 1 – three services) or Group B (Situation 6 with two services and Situation 11 with one service) in advance of each experimental run. As a result, Situation 6 and 11 was combined and each subject required no more than 30 minutes for all three steps. The final descriptions of the situations for each group are provided in Tab. 6.

Tab. 6: Experimental design, assignment of groups, situations and services

Group	Situation	Textual Description	Service
A	1	It's Thursday morning. I get site-specific weather information when I am brushing my teeth in the bathroom. Based on weather information and my calendar, free-time event suggestions are given (e.g., Today, 8 p.m. – Sneak preview at CinemaOne). Do you want to order tickets?	Weather Information Event Recommendation Ticket Order
B	6, 11	Happily, it's Saturday morning. My partner takes a shower listening to his favorite music. Leaving the bathroom, I step into the room. I am welcomed by music from my own music collection. The music starts at the point in the playlist where I stopped listening the evening before. I stop the music by a hand gesture because I want to take a shower. While I take a shower my personal news collage is dis-	Personalized Music Personalized News Collage Adaptive News

Group	Situation	Textual Description	Service
		played. Having finished the shower, I dry myself with a towel and then brush my teeth. At the same time, I can take the news with me.	

The sampling procedure was as follows. We e-mailed an invitation to all students that are studying or had studied at the faculty of Digital Media at Furtwangen University. In this invitation, the experiment was promoted by claiming that each participant will receive 5 Euro. Further, it contained a link to a website, on which the students were able to register for the experiment by choosing one out of 55 time slots – each 30 minute slot intended for one student. After all slots were booked out, no one could register anymore. Therefore, only the first 55 subjects were allowed to participate and to receive the 5 Euro after the experiment. The spatial placement of the six information and communication services and the physical composition of the bathroom is shown in Fig. 38. In addition,

Tab. 7 provides an overview of the implemented input and output functionality for each of the six services.

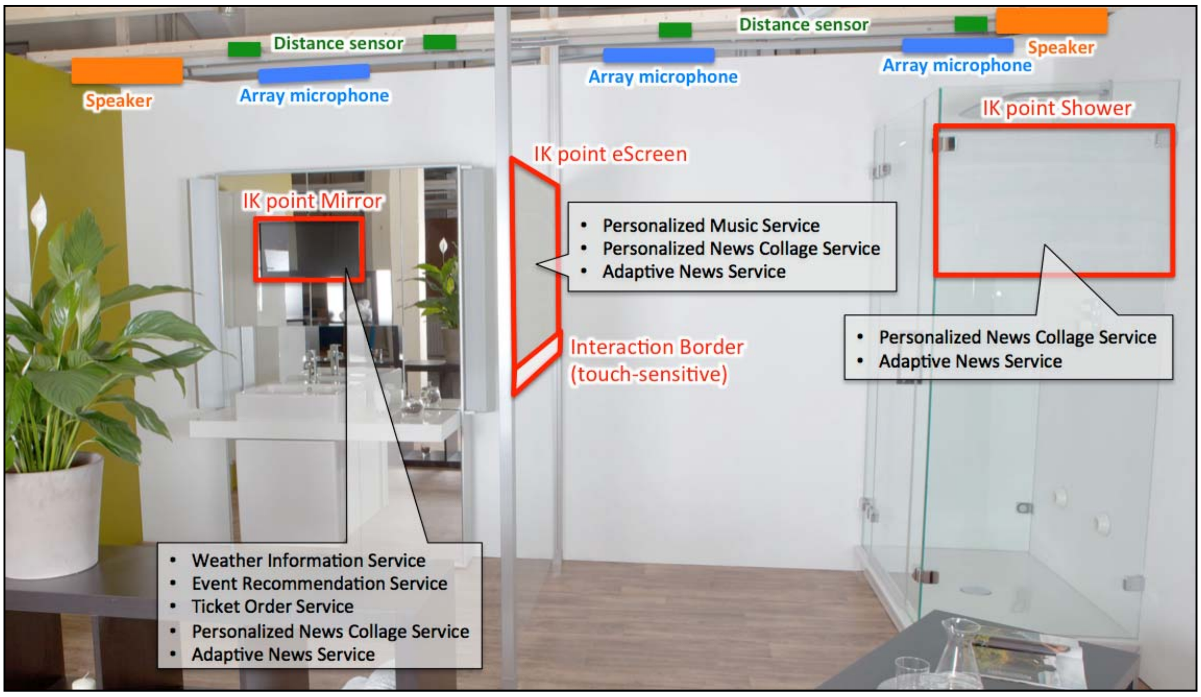


Fig. 38: Spatial placement of the six information and communication services

Tab. 7: Input and output functionality of the six information and communication services as implemented during the lab experiment (see Section 8 for technical details).

#	Service	Input	Output
1	Weather Information	Distance sensor – in front of the mirror	IK point Mirror: Today's weather information is displayed
2	Event Recommendation	Distance sensor – in front of the mirror	IK point Mirror: Three events are displayed

#	Service	Input	Output
3	Ticket Order	**Step 1:** Distance sensor – In front of the mirror and event recommendations are displayed **Step 2:** IK point Mirror: Ticket order request by touching an event **Step 3:** Array microphone: Acceptance or rejection of the ticket order request by voice command "yes" or "no".	Speaker: Verification question after Step 2: "Do you want to order tickets for the event …?"
4	Personalized Music	**Option 1:** Distance sensor – in front of the wall screen: Starts the playlist **Option 2:** IK point eScreen: wiping along the touch-sensitive interaction border stops the music	Speaker for the music and Wall Screen showing the current song of the playlist visually.
5	Personalized News Collage	IK point Shower: The following voice command starts the personalized news collage service: "Bathroom … [brief feedback tweet by the system via speaker] … what are the news?"	**Location A:** Subject stands in front of the IK point Mirror: the personalized news collage is displayed in form of text. Thus, only the mirror touchscreen is used. **Location B:** Subject stands in front of the IK point eScreen: the personalized news collage is displayed as a video clip. Thus, speaker and IK point eScreen present the content. **Location C:** Subject stands in front of the IK point Shower: the personalized news collage is displayed as a video clip. Thus, speaker and IK point Shower present the content.
6	Adaptive News	Distance sensor – in front of the IK points Mirror, eScreen or Shower.	See above, personalized news collage service. Depending on the location, the news adapt to the current location of the subject.

For the development of the questionnaire, we adopted Technology Acceptance constructs from Wixom and Todd (2005), Kamis et al. (2008) and Moore and Benbasat (1991). The constructs perceived usefulness, perceived ease of use, perceived enjoyment and behavioral intention to use were adapted from Wixom and Todd (2005) and Kamis et al (2008). Furthermore, we adapted the intention to purchase construct from Kamis et al. (2008) and derived the following three measures: (1) intention to pay for the information and communication service at (1) fixed costs, (2) based on a monthly subscription fee, and (3) based on a pay-per-use fee. We further used the compatibility construct of Rogers (2003) and Moore and Benbasat (1991) to measure the degree to which the services fit to the subjects' behavior and the situation, which also includes spatial placement and modality of the services. Before the questionnaire items were presented to the subjects, a concise definition of each service was provided as presented in D2.1, p. 21, Table 9). Among some demographic questions, we finally wanted to know how much the subjects would be willing to pay more for a bathroom that provides the corresponding service infrastructure. Consistent with prior work we used seven-point Likert scales ranging from very unlikely (1) to very likely (7) for the items related to purchase intentions and strongly disagree (1) to strongly agree (7) for all the others. The questionnaire items are shown in Tab. 8.

The interviews were finally conducted to ask for any problems the subjects had to deal with during the lab experiment. Among questions on the appropriateness of the spatial placement and modality of the various services, subjects were also asked for any improvements with regard to ease of use and missing functionality. Finally, we asked subjects to list information and communication services that they would find relevant in addition to those provided in the current implementation of the bathroom.

Tab. 8: Questionnaire items. Note: […] was replaced with the corresponding service.

No.	Construct and scale item wording
	Perceived enjoyment
PEN1	Using the […] service in the bathroom was fun.
PEN2	My attitude toward using the […] service in the bathroom is positive.
PEN3	Using the […] service was in the bathroom a pleasant experience.
PEN4	Using the […] service in the bathroom was very interesting.
	Perceived Usefulness
PU1	Using the […] service in the bathroom would increase my effectiveness.
PU2	Using the […] service in the bathroom would increase my efficiency.
PU3	Using the […] service in the bathroom would increase my overall performance.
PU4	I would find the […] service useful in the bathroom.
	Perceived ease of use
PEU1	Using the […] service was easy for me.
PEU2	My interaction with the […] service was clear and understandable.
PEU3	Learning to use the […] service was easy for me.
PEU4	It was easy for me to become skillful at using the […] service.
	Intention to use
IU	I would use the […] service in the bathroom.
	Intention to pay
IPOnce	If you actually had the money, how likely is it that you would pay for the […] service at a fixed price?
IPMonthly	If you actually had the money, how likely is it that you would pay a monthly fee for the […] service?
IPPerUse	If you actually had the money, how likely is it that you would pay a pay-per-use fee for the […] service?
	Fit of service and situation, behavior, modality and spatial placement
Situation-Fit	The […] service fits well to the situation I just played through.
Behavior-Fit	Using the […] service in the bathroom would fit well to my behavior.
Modality-Fit	I found that the modality of content presentation fits well to the […] service.
Spatial-Fit	I found that the spatial placement of the contents fits well to the […] service.

9.2 Results

Overall, 55 subjects participated in the lab experiment during the last week in June 2011. Group A (Situation 1, Services 1-3) was comprised of 10 male and 17 female subjects with ages ranging from 20 to 24 (n=19), 25 to 29 (n=4), and 30 to 34 (n=4). By contrast, Group B (Situations 6 and 11, Services 4-6) had 12 male and 16 female subjects with ages below 20 (n=1), from 20 to 24 (n=17), 25-29 (n=7), and 30-34 (n=3). The descriptive statistics are shown in Tab. 9 for each service.

In order to test the reliability of the measurement scales for perceived enjoyment, perceived usefulness and perceived ease of use, Cronbach's Alpha values were calculated. All values lie above the recommended threshold of .70 (Nunnally 1967) and thus, all scales for all six services can be said to be reliable. Accordingly, the aggregated scores, i.e. mean value and standard deviation have been calculated, too. These scores are also shown in Tab. 9.
The mean values of each construct have been further tested whether they lie significantly above or below the neutral scale value of 4 (neither). The significance of these one-sample t-tests are also provided in Tab. 9 and were indicated by the asterisk notation (* = p < .05 / ** = p < .01 / *** = p < .001).

Tab. 9: Descriptive statistics. Note: the following format is used: Mean (Standard deviation) * = p < .05 / ** = p < .01 / *** = p < .001, where the p-values are derived from one-sample t-test with a test value of 4.

	Weather Inform.	Event Recom.	Ticket Order	Person. Music	Pers. News Collage	Adaptive News
Item	Group A, n=27			Group B, n=28		
Perceived enjoyment						
PEN1	5.56 (1.19)	5.63 (1.04)	5.04 (1.63)	6.32 (0.77)	5.79 (1.10)	6.18 (0.77)
PEN2	6.00 (0.92)	5.19 (1.24)	4.59 (1.65)	6.36 (0.68)	5.36 (1.34)	6.04 (0.92)
PEN3	6.00 (1.00)	5.04 (1.16)	4.63 (1.57)	6.25 (0.70)	5.54 (1.20)	5.75 (1.00)
PEN4	5.56 (1.16)	5.59 (1.52)	5.22 (1.42)	6.11 (1.00)	5.86 (1.21)	6.18 (0.77)
PEN Alpha	.90	.91	.93	.74	.89	.90
PEN	**5.78*** (0.94)**	**5.36*** (1.02)**	**4.87** (1.42)**	**6.26*** (0.60)**	**5.63*** (1.06)**	**6.04*** (0.76)**
Perceived usefulness						
PU1	6.33 (0.78)	4.56 (1.70)	4.00 (1.57)	5.71 (1.36)	4.57 (1.57)	4.71 (1.78)
PU2	5.81 (1.27)	5.07 (1.44)	4.56 (1.58)	5.32 (1.28)	4.71 (1.78)	4.68 (1.68)
PU3	4.63 (1.45)	4.37 (1.76)	3.59 (1.60)	5.14 (1.51)	4.00 (1.63)	4.39 (1.64)
PU4	6.30 (0.91)	5.07 (1.00)	4.41 (1.47)	5.82 (1.39)	5.00 (1.56)	5.11 (1.55)
PU Alpha	.84	.87	.93	.91	.91	.92
PU	**5.77*** (0.93)**	**4.77** (1.27)**	**4.14 (1.41)**	**5.50*** (1.23)**	**4.57* (1.46)**	**4.72* (1.49)**
Perceived ease of use						
PEU1	6.26 (1.13)	5.93 (1.21)	4.89 (1.74)	6.25 (1.08)	5.00 (1.66)	6.39 (0.74)
PEU2	6.19 (1.11)	5.52 (1.19)	4.96 (1.53)	5.86 (1.15)	5.43 (1.29)	6.29 (0.94)
PEU3	6.26 (0.94)	6.11 (0.70)	5.89 (0.97)	6.50 (0.75)	6.11 (0.92)	6.61 (0.63)
PEU4	6.15 (0.86)	5.70 (1.03)	5.15 (1.73)	6.39 (0.69)	5.68 (1.39)	6.32 (0.95)
PEU Alpha	.86	.82	.91	.87	.90	.89
PEU	**6.21*** (0.86)**	**5.81*** (0.85)**	**5.22*** (1.36)**	**6.25*** (0.79)**	**5.55*** (1.17)**	**6.40*** (0.71)**
Intention to use						
IU	6.33*** (0.73)	4.67* (1.52)	3.78 (1.74)	6.75*** (0.52)	4.86* (1.78)	5.14** (1.56)
Intention to pay						
IPOnce	5.07** (1.62)	4.41 (1.89)	3.56 (1.74)	5.89*** (1.34)	4.50 (1.80)	4.86** (1.53)
IPMonthly	2.93** (1.69)	2.74*** (1.51)	2.19*** (1.24)	3.00** (1.47)	2.57*** (1.73)	2.32*** (1.47)
IPPerUse	2.11*** (1.70)	2.44*** (1.89)	2.74** (2.14)	1.93*** (1.46)	2.04*** (1.48)	1.82*** (1.44)
Fit of service and situation, behavior, modality and spatial placement						
Situation-Fit	6.19*** (0.96)	5.52*** (1.01)	5.19*** (1.06)	6.29*** (0.94)	5.64*** (1.31)	5.79*** (1.07)
Behavior-Fit	6.07*** (1.17)	4.48 (1.76)	3.93 (1.71)	6.54*** (0.79)	4.86** (1.63)	5.11** (1.64)
Modality-Fit	5.15*** (1.38)	5.04** (1.40)	4.74** (1.23)	5.25*** (1.53)	5.79*** (1.40)	5.36*** (1.50)
Spatial-Fit	5.41*** (1.60)	4.85** (1.49)	4.63* (1.52)	5.96*** (0.74)	5.68*** (1.16)	5.96*** (1.04)

Results with regard to the amount how much subjects would be willing to pay more for a bathroom that provides the corresponding service infrastructure are depicted in Fig. 39. Even though the mean values of Group A (18.89%, standard deviation = 23.77) and Group B (26.36%, standard deviation = 29.46) differ, these differences are not significant by applying ANOVA (t-value = 1.04; p = .31). Accordingly subjects would pay 22,8 percent of the costs of a traditional bathroom on average more for a service-enabled bathroom environment.

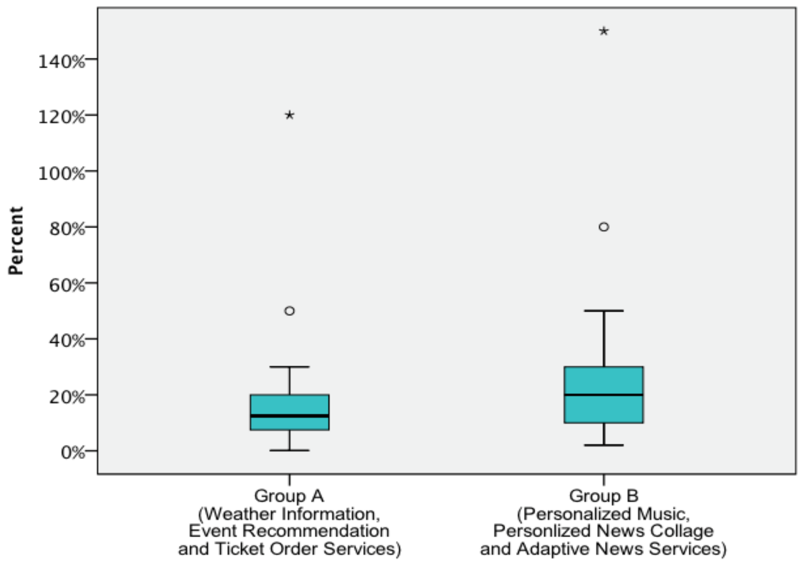


Fig. 39: Subjects' willingness to pay more in % for a bathroom that provides the corresponding service infrastructure.

In addition to the quantitative data of the questionnaire-based results above, the subjects' feedback related to the six information and communication services and general remarks from the interviews was categorized into four topics: (1) interaction, (2) spatial placement, (3) presentation quality, and (4) missing functionality. Tab. 12 summarizes this feedback and can be found in the Appendix.
As a final result of the interviews, the frequency-ranked list of additional information and communication services relevant to the subjects of the lab experiment is provided in Tab. 10.

Tab. 10: Additional information and communication services relevant to the subjects

Rank	Service	Frequency
1	Date and time service	22 / 40,0%
2	Social network service (e.g., Facebook, Google+, etc.)	19 / 34,5%
3	eMail service	18 / 32,7%
	Calendar service (e.g., Google Calendar, iCal, etc.)	
5	Movie service (e.g., Netflix, iTunes movie rentals, etc.)	13 / 23,6%
	Radio service	
7	Television service (e.g., Zattoo Live TV, etc.)	12 / 21,8%
8	Twitter stream service	10 / 18,2%
9	Telecommunication service (e.g., Telephone, Skype, etc.)	4 / 7,3 %
10	Other services such as home automation (e.g., light controls, etc.), music clips, podcasts, messenger services, product recommendations	>= 2 / 3,6%

9.3 Discussion

In this paragraph, the results of the lab experiment are discussed and suggestions for an improvement of the bathroom environment are provided. First of all and according to the results of the descriptive statistics provided in Tab. 9, subjects found that all services were perceived enjoyable and fun to use. Only the ticket order service was perceived less enjoyable in comparison to the others (mean = 4.87 at the .01 significance level). This may indicate that a ticket ordering process involves a business transaction, in which rational mental processing is required and this rational factor may suppress perceived enjoyment to some degree.

Second, weather information service and personalized music service were perceived as most useful, followed by the event recommendation service, personalized news collage service and adaptive news service. Only the ticket order service was not significantly perceived as useful as the other services. By considering the qualitative feedback of Tab. 12, it might be the case that missing information on available seat categories, missing ticket prices for these categories and more information about events such as movie trailers represents a reason for these evaluations.

Third, all six services were easy to use because all aggregated ease of use scores lie significantly above the mean values at the highest level of significance ($p < .001$). Ease of use of the adaptive news service was rated highest, which can be explained by its implicit interaction, i.e. subjects had only to move around to take the news with them and thus, no explicit interaction such as a voice command or a touching gesture was required.

Fourth, subjects intend to use predominantly the weather information service and the personalized music service in the bathroom. These results are consistent with the usefulness evaluations. Moreover, the behavioral intentions to use the ticket order service have been rated negatively, however, the negative ratings are not significant. The reasons for these rather negative ratings are discussed above but further hints for improving the ticket order service can be derived directly from Tab. 12.

Fifth, result regarding the intention to pay measures can be used to derive a corresponding revenue model for each of the six services. Correspondingly and throughout all services, subjects were neither likely to pay a monthly fee nor to adopt a pay-per-use model. By contrast, subjects would rather pay, if at all, one-time a fixed price. In particular, the personalized music service, the weather information service and the adaptive news service should be promoted with a one-time / fixed price revenue model. The remaining services may be promoted and offered to potential customers as part of a service bundle. Accordingly, subjects were willing to pay 22.8 % of the price of a traditional bathroom on top for a corresponding service infrastructure. Therefore, the services might also be offered together with that service structure.

Sixth, regarding the fit of service with its offered modality and spatial placement, the bathroom situation and subject's behavior, all services performed well overall resulting in high fit scores. Only the event recommendation service and ticket order service do not fit very well to the subjects' behavior in the bathroom. One reason may lie in the current design of the implementation; another reason may be the fact that event recommendations and related ticket purchase decisions do not fit into the morning rituals of the subjects.

Eighth, Tab. 12 gives a good overview of feedback and recommendations that can be used for the improvement of the current bath environment. In particular, feedback that was given multiple times (as indicated by the frequency value in brackets) should be taken seriously into account for a revision of the current implementation. Just to give one example, the content of the mirror-based-serves should not be displayed in the center of the mirror but rather in the periphery (23 subjects had given this feedback).

Ninth, Tab. 10 can be used to select and implement additional information and communication services that are also relevant to the subjects of this lab experiment.
Finally, it must be stated that the results and implications of this experiment are based on a sample of students with a strong technical background. Thus, these results are biased in the sense that technology-savvy subjects have participated. Nonetheless, those subjects are al-

so potential customers of bath furniture companies, and in particular of those companies that will content-enhanced interactive bathroom environments as being tested within this IKS Ami Use Case.

10 Re-Iteration of AmI Use Case

In the re-iteration phase of the AmI Use Case, the objective was to validate the components developed in the context of IKS reference implementation. At first, we aligned the reference architecture for a semantic content management system (SCMS) with the logical architecture (c.f. 6) of the AmI case system to evaluate, which logical components could be reused on which architectural layers. Afterwards we evaluated all components of the IKS reference implementation regarding their benefit for the realization of the identified bathroom situations (cf. Section 2). Next, components with suitable functionality were integrated and validated.

10.1 Alignment of SCMS Architecture

When comparing the SCMS reference architecture developed in IKS with the logical architecture developed within this task, several modules on several architectural layers can be aligned. This alignment can be seen in Fig. 40. All blue marked modules are modules from the logical AmI case architecture, which can also be found in the SCMS reference architecture.

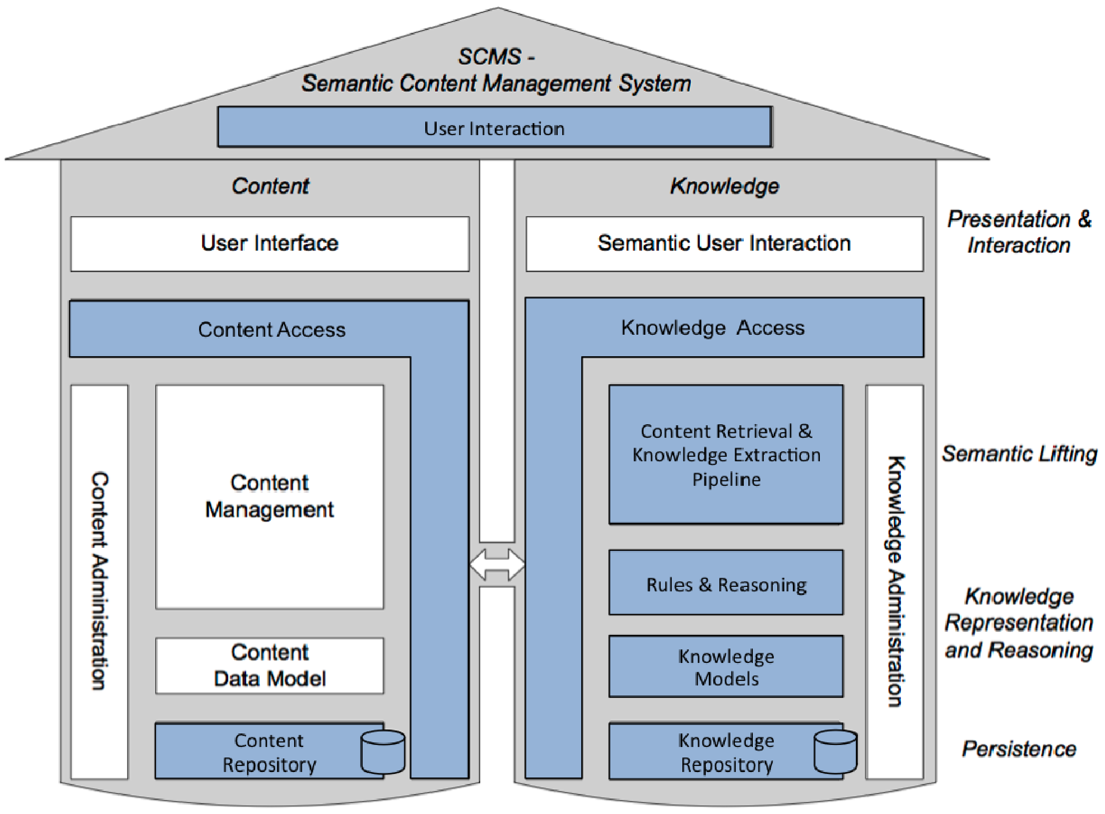


Fig. 40: Alignment of Logical AmI Case Modules with the SCMS Reference Architecture

While several modules can be aligned because they provide a comparable functionality, some other modules only exists in the SCMS architecture as well as some modules only exist in the AmI case architecture. This comes from the different specializations of the systems for the architectures were developed for. While a UIS like the AmI case system requires an

ambient management layer, the content management systems realized on the SCMS reference architecture require knowledge and content administration functionalities, but not vice versa. This emerges from the fact that the AmI case, as a use-case for IKS, has a slightly different focus, to also test the components in another research field than the classical content management and interaction.
In the following the components realized as reference implementation for the reference architecture for SCMSs are listed and evaluated regarding their reuse in the AmI case system.

10.2 Evaluation of IKS Components regarding the AmI Case

In this section already available components that are developed in the scope of IKS reference implementation will be examined.
Almost the entire software product developed in the scope of IKS reference implementation belongs to Stanbol Project[58], which is an incubating project under the umbrella of Apache Software Foundation. Besides, there are VIE[59] and VIE² [60] components that provide interaction capabilities with the user.

At first, the components of Apache Stanbol will be examined:

- **Stanbol CMS Adapter:** This component provides extracting the already available semantics in the content model of a content repository as ontology. The ontology extraction process is done by defining bridge definitions. It is possible to define bridges only for content repositories that support JCR or CMIS specifications as they allow a standard way to access to content in the repository.

- **Stanbol Content Hub:** This is a quite recent component, which has a quite preliminary implementation. It basically provides management of submitted contents together with the enhancements obtained by the Enhancer Component, which will be explained below.

- **Stanbol Enhancer:** This stateless interface allows the caller to submit content to the Apache Stanbol enhancer engines and get the resulting enhancements formatted as RDF. It mainly provides people, location and organization entities from the text and tries to further enhance those entities by making use of external linked data sources.

- **Stanbol Entity Hub:** This component provides the connection to external linked open data sites as well as using indexes of them locally. Its services allow managing a network of sites to consume entity information and to manage entities locally.

- **Stanbol Fact Store:** This component provides storage and retrieval of relations between the entities mentioned in the Entity Hub component.

- **Stanbol Ontology Manager:** Stanbol Ontology Manager provides CRUD (Create, Update, Delete) functionalities on a set of ontologies or a single ontology. It also provides storage and retrieval of ontologies to/from a knowledge base behind. In addition, it allows the construction and management of sub-networks of ontologies extracted from the entire knowledge base.

[58] http://incubator.apache.org/stanbol/
[59] http://github.com/IKS/VIE
[60] http://github.com/IKS/VIE-2

- **Stanbol Reasoner:** This component provides reasoning capabilities, both by wrapping off-the-shelf DL reasoners and by providing an endpoint for accessing eternal reasoners via the OWLlink protocol.
- **Stanbol Reengineer:** This component provides transformation from XML data and relational databases to RDF.
- **Stanbol Rules:** This component provides functionalities for rule management and execution, including the concatenation of rules into recipes and the ordered execution of recipes on ontologies.

Besides Apache Stanbol components, there are two interaction components:

- **VIE:** VIE is the access point to editable content on your pages. It parses RDFa annotations on a page and makes them accessible as JavaScript objects backed by Backbone.js models, views and collections.
- **VIE^2:** VIE^2 is the semantic enrichment layer on top of VIE. Through it you can query and find related content for your editables. VIE^2 can talk to services like Apache Stanbol and OpenCalais to find related information for your content.

In the following table (cf. Tab. 11) the discussion results for the integration of the specific IKS component within the AmI case are summarized. The AmI case developers elaborated the list after consultation with the appropriate IKS reference architecture developers.

Tab. 11: Evaluation of components of IKS Reference Implementation from an AmI use case perspective

IKS reference implementation component	**Equivalent component in the AmI case system**	**Form of integration**	**Expected advantages of integration and integration discussion**
Stanbol CMS Adapter	SCEM, which has similar, but proprietary functionality	The component is used by SCEM using REST interface	• Contents (i.e. weather information) from other IKS-capable CMS system can be integrated in the AmI Case System • Will be integrated to demonstrate the the content item retrieval from other CMS (e.g. Adobe CRX) • Meet the requirements of situation 1 (cf. Section 2)
Stanbol Content Hub	none	The component will not be integrated (see next column)	The component is in an early stage of development and can't currently be integrated
Stanbol Enhancer	none	The component is used by VIE^2 using REST interface	• Will be integrated to be able to present the user hyperlink text news instead of plain text news • Will be used to extract organization, place and location entities from the news texts and provide extracted enhancements • Meet the requirements of situation 1, 6, 11 (cf. Section 2)

IKS reference implementation component	**Equivalent component in the AmI case system**	**Form of integration**	**Expected advantages of integration and integration discussion**
Stanbol Entity Hub	none	The component is used by SCEM using REST interface	• Will be integrated for querying external linked data sets for further entities related with ones extracted by the Enhancer. • Expected advantages are a higher speed and better accuracy when retrieving and refactoring content items • Meet the requirements of situation 1, 6, 11 (cf. Section 2)
Stanbol Fact Store	none	The component will not be integrated (see next column)	• The component is in an early state of development and can't currently be integrated • The idea behind to create n-triples over statements would also be interesting for the knowledge management in the AmI case system
Stanbol Ontology Manager	TNT, which currently only stores in files and SCEM	The component will be integrated as storage layer below TNT and for some ontological tasks in SCEM	• Expected advantage are that knowledge models managed within the AmI case system can also be accessed by other RESTful applications, e.g. for external enhancement • Stored ontologies can also be accessed in the web browser • Stanbol Ontology Manager Store: Enables persistent storage of TNT data in a more structured way (cf. previously mentioned features) than in OWL files • Stanbol Ontology Manager OntoNet is used by the SCEM Refactorer & Reengineer components • Meet the requirements of situation 1, 6, 11 (cf. Section 2)
Stanbol Reasoners	TNT	The component will not be integrated (see next column)	• The functionality is not required by the AmI case system • In the integration discussion no reason to integrate the component could be found; TNT also already provides this functionality, which is also not used in the system, since the semantic rule-based reasoning provided by the JENA framework is enough for the required functionality • Furthermore the component can currently not be integrated because it is currently completely refurbished because of licence issues
Stanbol Reengineer	SCEM	The component is a replacement for KReS	Has already been integrated in a former version in KReS and will be integrated to bring the stability enhancements of the new version to the AmI case
Stanbol Rules	SCEM	The component is a replacement for KReS	Has already been integrated in a former version in KReS and will be integrated to bring the stability enhancements of the new version to the AmI case

IKS reference implementation component	Equivalent component in the AmI case system	Form of integration	Expected advantages of integration and integration discussion
IKS VIE2	AIR-based media player for content item presentation, which has less functionality	The component will be integrated into the Adobe AIR based presentation component as interactive website	Will be integrated to enable the user to interact with some content items (e.g. text news) in some new ways: • Accessing meta informations (e.g. information about the places mentioned in a news article) • Browsing contents (e.g. possibility to read not only the current news article but also articles similar to the same) • Meet the requirements of situation 1 (cf. Section 2)

10.3 Validation of IKS Components

In this section, there will be a detailed explanation about the process of the IKS reference implementation components that we integrated into the AmI system in the re-iteration phase. We will explain how those components are integrated into the system and what problems we faced.
An instance of Apache Stanbol running in a separate Java Virtual Machine is used to make use of most of the components developed in the IKS reference implementation. We access services of those components through their RESTful services. For example the Stanbol CMS Adapter, Stanbol Enhancer and Stanbol Ontology Manager Store are accessed through their RESTful services. For the Stanbol Rules, Stanbol Reengineer and Stanbol Ontology Manager Ontonet, we have added separate libraries into the AmI system to be able use these components.
In the following subsections, the integration process of each component will be explained in detail.

10.3.1 Stanbol CMS Adapter

In the re-iteration phase, the first component we integrated into the system is the CMS Adapter. In the system, it is used to extract semantics of content model and the actual content within the content management systems. In this way, we achieved data retrieval from different types of services including content repositories supporting JCR or CMIS specifications and non-semantic sources complying with the UIS guidelines determined previously. We used the CRX content management system of Day/Adobe Software holding the external weather information data. First, we created node types having the necessary structure representing the weather information we used in the system. We defined the following node types:

- Location
- ShortDescription
- TimeInterval
- WeatherForecast
- WeatherInformation

Then we define bridges to map content repository to an ontology. We extract properties having semantic value of content repository objects which have one of the defined types above. This semantic lifting process is one of the services that Stanbol Project provides for the sake of a semantic CMS. Because extracting such an ontology based on the content model of repository would allow semantic functionalities on top of it. Such kind of a semantic lifting is also complying with the UIS guidelines, which state to offer semantic lifting capabilities for contents. Example bridge definitions to get necessary information from content repository are like the following ones:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<BridgeDefinitions
        xmlns="mapping.model.servicesapi.cmsadapter.stanbol.apache.org">
        <InstanceBridge>
                <Query>/WeatherInformations/WeatherLocation</Query>
                <PropertyBridge>
                        <PredicateName>name</PredicateName>
                </PropertyBridge>
        </InstanceBridge>
        <InstanceBridge>
                <Query>/WeatherInformations/Day1/WeatherInfo1</Query>
                <PropertyBridge>
                        <PredicateName>shortPrediction</PredicateName>
                </PropertyBridge>
                <PropertyBridge>
                        <PredicateName>highTemperature</PredicateName>
                </PropertyBridge>
                <PropertyBridge>
                        <PredicateName>lowTemperature</PredicateName>
                </PropertyBridge>
                <PropertyBridge>
                        <PredicateName>actualTemperature</PredicateName>
                </PropertyBridge>
                <PropertyBridge>
                        <PredicateName>location</PredicateName>
                </PropertyBridge>
                <PropertyBridge>
                        <PredicateName>timeInterval</PredicateName>
                </PropertyBridge>
                <PropertyBridge>
                        <PredicateName>isRealizedBy</PredicateName>
                </PropertyBridge>
        </InstanceBridge>
        <InstanceBridge>
                <Query>/WeatherInformations/Day1/WeatherForecast1</Query>
                <PropertyBridge>
                        <PredicateName>realizes</PredicateName>
                </PropertyBridge>
                <PropertyBridge>
                        <PredicateName>forecastString</PredicateName>
                </PropertyBridge>
        </InstanceBridge>
        <InstanceBridge>
                <Query>/WeatherInformations/Day1/TimeInterval1</Query>
                <PropertyBridge>
                        <PredicateName>startDate</PredicateName>
                </PropertyBridge>
                <PropertyBridge>
                        <PredicateName>endDate</PredicateName>
                </PropertyBridge>
        </InstanceBridge>
        <InstanceBridge>
                <Query>/WeatherInformations/Day1/ShortDescription1</Query>
                <PropertyBridge>
                        <PredicateName>description</PredicateName>
                </PropertyBridge>
        </InstanceBridge>
</BridgeDefinitions>
```

Then further rules are executed on the extracted ontology to get actual content items that are used in the system. The rule which is used to get actual weather forecast content items from the content repository can be seen below:

```
    <owl:Thing rdf:about="http://amicase.iks-
project.eu/meta/refactoring_rules.owl#weather_cmsadapter_rule">
        <rdf:type rdf:resource="&rmi;KReSRule"/>
        <rdf:type rdf:resource="&owl;NamedIndividual"/>
        <rmi:hasDescription rdf:datatype="&xsd;string">If there is an XML element called
&lt;forecast&gt; and a subelement &lt;location&gt;, then the first element is a WeatherInfor-
mation, and the second relates to the first via the forecastForPlace prop-
erty.</rmi:hasDescription>
        <rmi:hasBodyAndHead rdf:datatype="&xsd;string">amireeng = &lt;http://amicase.iks-
project.eu/content/reengineer#&gt; .
odp_weather =
&lt;http://www.ontologydesignpatterns.org/iks/ami/2011/02/WeatherForecast.owl#&gt; .
odp_place = &lt;http://www.ontologydesignpatterns.org/iks/ami/2011/02/Location.owl#&gt; .
odp_content =
&amp;lt;http://www.ontologydesignpatterns.org/iks/ami/2011/02/Content.owl#&amp;gt; .
odp_creal =
&amp;lt;http://www.ontologydesignpatterns.org/iks/ami/2011/02/ContentRealizations.owl#&amp;gt;
.
odp_conf =&lt;http://im.dm.hs-furtwangen.de/ontologies/ami-case/context-
configuration.owl#&gt;.
odp_ti =  &amp;lt;http://www.ontologydesignpatterns.org/cp/owl/timeinterval.owl#&amp;gt; .
rdf = &lt;http://www.w3.org/1999/02/22-rdf-syntax-ns#&gt; .
rdfs = &lt;http://www.w3.org/2000/01/rdf-schema#&gt; .
wo = &lt;http://weatherInformation#&gt; .
info = &lt;http://www.ontologydesignpatterns.org/cp/owl/informationrealization.owl#&gt; .

wflr[ has(rdf:type, ?weatherInfo, wo:class-weatherInformation) .
values(wo:dprop-lowTemperature, ?weatherInfo, ?lowTemperature).
values(wo:dprop-actualTemperature, ?weatherInfo, ?actualTemperature).
values(wo:dprop-highTemperature, ?weatherInfo, ?highTemperature).
has(rdf:type, ?location, wo:class-location).
values(wo:dprop-name, ?location, ?locationName).
has(wo:oprop-isRealizedBy, ?weatherInfo, ?forecast).
values(wo:dprop-forecastString, ?forecast, ?forecastString).
has(wo:oprop-shortPrediction, ?weatherInfo, ?shortPrediction).
values(wo:dprop-description, ?shortPrediction, ?shortPredictionValue).
has(wo:oprop-timeInterval, ?weatherInfo, ?timeInterval).
values(wo:dprop-startDate, ?timeInterval, ?startDate).
values(wo:dprop-endDate, ?timeInterval, ?endDate).
-&gt;
is(odp_weather:WeatherInformation, ?weatherInfo) .
values(odp_weather:actualTemperature, ?weatherInfo, ?actualTemperature).
values(odp_weather:forecastPredictsLowTemperature, ?weatherInfo, ?lowTemperature).
values(odp_weather:forecastPredictsHighTemperature, ?weatherInfo, ?highTemperature).
is(odp_place:City, ?location).
has(odp_weather:forecastForPlace, ?weatherInfo, ?location).
values(rdfs:label, ?location, ?locationName).
is(odp_weather:WeatherForecast, ?forecast).
has(info:isRealizedBy, ?weatherInfo, ?forecast).
has(info:realizes, ?forecast, ?weatherInfo).
has(odp_content:hasContentType, ?forecast, odp_weather:weather).
has(odp_creal:isOfForm, ?forecast, odp_conf:StructuredContent).
values(odp_weather:weatherForecastString, ?forecast,?forecastString).
is(odp_weather:WeatherType, ?shortPrediction).
values(wo:dprop-description, ?shortPrediction, ?shortPredictionValue).
has(odp_weather:forecastPredictsWeather, ?weatherInfo, ?shortPrediction).
is(odp_ti:TimeInterval, ?timeInterval).
has(odp_weather:forecastValidDuringTime, ?weatherInfo, ?timeInterval).
has(odp_ti:hasIntervalStartDate, ?timeInterval, ?startDate).
has(odp_ti:hasIntervalEndDate, ?timeInterval, ?endDate)]</rmi:hasBodyAndHead>
    </owl:Thing>
```

This rule has a special syntax which is handled by the Rules component of Apache Stanbol.

10.3.2 Stanbol Enhancer

As explained in Section 7.5 Semantic Content Extractor Module, news content items are obtained through different external services. After adapting the retrieved content is presented to the user as plain text.
To present a semantically rich content to the user we have used the Stanbol Enhancer. It basically runs several enhancement engines on a text input. We fed the Stanbol Enhancer with the text content of retrieved news items from external services. Enhancement engines first extract the person, location and organization entities from the text by using natural language processing. It then uses other enhancement engines to get further enhancements based on

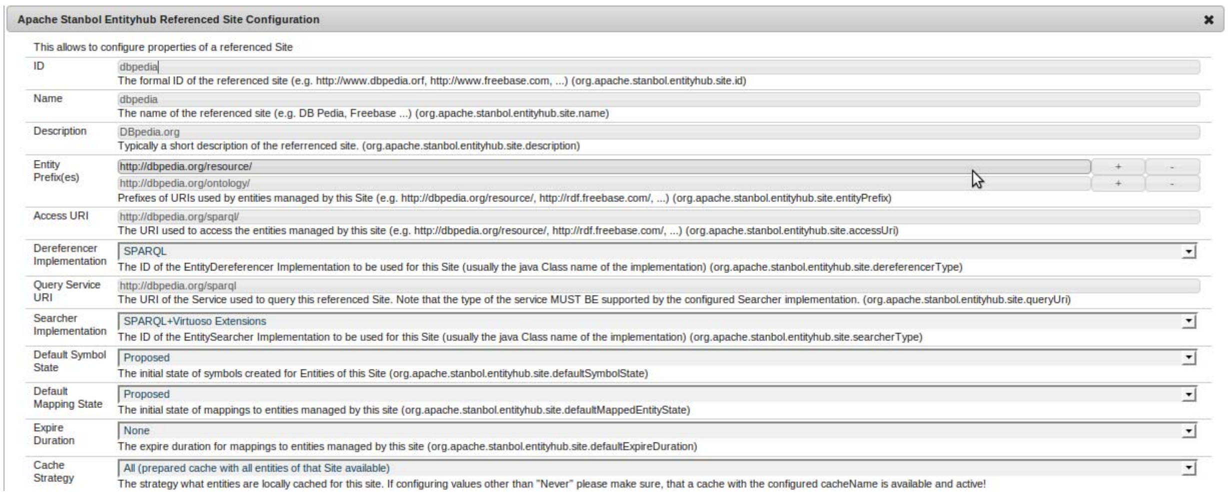
the ones extracted by natural language processing. An example enhancement extracted after natural language processing is shown in the following:

```
  <rdf:Description rdf:about="urn:enhancement-95388e0b-77d7-8ea4-b862-2f1d252a9632">
    <rdf:type rdf:resource="http://fise.iks-project.eu/ontology/Enhancement"/>
    <j.1:extracted-from rdf:resource="urn:content-item-sha1-
a5157ebee0aa633c85762fa961b04f4397ea2ee4"/>
    <j.0:created rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2011-08-
24T14:59:37.448Z</j.0:created>
    <j.0:creator
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">org.apache.stanbol.enhancer.engines.ope
nnlp.impl.NEREngineCore</j.0:creator>
    <rdf:type rdf:resource="http://fise.iks-project.eu/ontology/TextAnnotation"/>
    <j.1:selected-text
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Liverpool</j.1:selected-text>
    <j.1:selection-context rdf:datatype="http://www.w3.org/2001/XMLSchema#string">e squad.
Meanwhile, Wenger is insisting to any media member who will listen that he is satisified with
the squad he has put ogether. As one who watched it create few scoring opportunities at New-
castle and two in 90 minutes at home to Liverpool, I&amp;#8217;ll use the words of my old
mates in Ireland: &amp;#8220;He has seen them play, the &amp;#8221;  Last week&amp;#8217;s
match against Udinese had a Theo Walcott goal within four</j.1:selection-context>
    <j.0:type rdf:resource="http://dbpedia.org/ontology/Place"/>
    <j.1:confidence
rdf:datatype="http://www.w3.org/2001/XMLSchema#double">0.5268414589329835</j.1:confidence>
    <j.1:start rdf:datatype="http://www.w3.org/2001/XMLSchema#int">2771</j.1:start>
    <j.1:end rdf:datatype="http://www.w3.org/2001/XMLSchema#int">2780</j.1:end>
  </rdf:Description>
```

It is very easy to use this Stanbol Enhancer. The only necessary thing to do is submitting the text content to its RESTful services. It returns several enhancements like the one above regarding to the submitted content in RDF format. This retrieved enhancements in RDF will be used by the VIE^2 component to be used for a richer graphical user interface.

10.3.3 Stanbol Entityhub

The Entityhub component is used to query external linked data sites for getting further entities about the ones extracted from the Stanbol Enhancer component.
This component enables configuration of any number of external linked data sites. In the AmI system, it is used with its default configuration. The default configuration allows DBPedia queries for already existing entities related with a given keyword. Fig. 41 shows such configuration to query DBPedia.


Apache Stanbol Entityhub Referenced Site Configuration

This allows to configure properties of a referenced Site

ID	dbpedia The formal ID of the referenced site (e.g. http://www.dbpedia.orf, http://www.freebase.com, ...) (org.apache.stanbol.entityhub.site.id)
Name	dbpedia The name of the referenced site (e.g. DB Pedia, Freebase ...) (org.apache.stanbol.entityhub.site.name)
Description	DBpedia.org Typically a short description of the referrenced site. (org.apache.stanbol.entityhub.site.description)
Entity Prefix(es)	http://dbpedia.org/resource/ http://dbpedia.org/ontology/ Prefixes of URIs used by entities managed by this Site (e.g. http://dbpedia.org/resource/, http://rdf.freebase.com/, ...) (org.apache.stanbol.entityhub.site.entityPrefix)
Access URI	http://dbpedia.org/sparql/ The URI used to access the entities managed by this site (e.g. http://dbpedia.org/resource/, http://rdf.freebase.com/, ...) (org.apache.stanbol.entityhub.site.accessUri)
Dereferencer Implementation	SPARQL The ID of the EntityDereferencer Implementation to be used for this Site (usually the java Class name of the implementation) (org.apache.stanbol.entityhub.site.dereferencerType)
Query Service URI	http://dbpedia.org/sparql The URI of the Service used to query this referenced Site. Note that the type of the service MUST BE supported by the configured Searcher implementation. (org.apache.stanbol.entityhub.site.queryUri)
Searcher Implementation	SPARQL+Virtuoso Extensions The ID of the EntitySearcher Implementation to be used for this Site (usually the java Class name of the implementation) (org.apache.stanbol.entityhub.site.searcherType)
Default Symbol State	Proposed The initial state of symbols created for Entities of this Site (org.apache.stanbol.entityhub.site.defaultSymbolState)
Default Mapping State	Proposed The initial state of mappings to entities managed by this site (org.apache.stanbol.entityhub.site.defaultMappedEntityState)
Expire Duration	None The expire duration for mappings to entities managed by this site (org.apache.stanbol.entityhub.site.defaultExpireDuration)
Cache Strategy	All (prepared cache with all entities of that Site available) The strategy what entities are locally cached for this site. If configuring values other than "Never" please make sure, that a cache with the configured cacheName is available and active!

Fig. 41: Default Entityhub configurations to query DBPedia

The Stanbol Enhancer component has an enhancement engine making use of the Entityhub. That enhancement engine sends a set of requests containing the entities extracted after natural language processing to the Entityhub so that the Entityhub can query configured external sites for related entities about the given input and further enhancements like the one below are obtained for the initial text e.g news article in our case.

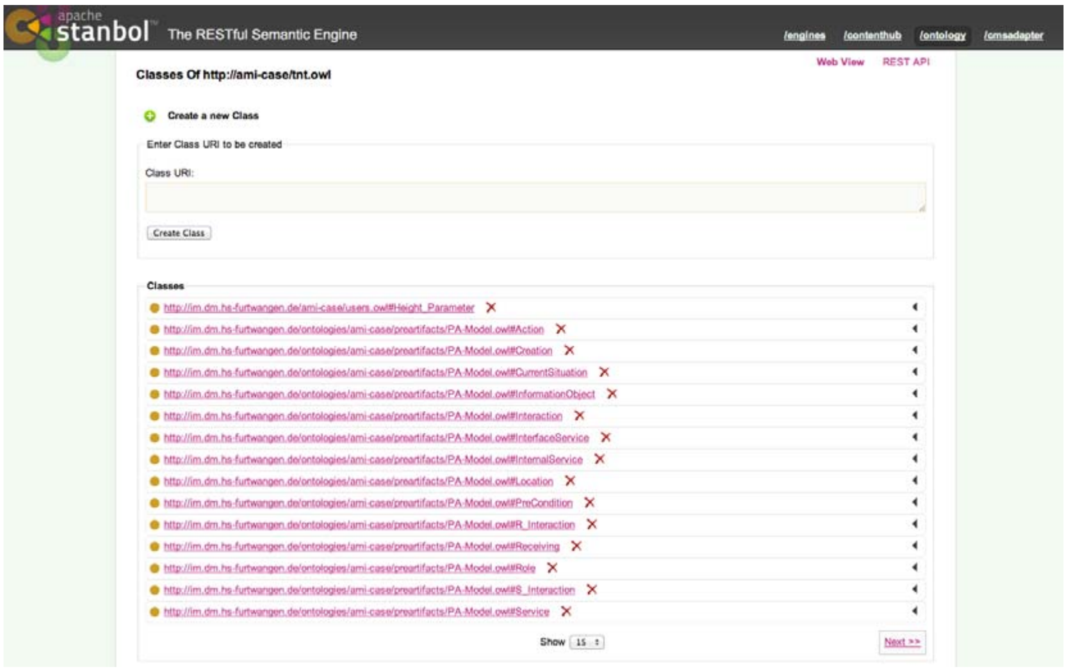
```
  <rdf:Description rdf:about="urn:enhancement-876e6f00-2e0a-3459-7828-0cec66b73a4b">
    <j.1:entity-type rdf:resource="http://dbpedia.org/ontology/Settlement"/>
    <j.1:entity-type rdf:resource="http://dbpedia.org/ontology/PopulatedPlace"/>
    <j.1:entity-type rdf:resource="http://dbpedia.org/ontology/Place"/>
    <j.1:entity-type rdf:resource="http://www.w3.org/2002/07/owl#Thing"/>
    <j.1:confidence
rdf:datatype="http://www.w3.org/2001/XMLSchema#double">5.5256953</j.1:confidence>
    <j.1:entity-label xml:lang="en">Liverpool, New South Wales</j.1:entity-label>
    <j.1:entity-reference
rdf:resource="http://dbpedia.org/resource/Liverpool,_New_South_Wales"/>
    <j.0:relation rdf:resource="urn:enhancement-95388e0b-77d7-8ea4-b862-2f1d252a9632"/>
    <rdf:type rdf:resource="http://fise.iks-project.eu/ontology/EntityAnnotation"/>
    <j.0:creator
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">org.apache.stanbol.enhancer.engines.ent
itytagging.impl.NamedEntityTaggingEngine</j.0:creator>
    <j.0:created rdf:datatype="http://www.w3.org/2001/XMLSchema#dateTime">2011-08-
24T14:59:37.627Z</j.0:created>
    <j.1:extracted-from rdf:resource="urn:content-item-sha1-
a5157ebee0aa633c85762fa961b04f4397ea2ee4"/>
    <rdf:type rdf:resource="http://fise.iks-project.eu/ontology/Enhancement"/>
  </rdf:Description>
```

The first enhancement example presented in Section 10.3.2 Stanbol Enhancer was about **"Liverpool"** entity. We can see that from its **"selected-text"** property. Entityhub brought us another entity related with Liverpool which is **"Liverpool, New South Wales"**.

10.3.4 Stanbol Ontology Manager

The Stanbol Ontology Manager was integrated in two components of the AmI case system On the one hand it's implicitly used in the SCEM Refactorer and Reengineer components, on the other hand it's used as new persistent storage layer below and for the TnT2 system.
Regarding the first integration the Stanbol Ontology Manager is used by SCEM because SCEM uses the Stanbol CMS Adapter, which again uses the Stanbol Ontology Manager. This integration required no further adaption from the AmI case perspective.
The second integration into the TnT2 system required the adjustment of the TnT2 storage system. Before the integration TnT2 stored the situational parts of the knowledge models as OWL-based dumps into files. This process was automatically triggered after a specific count of modifications and after a specific time period passed by. The disadvantage of this method was, that it was not possible to have a look at the stored statements except for accessing the storage file using an external semantic editor like Protégé[61]. To integrate the Stanbol Ontology Manager the REST interface of the component was used. To access the REST interface using Java the Jersey client libraries[62] were used to push or pull the ontological representations to the Stanbol Ontology Managers knowledge store.
The integration of the component was quite easy, since the only thing that needed to be adjusted was the hostname and port of the REST interface. After storing the knowledge models for the first time we were able to browse and check the ontological structures using the Stanbol Ontology Manager web interface (see Fig. 42). This made it more transparent to understand how the stored knowledge parts are constructed at runtime.

[61] http://protege.stanford.edu/
[62] http://jersey.java.net/



Classes Of http://ami-case/tnt.owl

Create a new Class

Enter Class URI to be created

Class URI:

Create Class

Classes

- http://im.dm.hs-furtwangen.de/ami-case/users.owl#Height_Parameter
- http://im.dm.hs-furtwangen.de/ontologies/ami-case/preartifacts/PA-Model.owl#Action
- http://im.dm.hs-furtwangen.de/ontologies/ami-case/preartifacts/PA-Model.owl#Creation
- http://im.dm.hs-furtwangen.de/ontologies/ami-case/preartifacts/PA-Model.owl#CurrentSituation
- http://im.dm.hs-furtwangen.de/ontologies/ami-case/preartifacts/PA-Model.owl#InformationObject
- http://im.dm.hs-furtwangen.de/ontologies/ami-case/preartifacts/PA-Model.owl#Interaction
- http://im.dm.hs-furtwangen.de/ontologies/ami-case/preartifacts/PA-Model.owl#InterfaceService
- http://im.dm.hs-furtwangen.de/ontologies/ami-case/preartifacts/PA-Model.owl#InternalService
- http://im.dm.hs-furtwangen.de/ontologies/ami-case/preartifacts/PA-Model.owl#Location
- http://im.dm.hs-furtwangen.de/ontologies/ami-case/preartifacts/PA-Model.owl#PreCondition
- http://im.dm.hs-furtwangen.de/ontologies/ami-case/preartifacts/PA-Model.owl#R_Interaction
- http://im.dm.hs-furtwangen.de/ontologies/ami-case/preartifacts/PA-Model.owl#Receiving
- http://im.dm.hs-furtwangen.de/ontologies/ami-case/preartifacts/PA-Model.owl#Role
- http://im.dm.hs-furtwangen.de/ontologies/ami-case/preartifacts/PA-Model.owl#S_Interaction
- http://im.dm.hs-furtwangen.de/ontologies/ami-case/preartifacts/PA-Model.owl#Service

Show 15

Next >>

Fig. 42: Exemplary View on Ontological Classes of the AmI Case Knowledge Model in the Stanbol Ontology Manager's Web Interface

Another sub-module of the Stanbol Ontology Manager is OntoNet, short for the Ontology Network Manager. OntoNet (artifact ID `org.apache.stanbol.ontologymanager.ontonet`) is the component previously labeled `eu.iksproject.kres.ontology`, and it is responsible for managing a controlled environment where single parts of the SCMS knowledge base can be activated and reasoned upon. This component, while not explicitly invoked by newly included Stanbol modules, is used by both the Stanbol Reengineer and Rules modules, e.g. for the execution of inference rules on the required ontologies only. It was therefore preserved in its newer incarnation during the re-iteration phase.
Because the latest OntoNet has a transitive dependency on Jersey libraries for RESTful Web Service support, this was detected as another potential cause for version conflict. This issue would affect the *Content Aggregator* and other AmI Case bundles that were required to invoke RESTful services and therefore depend on a Jersey client API. As with the Xerces case, the existence of a single version of the Jersey libraries compatible with all AmI modules was ensured by having the "*AmI Case Commons*" bundle export version 1.7 and all other Jersey instances removed from bundle classpaths and the default OSGi configuration.

10.3.5 Stanbol Reengineer

One of the Alpha Stack components to be already included in the AmI implementation prior to the reiteration phase is the Reengineer. As an Alpha Stack component, the reengineer consisted of the `eu.iksproject.kres.semion.reengineer` bundle, its dependencies and extensions for XML and relational databases. It has since been replaced with the newer `org.apache.stanbol.reengineer.base` bundle extended with `org.apache.stanbol.reengineer.xml`.
Although the refactoring process for Apache Stanbol broke apart the KReS RESTful interface monolithic module into multiple modules, one per component, we did not include these modules in the demonstrator and opted for accessing the Reengineer Java API instead. This allowed us to preserve the original implementation of the SCEM Reengineer service interface,

and to avoid the overhead of RESTful API calls and the addition of further dependencies on REST client libraries.
The main changes since the IKS Alpha version are merely structural rather than functional, and allowed a slimmer architecture and a less dense dependency graph. These changes include: (*i*) relaxation of non-essential dependencies, such as the removal of KReS Reasoners (now Stanbol Reasoners), which can now be excluded from the AmI environment without loss of functionality; (*ii*) simplified API for the Reengineer Manager, which now handles only reengineer implementations and no longer registers a refactorer (hence eliminating dependencies on the Rules and Refactor modules); (*iii*) KReS bundles for the *Xerces* and MySQL libraries are no longer required; (iv) the DB Reengineer is no longer registered, as no external data source used in the AmI Case exposes its relational database.
During the reiteration phase, we had to take heed of a possible side effect of the reduced bindings and dependencies of the new Stanbol modules. When refactoring Stanbol Reengineer, we eliminated the strict dependency on a custom wrapper bundle for the Xerces XML parser and opted for stating a dependency on Xerces Java packages version 2.7.1. Although this strategy allowed more flexibility with OSGi modules providing Xerces packages, it was found to cause version conflicts for the Jena TDB bundle version 0.3, which alone satisfies several Stanbol dependencies but includes the incompatible `xercesImpl` version 2.9.1. This issue was coped with by excluding the Jena TDB bundle from the OSGi environment, and replacing it with an "*AmI Case Commons*" bundle. This, among others, re-exports a "safe" version of the Jena libraries and dependencies, including Xerces 2.7.1.

10.3.6 Stanbol Rules

As with Stanbol Reengineer, the now obsolete `eu.iksproject.kres.semion.refactorer` bundle, which has already been integrated from IKS Alpha prior to the re-iteration phase, was replaced with the `org.apache.stanbol.rules.*` set of artifacts, which now incorporate refactoring functionalities.
Along with greater stability, reduced dependencies and a simpler API, now independent from both the Reasoners and the Reengineer APIs, Stanbol Rules provides functional enhancement in the form of additional support for rule language built-in predicates and atoms. These now include comparison, string manipulation, arithmetic atoms and production rules. As with Stanbol Reengineer, the Rules functionalities are accessed via the Java API instead of the RESTful one, in accordance with the previous iteration of the AmI implementation.

10.3.7 VIE^2

The VIE^2 Component from IKS Alpha has been integrated in the AmI case in order to demonstrate capability of IKS components to enhance the interaction capabilities with additional functionalities offered by enhanced content. The component has been exemplary used for the presentation of news content which is then semantically enhanced with VIE^2. The enhancement is then used to highlight entities on which additional information can be acquired. By clicking on the given entity additional information from DBpedia is presented to the user.
The VIE^2 component has been integrated in the system by modifying the Device Access Component (eu.iksproject.amicase.device_access_component) by adding functionality for browser based presentation. For each news presentation an HTML5 page with VIE^2 integration is dynamically produced which is than presented on a standard browser started from the Device Access Component itself.
Content enhancement in VIE^2 is performed by a direct call of the IKS Stanbol component which is addressed using dedicated VIE^2 Connectors. The integration of the component did not need any installation of additional packages in the OSGi environment because the VIE^2 core, which is all jscript coded could be loaded from a foreign server

(http://iks.dfki.de/showcase/) by simply linking it in the dynamically generated HTML code. In the same way a remote Stanbol enhancer (http://dev.iks-project.eu:8080/) as well as a local one could be accessed. Thanks to the flexibility offered by the HTML5/Jscript implementation of this IKS component the integration resulted in an easy procedure. The enhancement showed how a more interactive and flexible presentation of content inside the bathroom environment could be reached by adding the VIE^2 Interaction and Presentation component.

11 Discussion

The experience of applying the *Interactive Knowledge Stack* and Semantic Web technologies in general to the development of the intelligent bathroom use case allowed us to learn potential benefits of this approach to the design of Ubiquitous Information Systems as well as research challenges to be investigated in future work. In the following subsections, we will discuss our experiences and afterwards lessons learned from a number of different perspectives. According to the methodical procedure by SiDIS (cf. Section 2), we will start with the design of the knowledge representation for the UIS (cf. Section 4). Then, the role of conceptual models of the SiDIS conceptual modelling framework (cf. Section 4 and 5) in requirements engineering will be discussed. Furthermore, we will deal with our experiences in implementing the use case by means of IKS modules (cf. Section 10). These issues are followed by a discussion of the results of the user study (cf. Section 9).

11.1 Design of Knowledge Representation for UIS by means of IKS

In Section 4, we introduced an approach for designing semantic knowledge representations for UIS based on the combination of two methodological approaches: XD (Presutti et al., 2009) and SiDIS (Maass & Janzen, 2011). We described our proceeding when developing the knowledge representation consisting of the contextual and situational part (cf. SiDIS task 5) as well as their processing by the resulting UIS within the real-life context - the intelligent bathroom use case.
When modelling the situational part of the knowledge representation, we detected that the formalization of propositional CMs as OWL ontologies is sometimes challenging. OWL provides advantages through its logical expressivity, nevertheless it provides neither guidelines nor formal rules for mapping domain information to OWL axioms addressing a specific task, e.g., narrative and diagrammatic CMs to formalized propositional CMs (Bera et al., 2010; Maass & Janzen, 2011). In other words, the same narrative and diagrammatic CMs resulted in potentially different and incompatible formalizations as propositional CMs depending on the designer. This was a challenge when different propositional CMs, formalized by different designers, had to be used consistently.
In T4.1, we have addressed this problem by applying a pattern-based approach combined with AISM (Maass & Janzen, 2011), as described in Section 4.1. The use of the AIS ontology provided designers with a reference for identifying design solutions that would result in a consistent formal model. The designers used the expressiveness of the patterns to develop propositional CMs. We detected that this approach provides structure and guide designers without loss of too much freedom in modeling the situational part of the knowledge representation. This problem has already a general characterization in ontology engineering: methods for mapping non-ontological resources to ontologies (Nuzzolese et al., 2011), and is a fundamental motivation for Ontology Design Patterns (Gangemi, 2009) as used in the contextual part of the knowledge representation.

11.2 What System Designers Want …

Within the first quarter of T4.1, we detected that the requirements analysis of the AmI use case had some incomplete and missing requirements. So, we completed the analysis and added new requirements (cf. Appendix A). Nonetheless, the requirements did not fulfil their function within the development of the AmI use case as considered. The numerousness of requirements operating on a detailed level gave only minimal support for designing the architecture and later implementing the prototype. Thinking in usage situations as needed in real-life environment was not possible based on requirements, not to even think about of an automatic usability of the requirements within the development of the architecture or the system itself (cf. SiDIS task 6).

This problem is well known in software engineering; requirements engineers insist on the natural language representation of requirements to ensure that all stakeholders are "on board". Contrary to these needs, system designers demand an efficient automatic usability of knowledge generated during requirements engineering (Dubois, 1986). When developing propositional CMs as situational part of the knowledge representation for the intelligent bathroom based on the SiDIS conceptual modelling framework, we detected that this integrated framework can be applied within the whole Requirements Engineering (RE) process (Castaneda et al., 2010). We started with a natural language representation of requirements (cf. narrative CMs) that later was transformed into propositional CMs to be used by system designers, as well as by the UIS itself, in an efficient way. So, we gained traceable software artifacts, and requirements by means of semantic technologies that could be reused within diverse RE activities, and supported the overall problem solving to a greater extent than single, isolated conceptualizations (Brewster, 2004; Riechert, 2007; Lin, 1996; Veres, 2009). Currently, there are diverse fields of potential usage of semantic technologies, more precisely semantic knowledge representations, within RE, for instance describing requirements documents (Decker, 2005; Groza, 2007; Dragoni, 2010) or formally representing requirements (Dobson, 2006; Padilla, 2008), amongst others (Castaneda et al., 2010). However, semantic knowledge representation within RE seems to be focusing on the representation of requirements of future systems exclusively.

As described by our approach above, we assume that semantic knowledge representations could be used within the whole cycle of RE activities consisting of elicitation, representation, analysis and communication of requirements with the aforementioned advantages.

11.3 Implementing UIS by means of IKS

After specifying the requirements and developing the knowledge representation for the UIS, the system was implemented in a first version (cf. Section 7) and then re-iterated according to available IKS modules (cf. SiDIS task 7 and Section 10).

Comparing the Semantic CMS Reference Architecture (Christ & Nagel, 2011) with the logical architecture of the AmI system (cf. Section 6), most of the layers and modules were aligned. Due to the specific character of the AmI use case, there were modules that were beyond the scope of the use case, e.g., Knowledge Administration, or additional in the context of T4.1, e.g., an Ambient Management layer. In the following, we will discuss the application of the following IKS components within the AmI case: *Stanbol CMS Adapter, Stanbol Enhancer, Stanbol Entity Hub, Stanbol Ontology Manager* and *VIE^2*.

Stanbol Reengineer and *Stanbol Rules* were already integrated in the AmI system as a former version of KReS and will therefore not be discussed in this section. The *Stanbol Content Hub* could not be validated because of its early stage of development similar to the *Stanbol Fact Store*. Furthermore, *Stanbol Reasoner* was not integrated into the AmI system because its functionality was not required by the use case because the semantic rule-based reasoning provided by the JENA framework is sufficient.

Before addressing the specific aspects of the application of the IKS components it has to be remarked that the integration of the additional components lead to a performance decrease of the whole AmI case system. The startup of the system as well as the reactions on user inputs slowed down significantly. From our perspective this mainly comes from the increase of complexity of the new added components as well as from the simple aspect that the system now handles more components and dependencies as before. Also the new dependencies to external software, which has to be started in addition to the AmI case system, e.g. the IKS-capable CMSs, leads to a general performance reduction of the system. A suggested resolution for this problem could be the merging of all OSGi components off all separate systems into one Java runtime environment and with this the switch from heterogeneous interfaces like RESTful interfaces to less resource expensive interfaces like services on the OSGi level.

Stanbol CMS Adapter enables the extraction of already available semantics in the content model of a content repository. Through the integration of this component in the UIS, contents, e.g., weather information from IKS-capable CMSs (e.g. Day/Adobe CRX) should be retrieved. With the successful integration, we were able to retrieve data from different types of services including content repositories supporting JCR or CMIS specifications and non semantic sources according to the UIS guidelines determined before (cf. Section 3.1). We used the Day/Adobe CRX CMS to store external data, e.g., weather information.
Concerning the effort, we had to create specific node types first to represent the data structure we used internally, e.g., Location, ShortDescription, WeatherForecast. Then, bridges were defined to map the contents of the content repository to an ontology in the sense of a semantic lifting process. Afterwards, actual content items were retrieved and integrated in the ontology in a rule-based way.
Regarding the requirements of situation 1 (cf. Section 2) to gather actual contents from the web in a dynamic way, we state that Stanbol CMS Adapter cannot meet this requirement caused by a laborious and non-dynamic way of integrating new sources from the web. Ideally, the system should be able to identify useful and trusted information sources, gather new data from them, transform them to RDF/OWL, and employ them for some specific purpose, e.g., giving information about theatre events tonight. However, the current approach has shown to still be time-consuming when new information sources have to be added against the need of the UIS for spatially and temporally unrestricted access to information on the web. But, a clear requirement based on the aforementioned experience would be the development of methods and semantic representations that allow dynamic interoperability between an UIS and heterogeneous sources on the web.

Stanbol Enhancer generates content enhancements formatted in RDF regarding existing content items. The focus lies on the extraction of people, locations and organization entities from the content item and the enhancement of those entities by means of external linked data sources. Stanbol Enhancer was integrated to meet the requirements of situation 1, 6 and 11 in a better way that means to enable the presentation of hyperlinked text news instead of plain text news concerning organizations, location and people entities. The Stanbol Enhancer was fed with news contents retrieved from external services. Then, person, location and organization entities were extracted from the text using natural language processing technologies and the related enhancement RDFs were generated.
The integration of Stanbol Enhancer was quite easy caused by the available RESTful interface that was used to submit the text item that had to be enhanced. Then, several enhancements are returned in RDF format. Stanbol Enhancer represents a middleware component because the enhancements in RDF are used by the VIE^2 component that will be discussed later.

Stanbol Entity Hub is used to query external linked data sites for getting further entities related to the extracted entities of the aforementioned Stanbol Enhancer. Stanbol Entity Hub enables the configuration of any number of external linked data sites. In the AmI use case, it

is used with its default configurations that means querying DBpedia for already existing entities related to a given keyword.
Stanbol Entity Hub represents a middleware component because it supports Stanbol Enhancer used by the VIE^2 component that will be discussed later. Nonetheless, the manual and non-dynamic configuration of specific external linked data sites disagrees with the aforementioned need for methods and semantic representations that allow dynamic interoperability between an UIS and heterogeneous sources on the web.

Stanbol Ontology Manager provides CRUD (Create, Update, Delete) functionalities on a set of ontologies or a single ontology. It also provides storage and retrieval of ontologies. Furthermore, it allows the construction and management of sub-networks of ontologies extracted from the entire knowledge base. In the AmI use case, Stanbol Ontology Manager is used (1) by the Stanbol CMS Adapter and (2) as new persistent storage layer below and for the TnT2 system. Before the integration TnT2 stored the situational parts of the knowledge models as OWL-based dumps into files; this covered the disadvantage that it was not possible to have a look at the stored statements. By integrating Stanbol Ontology Manager, we could dispose this problem.
The integration of Stanbol Ontology Manager was quite easy by using the REST interface. After storing the knowledge models for the first time we were able to browse and check the ontological structures using the Stanbol Ontology Manager web interface. This made it more transparent to understand how the stored knowledge parts are constructed at runtime.

VIE^2 parses RDFa annotations on a page and makes them accessible as JavaScript objects. It represents a semantic enrichment layer that enables querying and adding related content items to enhance existing content. Within the AmI use case, VIE^2 was integrated to enhance content items by adding meta information (e.g. places mentioned in the content item, similar content items) automatically. This aspect should improve the presentation of contents to the user by offering meta information (cf. situation 1 (cf. Section 2)).
The enhancement by VIE^2 highlight entities that “offer” additional information. By drag and drop the given entity to the “DBpedia search” widget additional information from DBpedia is presented to the user. This widget is represented in form of a HTML5 page that is dynamically produced for each news item by means of the aforementioned supporting Stanbol components. The integration of VIE^2 is realized by loading VIE^2 core (JavaScript) from an external server by means of a link in the dynamically generated HTML pages.
Concerning the requirements of the AmI use case (cf. situation 1, 6 and 11 (cf. Section 2)) and the specific characteristic of an UIS, the browser paradigm of VIE^2 does not present a real improvement in the bathroom. We assume that an automatic and thoughtful offering of meta information, e.g., additional pictures related to news or events, would be more reasonable in the AmI context instead of an drag & drop interaction with a small browser window.

11.4 Usage of IKS-supported UIS by Potential Users

A user study with potential early adopters was carried out as part of SiDIS task 8 “Evaluation of Solution”. The study had the primary objective to get direct feedback from users interacting with the IKS-supported UIS on six concrete content-centred information and communication services: weather information service, event recommendation service, ticket order service, personalized music service, personalized news collage service, and adaptive news service. Hereby, perceived characteristics of the UIS were in the focus of investigation such as perceived enjoyment, usefulness and ease of use. Moreover, the fit of those services with user behaviour, spatial placement, modality and pre-defined bathroom situations was tested.
Findings indicate that all six information and communication services have been designed by the AmI team of IKS in such a way that they fit into every-day bathroom situations. Thus, the use of natural language representations of requirements, i.e. narrative CMs, has been

proven useful in this regard. Furthermore, subjects had primarily addressed improvements related to the interface and interaction design of the AmI system, and have also requested more detailed content, for example, in case of the weather information service (cf. Appendix B). These results are particularly relevant for a revision of individual technical components of the system and thus, will guide future work outside the IKS project.
A secondary objective of the study was to test the maturity of implementation and architecture of the IKS-supported UIS from an end-user perspective. During the five-day evaluation of the system, only two problems were observed related to a hard disc failure and a broken connection to a remote server. Overall, the maturity of the current prototypical UIS was therefore perceived rather high by the subjects. Only performance aspects need to be considered for a future revision of the system with regard to the feedback of the study's participants (cf. Appendix B).

12 Future Work

In D4.1, we introduced a methodical procedure for the design, implementation and evaluation of interactive knowledge-supported Ubiquitous Information Systems (UIS) by means of the intelligent bathroom use case in IKS project. This use case represents a "far out" vision of IKS for direct user interactions with embedded contents that are organized by a "Semantic CMS Technology Stack". This case combines advanced content and knowledge management with an ubiquitous computing scenario in a place everybody is familiar with - the bathroom. The use case shows how users can interact with contents in physical environments in a way that leaves the dimension of "small windows to the info sphere" as known by the "monitor paradigm" (Janzen et al., 2010b). For realizing the intelligent bathroom use in two tasks of IKS, we applied the *Situational Design Methodology for Information Systems (SiDIS, formerly known as CoDesA)* (Maass & Janzen, 2011; Janzen et al., 2010a) (cf. Section 2). SiDIS ensures an integrated design of information systems considering users, social interactions, and physical surroundings besides plain technical issues. The design methodology offers a holistic view regarding situations supported by information systems that enables a comprehensive understanding of interactions within complex socio-technical systems as represented by the intelligent bathroom use case.
Within this contribution, key research challenges for the AmI case were identified supported by related work (cf. Section 3). One of these challenges was the development of a knowledge representation for such highly dynamic environments. We introduced an approach consisting of a contextual and situational part and presented our procedure in designing and modelling the knowledge representation by means of Ontology Design Patterns (Gangemi, 2005; Gangemi & Presutti, 2009) that are provided by IKS and that are combined with conceptual models based on SiDIS (cf. Section 4 and 5). Next, the development of an architecture for the AmI use case according to the IKS reference architecture was elaborated (cf. Section 6) followed by a description of the implementation of the AmI system, especially of the presentation of contents in the bathroom based on the semantic knowledge representation (cf. Section 7 and 8). Then, we presented the results of an empirical evaluation of the resulting software integrated in the physical bathroom environment (cf. Section 9). The elaboration of the re-iteration of the whole system regarding a validation of all available modules of IKS Beta (cf. Section 10) is followed by a discussion of experiences and lessons learned when developing interactive knowledge-supported ubiquitous information systems by means of IKS (cf. Section 11).
Lessons learnt from the user study (cf. Section 9 and Appendix B) can be directly used for a future revision of the IKS-supported UIS and its individual knowledge-based technical components. Recommendations for future work are therefore to address performance issues, the interface and interaction design as well as to aggregate and present more specific content for particular services such as the weather information service. Furthermore, we aim to automate the translation processes between the three conceptual models in SiDIS.

Appendix A – Requirements

Almost all of the requirements were described and elaborated in D2.1 "AmI Case Requirements Specification" (Janzen et al., 2010) as this document itself is the requirement specification of the AmI Case. In T4.1 further refinement of requirements was carried on. Based on the situation descriptions and use cases, missing detailed requirements were detected. Besides, some of the requirements from D2.1 (Janzen et al., 2010) were updated and elaborated. New requirements were identified as functional and data requirements and mapped to the situations as well as the updated ones which are shown in Appendix A. Functional requirements were elaborated by explaining exemplary situations, involved actors; their goals etc. according to the requirement elicitation process used in D2.1 (Janzen et al., 2010).

Functional Requirements:

ID	Description	Situation
FR-210001	The System shall be capable of providing site specific weather information	1
FR-210002	The System shall be capable of providing free time even suggestions	1
FR-210003	The System shall be capable of providing the possibility to order tickets	1
FR-210107	The System shall be capable of recognizing a user	1-12
FR-210204	The System shall provide the possibility to leave a message	2
FR-210205	The System shall be capable of displaying a message to the user	2
FR-210007	The System shall be capable of giving style suggestions	3
FR-210008	The System shall be capable of giving styling tutorials	3
FR-210009	The System shall be capable of giving additional style information	3
FR-210010	The System shall be capable of providing product suggestions of a specific style	3
FR-210011	The System shall be capable of providing personalized product suggestions	4
FR-210012	The System shall be capable of suggesting products by matching personal characteristics (e.g. skin type)	4
FR-210013	The System shall be capable of suggesting a store fitting to the user nearby	4
FR-210403	The System shall be capable of providing a online ordering mechanism for products	4
FR-210501	The System shall be capable of reminding the user of a specific appointment	5
FR-210101	The System shall be capable of editing and browsing a calendar	5
FR-210017	The System shall be capable of managing incoming appointment requests	5
FR-210503	The System shall be capable of detecting appointment conflicts	5
FR-210504	The System shall be capable of resolving appointment conflicts and providing alternative solutions	5
FR-210020	The System shall be capable of playing personalized music	6
FR-210021	The System shall be capable of fading out music when a specific event occurs	6
FR-210022	The System shall be capable of reacting on voice commands	6
FR-210023	The System shall be capable of requesting personal news	6
FR-210024	The System shall be capable of generating a personal news collage	6
FR-210025	The System shall be capable of presenting a personal news collage	6
FR-210026	The System shall be capable of filling the bathtub with water of a specific temperature	7
FR-210027	The System shall be capable of calculating the appropriate time for filling the bathtub	7
FR-210028	The System shall be capable of presenting movies	8-9
FR-210029	The System shall be capable of presenting video broadcasts	8, 12
FR-210030	The System shall be capable of providing an interface to check business mails	8
FR-210031	The System shall be capable of presenting personal photos	9
FR-210032	The System shall be capable of playing sounds	9
FR-210033	The System shall be capable of dynamic bathroom illumination	9
FR-210034	The System shall be capable of projecting media contents on the wall or furniture	9
FR-210035	The System shall be capable of dispensing scents	9
FR-210036	The System shall be capable of giving personalized outfit suggestions	10
FR-210037	The System shall be capable of determining dressing outfits based on context information and appointments	10
FR-210038	The System shall be capable of checking cloth availability in the wardrobe	10
FR-210039	The System shall be capable of presenting radio news streams	11
FR-210040	The System shall be capable of displaying news messages	11

FR-210041	The System shall be capable of the transition of content types	11
FR-210042	The System shall be capable of restricting media contents (annotated contents) according to a specific user	12
FR-210043	The System shall be capable of adapting to the people in the bathroom	12

Data Requirements:

ID	Description	Situation
DR-210001	Weather information shall be managed in a standardized format	1, 10
DR-210002	Location information shall be managed in a standardized format	1-12
DR-210003	Timestamp information shall be managed in a standardized format	1-12
DR-210004	Calendar free time information shall be managed in a standardized format	1
DR-210102	User model shall be managed in a standardized format	1-12
DR-210006	Price information shall be managed in a standardized format	1
DR-210007	Event description information shall be managed in a standardized format	1
DR-210008	Bathroom context information shall be managed in a standardized format	1-12
DR-210009	A user created message shall be stored	2
DR-210201	Media data (e.g. photos) shall be managed in a standardized format	3, 9
DR-210011	Styling tutorials shall be managed in a standardized format	3
DR-210012	Product information shall be managed in a standardized format	3-4, 7
DR-210013	Inventory information shall be managed in a standardized format	3
DR-210014	Nearby product availability shall be managed in a standardized format	3-4
DR-210015	Person-related attributes (e.g. skin types, water temperature preference) shall be managed in a standardized format	4, 7
DR-210016	Stores preferred by a specific user shall be managed in a standardized format	4
DR-210101	Calendar with appointments shall be managed in a standardized format	5, 7, 10
DR-210018	Appointment requests shall be managed in a standardized format	5
DR-210019	Personal news shall be managed in a standardized format	6
DR-210020	News collages shall be managed in a standardized format	6
DR-210021	Personal music collection shall be managed in a standardized format	6
DR-210022	Music contents shall be managed in a standardized format	6
DR-210023	Movie repository shall be managed in a standardized format	8
DR-210024	Video streams shall be managed in a standardized format	8, 12
DR-210025	Movies shall be managed in a standardized format	8-9
DR-210026	Business mails shall be managed in a standardized format	8
DR-210027	Sounds shall be managed in a standardized format	9
DR-210028	Illumination profiles shall be managed in a standardized format	9
DR-210029	Scents shall be managed in a standardized format	9
DR-210030	Wardrobe information shall be managed in a standardized format	10
DR-210031	Audio news content shall be managed in a standardized format	11
DR-210032	Image-based news content shall be managed in a standardized format	11
DR-210033	Text-based news content shall be managed in a standardized format	11
DR-210034	Annotated contents shall be managed in a standardized format	12
DR-210103	Context Ontology shall be in a standard agreed format	1-12
DR-210106	Rule Specifications shall be in a standard agreed format	1-12
DR-210110	A multimedia model has to be present in order to adapt format of presentation to the specific device	1-12
DR-210111	A device model has to be present in order to adapt presentation to the given device	1-12

Appendix B – Qualitative Feedback of the Evaluation

Tab. 12: Critical feedback of the subjects. Note: (n) represents the frequency of subjects that gave this feedback in case n > 1

Service	Interaction	Spatial Placement	Presentation Quality	Missing functionality
1. Weather Information Service		Positioning of weather is too central; a peripheral position would be much better to have more	Weather information was too lean (e.g., no information about rain probability, moisture, forecast,	Weather forecast and more information by touch gesture (4)

		space for the mirror image (3)	morning, noon, afternoon, etc.) (9)	
2. Event Recommendation Service		Positioning of the events is too central; a peripheral position would be much better (e.g., at the lower border of the touchscreen) (2)	Event information was too lean (e.g., no information about prices and category, and trailer) (4)	Slider or touch functionality missing to get more information about the events (5) Movie or concert Trailers should be available (3) Management of appointments with friends for events (e.g., by Facebook / Google+ integration) should be possible (2) Bookmark or "read it later"-functionality Events should be primarily recommended for the next weekend A graphical calendar should be provided in combination with events
3. Ticket Order Service	Touching is better than speech commands (7) Speech command fine but breaks during speech output were too long (you have to wait too long to give an answer) (7) Combination of touch and speech is confusing (2) Speech control needs getting used to (2) Everything should be controlled by speech commands (2) Speech command should be optional, touch should be the default form of interaction Speaking loud is problematic in the bathroom and morning time (in particular when someone just woke up)		More information about price and category of when ordering a ticket (6) The quality of the speech output was not smooth but also not correct in case German speech was used in combination with English event titles (6)	Canceling tickets orders should be possible The ticket ordering process must be protected (e.g., by a pin)
4. Personalized Music Service	Stopping music via the touch-sensitive border	Size of the projection on the wall screen was too small (7)	Low projection quality, too low resolution (7)	It was not possible to select or control a particular song (4)

	is not intuitive and takes too long, better would be a brief touch on the border (6) Implicit start of music by proximity is not intuitive, i.e. start and stop of the music should be gesture-controlled Speech command “stop music” would be better than touch gesture A visual feedback of the touch-sensitive border for stopping the music would ease the understanding of that interaction element			Presentation of music collection is missing (2) Starting music via gestures Music should stop automatically when adaptive news service is active and presents content in combination with loudspeakers Music should be personalized according to time of the day and day of the week Music clip should be presented in combination with the song Music should be started by touch-sensitive border
5. Personalized News Collage Service	Scrolling bars were to thin for a smooth interaction (4) Touching is better than speech commands because there is no direct visual feedback that the command has been recognized, you have to wait and guess (4) Speaking with a bathroom needs getting used to (2) Initial speech command can be dropped, just ask the question “What are the news?” (2) Speech command fine but you have to wait too long to give an answer (2) Speech recognition should be more sensitive	Size of the projection on the wall screen and shower was too small (5) Positioning of the news is too central; a peripheral position would be much better (e.g., at the lower border of the touchscreen) (2)	Low projection quality, too low resolution (7) In case news are displayed as text, the text should be short and more strikingly like on Bild.de (2) News clip in the shower plays not smooth (2) Gender of the speech output should be customizable	Slider functionality missing to scroll through the news in case they are text-based (2) It was not possible to control the news clip (stop, pause, etc.) (2)
6. Adaptive News Service		The news should stay on the wall screen instead of being presented on the mirror	The news clips started from the beginning on the wall screen when leaving the shower, it should be a smooth transition (5) News in form of text and	Music should stop automatically when adaptive news service is active and presents content in combination with loudspeakers

			images in should be used in the shower, whereby in the Bild.de style (less text, more images) There should be no transition from video (shower and wall screen) to text (mirror), the video should keep playing	
Overall	The overall reaction time as a feedback to user interactions was too low, e.g. user recognition, switching on the lights, content presentation (8) Mirror gets dirty by touching it, thus the interaction design must be revised (4) Speech commands should be used only for uncritical, non-payment services but not for a ticket order (3) In general, microphones in the bathroom are problematic for interactions Body height was too small such that the system was not able to recognize the person and start the corresponding services (this problem was fixed during the first day of the lab experiment) Interaction is confusing because you don't know which services start automatically and which require a pre-defined command either by speech, gesture or touch	The widgets on the touchscreen are distracting. The mirror as such should be the "main functionality" in the bathroom. Thus the content should be placed more in the periphery (23) The positioning of the widgets on the touchscreen should be more flexible (9) Proximity between person and mirror is too high such that the system works properly (5) The size of the widgets on the touchscreen should be bigger (3)	There was no design or concept behind the content presentation, e.g., the widgets on the touchscreen seemed to be positioned without any layout in mind; the IKS logo should be smaller and more semi-transparent to reduce distraction (10) Video-based mirror is strange (2) The content should be displayed in the form of a slide show like a content ticker, it would then need less space.	The widgets on the touchscreen should be more personalized (e.g., weather information from various cities) (9) The wall should be touch-sensitive, too The bathroom should wish the person a good day when leaving (the voice of a movie star should be configurable)

Appendix C – Types of Semantic "Notification and Message Format"

Type	Message variant	Subject	Predicate	Object	Source
UserEnters					Device Input Interpreta-tion
	unknown	null	message:UserEnters	device-created ID of user	
	already known	User individual	message:UserEnters	device-created ID of user	
UserLeaves					Device Input Interpreta-tion
	unknown	null	message:UserLeaves	device-created ID of user	
	already known	User individual	message:UserLeaves	device-created ID of user	
UserRecognized					Device Input Interpreta-tion
		User individual	message:UserRecognized	device-created ID of user	
UserMovesToLocation					Device Input Interpreta-tion
		User individual	message:UserMovesToLocation	BathroomLocation individual	
ContextUserPositio-nAdjusted					Context Manage-ment Com-ponent
		User individual	mes-sage:ContextUserPositionAdjusted	BathroomLocation individual	
DeviceDetected					Context Manage-ment Com-ponent
		SPDOAmiCaseDe-vice individual	message:DeviceDetected	null	
DeviceLost					Context Manage-ment Com-ponent
		SPDOAmiCaseDe-vice individual	message:DeviceLost	null	
UserActivated					Device Input Interpreta-tion Com-ponent
	unknown	null	message:UserActivated	SPDOAmiCaseDe-vice individual	
	already known	User individual	message:UserActivated	SPDOAmiCaseDe-vice individual	
UserDeactivated					Device Input Interpreta-tion Com-ponent
	unknown	null	message:UserDeactivated	SPDOAmiCaseDe-vice individual	
	already known	User individual	message:UserDeactivated	SPDOAmiCaseDe-vice individual	
UserSwitched					Device Input Interpreta-tion Com-ponent
	unknown	null	message:UserSwitched	SPDOAmiCaseDe-vice individual	
	already known	User individual	message:UserSwitched	SPDOAmiCaseDe-vice individual	
UserSelectedElement					Device Input Interpreta-tion Com-ponent
	unknown	null	message:UserSelectedElement	SPDOAmiCaseDe-vice individual	
		ContentItem sub-	Content:hasIdentifier	ContentItem sub-	

Type	Message variant	Subject	Predicate	Object	Source
		class individual		class individual	
	already known	User individual	message:UserSelectedElement	SPDOAmiCaseDevice individual	
		ContentItem subclass individual	Content:hasIdentifier	ContentItem subclass individual	
PresentationAvailable					Situation Processing Component or Presentation Component
		ContentItem subclass individual	message:PresentationAvailable	SPDOAmiCaseDevice individual	
		ContentItem subclass individual	ContentRealizations:isOfForm	ContentForm individual	
RemovePresentation					Situation Processing Component or Presentation Component
		ContentItem subclass individual	message:RemovePresentation	SPDOAmiCaseDevice individual	
		ContentItem subclass individual	ContentRealizations:isOfForm	ContentForm individual	
PresentationSuccessful					Device Access Component
		ContentItem subclass individual	message:PresentationSuccessful	null	
ContentItemAvailable					Context Management Component
		ContentItem subclass individual	message:ContentItemAvailable	null	
		ContentItem subclass individual	rdf:type	ContentItem subclass	
		ContentItem subclass individual	ContentRealizations:isOfForm	ContentForm individual	
UserProvidesInformation					Dialog Management Component
	content	User Individual	message:UserProvidesInformation	InterfaceService individual	
	command	User Individual	message:UserProvidedInformation	"stop music"	
		ContentItem subclass individual	rdf:type	ContentItemSubclass	
SituationChanged					Situation Recognition Component
		null	message:SituationChanged	CurrentSituation individual	

Appendix D – Semion Example

Semion is one of the component of the KReS which provides the following functionalities:

- reengineering of structured non-RDF sources into RDF
- refactoring of RDF data sets to specific vocabularies

These two functionality are designed to be independent. But also to work in a sort of workflow whose aim is to transform the data source provided as input first without any assumption at the domain level, and then to organize the triplified knowledge according to some specific one or more vocabularies that cover the domain. In the context of the AmI case, assume an external service provides weather information as XML streams (via some web service) as shown in the following snippet.

```
<?xml version="1.0"?>
<dwml version="1.0" xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:noNamespaceSchemaLocation="http://www.nws.noaa.gov/forecasts/xml/DWMLgen/schema
/DWML.xsd">
<data>
  <location>
      <location-key>SanFrancisco</location-key>
      <point latitude="37.47" longitude="122.26"/>
  </location>
  <time-layout time-coordinate="local" summarization="12hourly">
      <layout-key>k-p12h-n14-3</layout-key>
       <start-valid-time period-name="Today">2010-24-09T06:00:00-07:00</start-valid-
time>
      <end-valid-time>2010-24-09T18:00:00-07:00</end-valid-time>
  </time-layout>
  <parameters applicable-location="SanFrancisco">
      <temperature type="maximum" units="Fahrenheit" time-layout="k-p12h-n14-3">
        <name>Daily Maximum Temperature</name>
        <value>38</value>
      </temperature>
      <temperature type="minimum" units="Fahrenheit" time-layout="k-p12h-n14-3">
        <name>Daily Minimum Temperature</name>
      </temperature>
       <probability-of-precipitation type="12 hour" units="percent" time-layout="k-
p12h-n14-3">
        <name>12 Hourly Probability of Precipitation</name>
        <value>27</value>
      </probability-of-precipitation>
      <weather time-layout="k-p12h-n14-3">
        <name>Weather Type, Coverage, and Intensity</name>
       <weather-conditions weather-summary="Mostly Cloudy"/>
      </weather>
  </parameters>
</data>
</dwml>
```

Weather Information XML file

The above XML, extracted from National Oceanic and Atmospheric Administration's National Weather Service[63], states the temperature (Min and Max) the probability of precipitation and the weather condition of a location (i.e. San Franscico) between a start and an end time. By using Semion reengineering functionality the XML is transformed according to a meta-model[64] that expresses the structure of an XML file to a RDF file. The reengineered RDF is composed by a set of triples that needs to be aligned to a particular vocabulary, ontology or set of terms, that now is the AmI case ontology. Alignments are performed by the refactoring module of Semion. It needs an input RDF data set and a set of rules that can be expressed both in SWRL or in a syntax called KReSRule. The following snippet contains the rules in KReSRule syntax for the alignment of generated RDF data set to the AmI case ontology.

```
weatherAlignment.txt
oxml = <http://ontologydesignpatterns.org/ont/iks/oxml.owl#> .
schema = <http://kres.iks-project.eu/weather/san_francisco/schema#> .
ami_context = <http://iks/ami-case/context#> .
ami_content = <http://iks/ami-case/content#> .
composite = <http://www.topbraid.org/2007/05/composite.owl#> .

locationRule[
```

63 http://www.weather.gov
64 http://ontologydesignpatterns.org/ont/iks/oxml.owl

```
                              is(oxml:XMLElement, ?x) .
                              has(oxml:hasElementDeclaration, ?x, schema:location)
.
                              has(composite:child, ?x, ?y) .
                              has(oxml:hasElementDeclaration, ?y, schema:location-
key) .
                              values(oxml:nodeValue, ?y, ?z)
                                        ->
                              is(ami_context:Location, ?x) .
                              values(ami_context:description, ?x, ?z)
                    ] .

weatherRule[
                              has(oxml:hasElementDeclaration, ?x, schema:weather)
.
                              has(composite:child, ?x, ?y) .
                              has(oxml:hasElementDeclaration, ?y, schema:weather-
conditions) .
                              has(oxml:hasXMLAttribute,  ?y,  ?cond) .
                              values(oxml:nodeValue, ?cond, ?condition) .
                              has(oxml:hasXMLAttribute,  ?x,  ?z) .
                              has(oxml:hasAttributeDeclaration, ?z,
schema:weather_time-layout) .
                              has(oxml:nodeValue, ?z, ?time) .
                              has(composite:child, ?time, ?timeStartEl) .
                              has(composite:child, ?time, ?timeEndEl) .
                              has(oxml:hasElementDeclaration, ?timeStartEl,
schema:start-valid-time) .
                              has(oxml:hasElementDeclaration, ?timeEndEl,
schema:end-valid-time) .
                              values(oxml:nodeValue, ?timeStartEl, ?startTime) .
                              values(oxml:nodeValue, ?timeEndEl, ?endTime) .
                                        ->
                              is(ami_content:Weather, ?x) .
                              is(ami_content:WeatherCondition, ?y) .
                              has(ami_content:condition, ?x, ?y) .
                              values(ami_context:description, ?y, ?condition) .
                              has(ami_content:startTime, ?x, ?startTime) .
                              has(ami_content:endTime, ?x, ?endTime)
                    ] .

locationWeatherRule[
                              has(oxml:hasElementDeclaration, ?x, schema:data) .
                              has(composite:child, ?x, ?location) .
                              has(composite:child, ?x, ?parameter) .
                              has(oxml:hasElementDeclaration, ?location,
schema:location) .
                              has(oxml:hasElementDeclaration, ?parameter,
schema:parameters) .
                              has(composite:child, ?parameter, ?weather) .
                              has(oxml:hasElementDeclaration, ?weather,
schema:weather)
                                        ->
                              has(ami_content:place, ?weather, ?location)
                    ]
```

KResRule Alignments

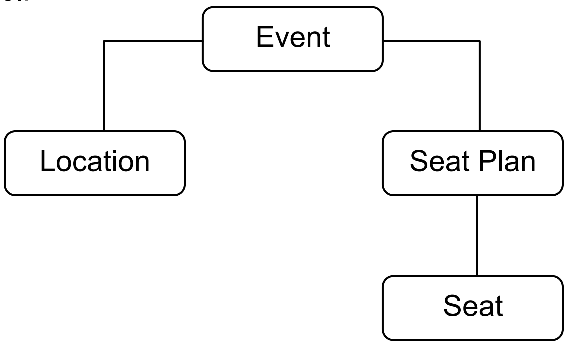
The result of the alignment is presented here:

```
<?xml version="1.0"?>
<rdf:RDF xmlns="http://kres.iksproject.eu/semion/autoGeneratedOntology#"
     xml:base="http://kres.iksproject.eu/semion/autoGeneratedOntology"
     xmlns:content="http://iks/ami-case/content#"
     xmlns:san_francisco="http://kres.iks-project.eu/weather/san_francisco#"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:context="http://iks/ami-case/context#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <owl:Ontology
rdf:about="http://kres.iksproject.eu/semion/autoGeneratedOntology">
        <owl:imports rdf:resource="http://iks/ami-case"/>
        </owl:Ontology>

<!--

///////////////////////////////////////////////////////////////////////////////////
////
    //
    // Individuals
    //

///////////////////////////////////////////////////////////////////////////////////
////
     -->




    <!-- http://kres.iks-project.eu/weather/san_francisco#root_data_0_location_1 --
>

    <owl:NamedIndividual rdf:about="http://kres.iks-
project.eu/weather/san_francisco#root_data_0_location_1">
        <rdf:type rdf:resource="http://iks/ami-case/context#Location"/>
        <context:description
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">SanFrancisco</context:descri
ption>
    </owl:NamedIndividual>



    <!-- http://kres.iks-
project.eu/weather/san_francisco#root_data_0_parameters_1_weather_7 -->

    <owl:NamedIndividual rdf:about="http://kres.iks-
project.eu/weather/san_francisco#root_data_0_parameters_1_weather_7">
        <rdf:type rdf:resource="http://iks/ami-case/content#Weather"/>
        <content:startTime
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">2010-24-09T06:00:00-
07:00</content:startTime>
        <content:endTime
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">2010-24-09T18:00:00-
07:00</content:endTime>
        <content:place rdf:resource="http://kres.iks-
project.eu/weather/san_francisco#root_data_0_location_1"/>
        <content:condition rdf:resource="http://kres.iks-
project.eu/weather/san_francisco#root_data_0_parameters_1_weather_7_weather-
conditions_3"/>
    </owl:NamedIndividual>
    <!-- http://kres.iks-
project.eu/weather/san_francisco#root_data_0_parameters_1_weather_7_weather-
conditions_3 -->

    <owl:NamedIndividual rdf:about="http://kres.iks-
project.eu/weather/san_francisco#root_data_0_parameters_1_weather_7_weather-condi
tions_3">
        <rdf:type rdf:resource="http://iks/ami-case/content#WeatherCondition"/>
        <context:description
rdf:datatype="http://www.w3.org/2001/XMLSchema#string">Mostly
Cloudy</context:description>
    </owl:NamedIndividual>
</rdf:RDF>
```

Aligned Weather Information

Appendix E – Exemplary Extracting of Content Model from a Jackrabbit Repository

Content model within a Jackrabbit content repository can be described through node type definitions. In order to demonstrate the content extraction from the Jackrabbit repository we have defined a sample content model. The following figure depicts the relation of the concepts of the sample model:



A new node type called “Event” is introduced in the JackRabbit repository using the type definition shown in the following snippet. It is defined as a subtype of “nt:base” node type and an instance of “Event” node type will have the following properties:

- startTime: denotes the start time of the event.
- endTime: denotes the end time of the event.
- takesPlace: a relation to another node which denotes the place where the event will take place.
- hasSeatPlan: a relation to another node which contains information about the seat plan.

```
/*Node type definition of Event */

// The namespace declaration
<content = 'http://iks/ami-case/content'>

// Node type name
[content:Event]

// Supertypes
> nt:base
```

```
// This node type supports orderable child nodes
orderable

// This is a mixin node type
//mixin

// Nodes of this node type have a property called 'description' of type STRING
- description (STRING)

// and it is...
//mandatory autocreated protected

// It has an on-parent-version setting of ...
copy

// Nodes of this node type have a property called 'startTime' of type STRING
- startTime (DATE)

// and it is...
//mandatory autocreated protected

// It has an on-parent-version setting of ...
copy

// Nodes of this node type have a property called 'endTime' of type STRING
- endTime (DATE)

// and it is...
//mandatory autocreated protected

// It has an on-parent-version setting of ...
copy

// Nodes of this node type have a property called 'takesPlace' of type PATH
- takesPlace (PATH)

// It has an on-parent-version setting of ...
copy

// Nodes of this node type have a property called 'hasSeatPlan' of type PATH
- hasSeatPlan (PATH)
// It has an on-parent-version setting of ...
copy
```

Node type definition of Event

The next snippet displays the node type definition of “Location”. It is defined as a subtype of “nt:base” nodetype and an instance of “Location” will have the following properties:

- name: denotes the name of the location.
- gps: involves the gps information of the location.
- indoor: indicates whether the event is an indoor event or an outdoor event.

```
/*Node type definition of Location Node Type/

// The namespace declaration
<context = 'http://iks/ami-case/context'>

// Node type name
[context:Location]

// Supertypes
> nt:base

// This node type supports orderable child nodes
orderable

// This is a mixin node type
//mixin
```

```
// Nodes of this node type have a property called 'name' of type STRING
- name (STRING)

// and it is...
//mandatory autocreated protected

// It has an on-parent-version setting of ...
copy

// Nodes of this node type have a property called 'gps' of type STRING
- gps (STRING)

// It has an on-parent-version setting of ...
copy

// Nodes of this node type have a property called 'indoor' of type STRING
- indoor (BOOLEAN)

= true
// and it is...
//mandatory autocreated protected


// It has an on-parent-version setting of ...
copy
```

Node type definition of Location

The following snippet introduces a node type called “SeatPlan” to Jackrabbit repository. It is defined as a subtype of “nt:base” node type and an instance of “SeatPlan” will have the following property:

- hasSeat: is a multivalued property which contains a list of seat

```
/*Node type definition of SeatPlan Node Type*/

// The namespace declaration
<content = 'http://iks/ami-case/content'>

// Node type name
[content:SeatPlan]

// Supertypes
> nt:base

// This node type supports orderable child nodes
orderable

// This is a mixin node type
//mixin

// Nodes of this node type have a property called 'hasSeat' of type PATH
- hasSeat (PATH)

// and multi-valued
multiple

// It has an on-parent-version setting of ...
copy
```

Node type definition of SeatPlan

Node type definition shown in the next snippet introduces a node type called “Seat”. It is defined as a subtype of “nt:base” node type and an instance of “Seat” will have the following properties:

- row: indicates the row of the seat in the seat plan
- column: indicates the column of the seat in the seat plan

```
/*Node type definition of Seat Node Type*/

// The namespace declaration
<content = 'http://iks/ami-case/content'>

// Node type name
[content:Seat]

// Supertypes
> nt:base

// This node type supports orderable child nodes
orderable

// This is a mixin node type
//mixin

// Nodes of this node type have a property called 'row' of type STRING
- row (STRING)

// and it is...
//mandatory autocreated protected

// It has an on-parent-version setting of ...
copy

// Nodes of this node type have a property called 'column' of type STRING
- column (STRING)

// and it is...
//mandatory autocreated protected


// It has an on-parent-version setting of ...
copy
```

Node type definition of Seat

Once these node types are defined in the Jackrabbit repository it becomes possible to create nodes of these node types. For instance, a movie event located at a place with a seat plan can be defined in the repository. The purpose of the CMS Adapter is to transform already available semantics as defined through above node type definitions and their instances as ontology in the knowledge base[65]. A mapping definition, which consists of bridge definitions for the above node type definitions, is shown here:

```
<BridgeDefinitions xmlns="model.jcr2ont.persistence.iks.srdc.com.tr">
  <InstanceBridge>
    <Query>/Events/%</Query>
    <PropertyBridge>
      <PredicateName>description</PredicateName>
    </PropertyBridge>
    <PropertyBridge>
      <PredicateName>hasSeatPlan</PredicateName>
    </PropertyBridge>
    <PropertyBridge>
      <PredicateName>endTime</PredicateName>
    </PropertyBridge>
    <PropertyBridge>
      <PredicateName>startTime</PredicateName>
    </PropertyBridge>
    <PropertyBridge>
      <PredicateName>takesPlace</PredicateName>
    </PropertyBridge>
  </InstanceBridge>
  <InstanceBridge>
    <Query>/Locations/%</Query>
```

[65] Alpha Stack Report for Semantic Data Access and Persistence Components, http://www.interactive-knowledge.org/sites/www.interactive-knowledge.org/files/iks_d54_SemanticPersistenceComponents_V0.2.pdf

```
    <PropertyBridge>
      <PredicateName>name</PredicateName>
    </PropertyBridge>
    <PropertyBridge>
      <PredicateName>gps</PredicateName>
    </PropertyBridge>
    <PropertyBridge>
      <PredicateName>inside</PredicateName>
    </PropertyBridge>
  </InstanceBridge>
  <InstanceBridge>
    <Query>/SeatPlans/%</Query>
    <PropertyBridge>
      <PredicateName>hasSeat</PredicateName>
    </PropertyBridge>
  </InstanceBridge>
  <InstanceBridge>
    <Query>/Seats/%</Query>
    <PropertyBridge>
      <PredicateName>column</PredicateName>
    </PropertyBridge>
    <PropertyBridge>
      <PredicateName>row</PredicateName>
    </PropertyBridge>
  </InstanceBridge>
</BridgeDefinitions>
```

Mapping description

CMS Adapter first transforms the node types as ontology classes in the target knowledge base and then starts to process the bridge definitions in the mapping definition. The following snippet depicts the generated classes from the node types.

```
…

    <!-- http://iks/ami-case/content#Event -->

    <owl:Class rdf:about="&content;Event">
        <rdfs:subClassOf rdf:resource="&nt;base"/>
    </owl:Class>



    <!-- http://iks/ami-case/content#Seat -->

    <owl:Class rdf:about="&content;Seat">
        <rdfs:subClassOf rdf:resource="&nt;base"/>
    </owl:Class>



    <!-- http://iks/ami-case/content#SeatPlan -->

    <owl:Class rdf:about="&content;SeatPlan">
        <rdfs:subClassOf rdf:resource="&nt;base"/>
    </owl:Class>



    <!-- http://iks/ami-case/context#Location -->

    <owl:Class rdf:about="&context;Location">
        <rdfs:subClassOf rdf:resource="&nt;base"/>
    </owl:Class>


 …
```

Generated Classes from the Node Types of Repository Model

The bridge definition transforms the events under the path “/Events/” with the specified properties described in the “PropertyBridge” clauses. In a similar fashion, locations, seat plans and seats are also transformed based on the bridge definitions. An example event converted as an individual is shown here:

```
<content:Event rdf:about="&workspacename;Avatar_1d0972a5-7d00-47ef-b71b-b074d5a14700">
   <rdf:type rdf:resource="&owl;Thing"/>
   <workspacename:startTime rdf:datatype="&xsd;dateTime"
       >2010-04-09T18:00:16.691+03:00</workspacename:startTime>
   <workspacename:endTime rdf:datatype="&xsd;dateTime"
       >2010-04-09T21:00:16.712+03:00</workspacename:endTime>
   <workspacename:description
rdf:datatype=”&xsd;normalizedString”>Avatar</workspacename:description>
   <jcr2ont:path
       >/Events/Avatar</jcr2ont:path>
   <jcr2ont:primaryNodeType>content:Event</jcr2ont:primaryNodeType>
   <workspacename:takesPlace    rdf:resource=”&workspacename;MoviPol_Cinema_bf45ffe1-d816-4157-
8d57-dde085c75009”/>
   <workspacename:hasSeatPlan     rdf:resource=”&workspacename;MoviPol_Seat_Plan_539d3f41-24b1-
4ef3-8d79-1c5bf6ed0d44”/>
    </content:Event>
```

An instance of event extracted from the Jackrabbit Repository

Once the ontology is extracted from the content model it may need further process to align with AmI Case ontologies. This can be achieved by processing SWRL rules on the generated ontology. When the existing repository model is transformed through CMS Adapter component, the need for synchronizing the changes in the repository model with the knowledgebase arises. As described in Alpha Stack Report for Semantic Data Access and Persistence Components CMS Adapter provides a RESTful API for receiving updates from the repository.

Appendix F – Survey of Features of Device Representation

Feature	Description	Representation in Turtle (Extract)
Function	Functionality that is offered by the device concerning input and output of specific types of contents → represented by SPDO Core concept Function	@prefix owl: <http://www.w3.org/2002/07/owl#> . @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> . @prefix : <http://im.dm.hs-furtwangen.de/ontologies/spdo/2010b/SPDO_AmICaseDevice.owl#> . :Function a owl:Class; rdfs:comment "AmI case: device-specific usage for describing functionality of device concerning its capability to present multimedia contents of diverse types --> currently, mpeg 7 multimedia content item is used, can be modelled differently in IKS ontology"@en; rdfs:subClassOf [a owl:Restriction; owl:onProperty :isFunctionOf; owl:allValuesFrom :Product] . :enablesInputOf rdfs:domain :Function . :specifiesAudioChannel rdfs:domain :Function . :specifiesPresentationArea rdfs:domain :Function . :description-of-function rdfs:domain :Function . :supportsContentEncoding rdfs:domain :Function . :enablesOutputOf rdfs:domain :Function . :belongsToFunctionCategory rdfs:domain :Function . :isFunctionOf rdfs:domain :Function . :hasFunction rdfs:range :Function . :isSupportedBy rdfs:range :Function . :AudioChannelIsSpecifiedBy rdfs:range :Function . :PresentationAreaIsSpecifiedBy rdfs:range :Function .
ContentType	Type of the content	@prefix owl: <http://www.w3.org/2002/07/owl#> .

Feature	Description	Representation in Turtle (Extract)
	that have to be presented by a device: audio, video, image and text → currently represented by concept MultimediaContent imported by MPEG7[66] ontology	@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> . @prefix : <http://im.dm.hs-furtwangen.de/ontologies/spdo/2010b/SPDO_AmICaseDevice.owl#> . :MultimediaContent a owl:Class; rdfs:subClassOf rdfs:Resource . :Creation rdfs:subClassOf :MultimediaContent . :Segment rdfs:subClassOf :MultimediaContent . :AudioVisual rdfs:subClassOf :MultimediaContent . :Video rdfs:subClassOf :MultimediaContent . :Audio rdfs:subClassOf :MultimediaContent . :Multimedia rdfs:subClassOf :MultimediaContent . :Image rdfs:subClassOf :MultimediaContent . :fitsTo rdfs:domain :MultimediaContent . :enablesInputOf rdfs:range :MultimediaContent . :enablesOutputOf rdfs:range :MultimediaContent . :fitsTo rdfs:range :MultimediaContent .
PresentationArea	Specification of devices that enable the presentation of video, image and/or text concerning the size of the presentation area, e.g. DIN A4 → represented by plugin concept PresentationArea	@prefix owl: <http://www.w3.org/2002/07/owl#> . @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> . @prefix : <http://im.dm.hs-furtwangen.de/ontologies/spdo/2010b/SPDO_AmICaseDevice.owl#> . :PresentationArea a owl:Class; rdfs:comment "AmI case: specification of devices that enable the presentation of video, image and/or text concerning the size of the presentation area, e.g. DIN A4 e.g., Beamer can present content in size 2x3 meters"@en; rdfs:label "Presentation area"@en . :description-of-presentation-area rdfs:domain :PresentationArea . :PresentationAreaIsSpecifiedBy rdfs:domain :PresentationArea . :specifiesPresentationArea rdfs:range :PresentationArea .
AudioChannels	Specification of devices that enable the presentation of video and/or audio concerning the audio channels, e.g. mono, stereo, Dolby ProLogic → represented by plugin concept AudioChannel	@prefix owl: <http://www.w3.org/2002/07/owl#> . @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> . @prefix : <http://im.dm.hs-furtwangen.de/ontologies/spdo/2010b/SPDO_AmICaseDevice.owl#> . :AudioChannel a owl:Class; rdfs:comment "AmI case: Specification of devices that enable the presentation of video and/or audio concerning the audio channels, e.g. mono, stereo, Dolby ProLogic e.g., Loudspeaker enables output of content of type audio (stereo)"@en; rdfs:label "Audio channel"@en . :description-of-audio-channel rdfs:domain :AudioChannel . :AudioChannelIsSpecifiedBy rdfs:domain :AudioChannel . :specifiesAudioChannel rdfs:range :AudioChannel .
ContentEncoding	Encoding of content items and functionality of devices regarding supported codecs, e.g., xvid, divx, mov → represented by plugin concept ContentEncoding	@prefix owl: <http://www.w3.org/2002/07/owl#> . @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> . @prefix : <http://im.dm.hs-furtwangen.de/ontologies/spdo/2010b/SPDO_AmICaseDevice.owl#> . :ContentEncoding a owl:Class; rdfs:comment "AmI case: Encoding of content items and functionality of devices regarding supported codecs, e.g., xvid, divx, mov e.g., Loudspeaker enables output of content of type audio (stereo) and encoding WMA, MP3"@en; rdfs:label "Content encoding"@en . :description-of-content-encoding rdfs:domain :ContentEncoding . :isSupportedBy rdfs:domain :ContentEncoding . :supportsContentEncoding rdfs:range :ContentEncoding .
DeviceStatus	Representation of	@prefix owl: <http://www.w3.org/2002/07/owl#> .

[66] http://mpeg.chiariglione.org/standards/mpeg-7/mpeg-7.htm

Feature	Description	Representation in Turtle (Extract)
	status of device that means available (for content presentation), busy, out of order → represented by SPDO Core concept ProductStatus	@prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> . @prefix : <http://im.dm.hs-furtwangen.de/ontologies/spdo/2010b/SPDO_AmICaseDevice.owl#> . :ProductStatus a owl:Class; rdfs:subClassOf [a owl:Restriction; owl:onProperty :isProductStatusOf; owl:allValuesFrom :Product] . :isProductStatusOf rdfs:domain :ProductStatus . :status-category rdfs:domain :ProductStatus . :description-of-device-status rdfs:domain :ProductStatus . :hasStatus rdfs:range :ProductStatus .
IndoorLocation	Representation of location of device inside the bathroom, e.g., near sink, inside shower → not represented in device representation, link to further representations	-
Visibility	Representation of access rights of devices, e.g., device enables content presentation for all users or just for Anna (public, private); equivalent to access rights of contents → represented by plugin concept Visibility with sub classes Private- and PublicVisibility	*Private Visibility (sub class of Visibility)* @prefix owl: <http://www.w3.org/2002/07/owl#> . @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> . @prefix : <http://im.dm.hs-furtwangen.de/ontologies/spdo/2010b/SPDO_AmICaseDevice.owl#> . :PrivateVisibility a owl:Class; rdfs:label "Private visibility"@en; rdfs:comment "AmI case: content presentation is available just for specific user"@en; rdfs:subClassOf :Visibility . :isVisibilityOf rdfs:domain :PrivateVisibility . :assignedToUser rdfs:domain :PrivateVisibility . :hasVisibility rdfs:range :PrivateVisibility . *Public Visibility (sub class of Visibility)* @prefix owl: <http://www.w3.org/2002/07/owl#> . @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> . @prefix : <http://im.dm.hs-furtwangen.de/ontologies/spdo/2010b/SPDO_AmICaseDevice.owl#> . :PublicVisibility a owl:Class; rdfs:label "Public visibility"@en; rdfs:comment "AmI case: content presentation is available for all users"@en; rdfs:subClassOf :Visibility . :isVisibilityOf rdfs:domain :PublicVisibility . :hasVisibility rdfs:range :PublicVisibility .
UserActivity	Representation of sensor based input regarding implicit and explicit user activities in the bathroom; functions of devices can enable these input	@prefix owl: <http://www.w3.org/2002/07/owl#> . @prefix rdfs: <http://www.w3.org/2000/01/rdf-schema#> . @prefix : <http://im.dm.hs-furtwangen.de/ontologies/spdo/2010b/SPDO_AmICaseDevice.owl#> . :UserActivity a owl:Class; rdfs:comment "AmI Case: Sensor based input concerning activities by the user in the bathroom"@en; rdfs:label "User activity"@en . :ImplicitUserActivity rdfs:subClassOf :UserActivity . :ExplicitUserActivity rdfs:subClassOf :UserActivity . :description-of-user-activity rdfs:domain :UserActivity . :enablesInputOf rdfs:range :UserActivity .

Appendix G – Exemplary Representation of Touch Screen Device

According to the aforementioned device representation, we modeled two exemplary individuals of devices. So, we tested the applicability of the resulting device representation according to our needs in the AmI case.
Now, we will shortly elaborate the representation of the touch screen device. First, the touch screen is described by a specific product ID, in our case the EAN. It has a manufacturer (Elo), a brand (Elo), a name (Elo 1715L) and a color (dark grey).
Furthermore, the touch screen offers a specific function. This function enables output of contents of type video, text and image. It further specifies an area of content presentation of 1280x1024 pixels. Third, the function of the touch screen supports a specific content encoding (xvid, mov, divx). Beside output functionalities, the function of the device also enables the input of explicit user activity; more precisely touching the screen, e.g., pressing a button. Concerning the visibility of the device, the touch screen offers a private visibility that is assigned to a specific user named Anna. That means the touch screen presents contents only to Anna. Currently, the touch screen is located near the door and available for presentation (device status). Last, the touch screen possesses a unique UPnP ID that allows an exact identification of the device by the AmI system. The following Turtle code shows the representation of the touch screen device:

```
Namespace(mpeg7 = <http://metadata.net/mpeg7/mpeg7.owl#>)
Namespace(owl = <http://www.w3.org/2002/07/owl#>)
Namespace(SPDO_AmICaseDevice = <http://im.dm.hs-
furtwangen.de/ontologies/spdo/2010b/SPDO_AmICaseDevice.owl#>)
Namespace(rdfs = <http://www.w3.org/2000/01/rdf-schema#>)
Namespace(rdf = <http://www.w3.org/1999/02/22-rdf-syntax-ns#>)
Namespace(SPDO = <http://im.dm.hs-
furtwangen.de/ontologies/spdo/2010b/SPDO.owl#>)
Namespace(SPDO_AmICaseDevice_Instance = <http://im.dm.hs-
furtwangen.de/ontologies/spdo/2010b/
SPDO_AmICaseDevice_Instance.owl#>)
Namespace(xsd = <http://www.w3.org/2001/XMLSchema#>)
Namespace(dc = <http://purl.org/dc/elements/1.1/>)

Ontology(<http://im.dm.hs-furtwangen.de/ontologies/spdo/2010b/
SPDO_AmICaseDevice_Instance.owl>
Individual(SPDO_AmICaseDevice_Instance:Touchscreen_Elo
   type(owl:Thing)
   type(SPDO_AmICaseDevice:Device)

value(SPDO:hasID SPDO_AmICaseDevice_Instance:EAN_TouchScreen_Elo)
value(SPDO:hasColor SPDO_AmICaseDevice_Instance:DarkGreyColor)
value(SPDO:hasFunction
SPDO_AmICaseDevice_Instance:Function_Touchscreen_Elo)
value(SPDO_AmICaseDevice:hasVisibility
SPDO_AmICaseDevice_Instance:PrivateVis_Anna)
value(SPDO_AmICaseDevice:hasIndoorLocation
SPDO_AmICaseDevice_Instance:Location_nearDoor)
value(SPDO:hasBrand SPDO_AmICaseDevice_Instance:EloBrand)
value(SPDO:hasManufacturer
SPDO_AmICaseDevice_Instance:Manufacturer_Elo)
value(SPDO:hasStatus
SPDO_AmICaseDevice_Instance:DeviceStatus_available)
value(SPDO_AmICaseDevice:upnp-id  "uuid:550e8400-e29b-11d4-a716-
446655440000::urn:schemas-upnp-org:service:AmI-Case-Device:2"@en)
value(SPDO:name  "Elo 1715L"@en)
)
)
```

Appendix H – Exemplary Handling of Devices Regarding Situation 1

In this section an exemplary workflow of handling of devices in situation 1 is provided.

1. With system start up several device concerning audio input, video/image output and activity recognition are available in the bathroom. After detecting all devices in the context of the *devices integration* step the semantic representations of the devices as well as the related OSGi bundles are retrieved.
2. After integrating the device representations in the knowledge base, the OSGi services are registered within the I/O management module to ensure hardware access on the devices.
3. Anna enters the bathroom.
4. A device recognizes that Anna entered the room and communicates this event using the Activity Service interface via the prior registered OSGi service.
5. The I/O management module broadcasts the activity to all other major system modules.
6. These steps are – from an I/O management perspective – repeated several times also for the recognition that Anna starts brushing teethes.
7. When the system decides to present weather information to Anna on the screen in front of her (site-specific weather information is an information that has no private access rights) the I/O management module handles this task vice versa: After evaluating the appropriate devices in the *device selection* step, the OSGi service of the screen is used to present the current weather information.
8. During the presentation of content, the status of the presentation device is set to busy to ensure that only one content item is presented at the same time.

A special form of content input is audio input. Since it is much more complex to interpret audio input than simple user activities, this task is not performed by the I/O management module. The communication module manages the interpretation of natural language input that means specific audio input. The communication module is able to listen directly to audio input provided by audio devices currently available in the bathroom. To realize this process, a specific service interface is offered by the I/O management module.

Appendix I – Interface Between Components

As described in prior sections, the architecture includes several components that offer interfaces to retrieve or submit information. These interfaces will be elaborated in the following table. The communication between these interfaces is standardized by the approach explained in Section 6.8.

Component	**Interface**	**Input parameters**	**Return value**
Device input interpretation component	Context event listener interface (needs to be implemented and registered)	None	Context event URI
	Content presentation feedback interface	Content presentation event URI	None
	Sensor data input	Sensor data	None

	interface		
Device access component	Content item presentation interface	Content item presentation command URI	None
	Device adjustment interface	Device adjustment command URI	None
	Output presentation		Output content (needs to be implemented and registered)
Situation processing component	Situation change notification interface	Situation event URI	None
Situation recognition component	Situation event listener interface (needs to be implemented and registered)	None	Situation event URI
Situation change management component	Context change notification interface	Context event URI	None
Device selection component	Device status listing interface	None	Device status listing event URI
Knowledge base	I/O device representation event listener interface (needs to be implemented and registered)	None	Knowledge event URI
	I/O device representation modification interface	Knowledge modification command URI	None
	Context representation event listener interface (needs to be implemented and registered)	None	Context event URI
	Context representation modification interface	Context modification command URI	None
	Situation representation event listener interface (needs to be implemented and registered)	None	Situation event URI
	Situation representation modification interface	Situation modification command URI	None
	Message creation interface	Message	URI
	Message retrieval	URI	Message

	interface		
Multi-modal interpretation component	Input Data Listener	URI	none
Presentation component	Presentation	none	URI
Dialog management component	Situation event Listener	Situation event URI	none
	Situation event notifier	none	Situation event URI
Content filter component	Filtered content retrieval interface	Knowledge querying command URI (*references original content set and filters*)	Content item response event URI
Content reengineer component	Content reengineering interface	Knowledge modification command URI	none
	Content reengineering listener	none	Content item modification event URI
Content refactoring component	Content refactoring interface	Knowledge modification command URI (*references refactoring rules*)	none
	Content refactoring listener	none	Content item modification event URI
Content aggregator component	Aggregated content retrieval interface	Knowledge querying command URI	Content item response event URI (*references aggregated content*)

References

Aarts, E. "Ambient Intelligence: A Multimedia Perspective," *IEEE MultiMedia* (11:1), 2004, pp. 12--19.

Aarts, E. and Roovers, R. "IC Design Challenges for Ambient Intelligence"'DATE '03: Proceedings of the conference on Design, Automation and Test in Europe', IEEE Computer Society, Washington, DC, USA, 2003, pp. 1002.

Alexander, C. *A Pattern Language: Towns, Buildings, Construction.*, Oxford University Press, 1978.

Behrendt, W., Gangemi, A., Maass, W. and Westenthaler, R. "Towards an Ontology-Based Distributed Architecture for Paid Content"'The Semantic Web: Research and Applications, Second European Semantic Web Conference, ESWC 2005, Heraklion, Crete, Greece, May 29 - June 1, 2005, Proceedings', 2005, pp. 257-271.

Bera, P., Krasnoperova, A. and Wand, Y. "Using Ontology Languages for Conceptual Modeling," *Journal of Database Management* (21:1), 2010, pp. 1–28.

Berners-Lee, T., Hendler, J. and Lassila, O. "The Semantic Web," *Scientific American* (284:5), 2001, pp. 34-43.

Brewster, C., O'Hara, K., Fuller, S., Wilks, Y., Franconi, E., Musen, M. A., Ellman, J. and Shum, S. B. "Knowledge Representation with Ontologies: The Present and Future," *IEEE Intelligent Systems* (19), 2004, pp. 72-81.

Bui, T. H. "Multimodal Dialogue Management---State of the Art"(TR-CTIT-06-01), Technical report, Centre for Telematics and Information Technology, University of Twente, Enschede, 2006.

Castañeda, V., Ballejos, L., Caliusco, M. L. and Galli, M. R. "The Use of Ontologies in Requirements Engineering," *Global Journal of Researches in Engineering* (10:6), 2010, pp. 2-8.

Christ, F., Nagel, B.: A Reference Architecture for Semantic Content Management Systems, Proceedings of EMISA 2011 workshop on "Enterprise Modelling and Information Systems Architectures", 2011 (to appear).

Cook, D. J., Augusto, J. C. and Jakkula, V. R. "Ambient intelligence: Technologies, applications, and opportunities," *Pervasive and Mobile Computing* (5), 2009, pp. 277-298.

Courtney, K.L. (2008). Privacy and Senior Willingness to Adopt Smart Home Information Technology in Residential Care Facilities. Methods of Information in Medicine, 47 (1), 76-81.

Crutzen, C. K. M. "Invisibility and the Meaning of Ambient Intelligence," *International Review of Information Ethics* (6), 2006, pp. 1-11.

Davis, R., Shrobe, H. and Szolovits, P. "What is a knowledge representation?," *AI Magazine* (14:1), 1993, pp. 17-33.

Decker, B., Ras, E., Rech, J., Klein, B. and Hoecht, C. "Self-organized Reuse of Software Engineering Knowledge supported by Semantic Wikis"'In Workshop on Semantic Web Enabled Software Engineering (SWESE)', 2005.

Dey, A. K. and Abowd, G. D. "Towards a better understanding of context and context-awareness"'In HUC ’99: Proc. of the 1st international symposium on Handheld and Ubiquitous Computing', Springer, 1999, pp. 304--307.

Dinev and Hart (2006), An Extended Privacy Calculus Model for E-Commerce Transactions, Information Systems Research, 17(1), 61-80.

Dobson, G. and Sawyer, P. "Revisiting Ontology-Based Requirements Engineering in the age of the Semantic Web"'International Seminar on "Dependable Requirements Engineering of Computerised Systems at NPPs"', 2006.

Dooley, J., Callaghan, V., Hagras, H., Bull, P. and Rohlfing, D. "Ambient intelligence - knowledge representation, processing and distribution in intelligent inhabited environments"'IET Conf. Publications', IEEE, 2006, pp. 51-59.

Dourish, P. "Keynote Speech: The Culture of Information: Ubiquitous Computing and Representations of Reality", Proc. of CLIHC 2005, 2005.

Dragoni, M., Da Costa Pereira, C. and Tettamanzi, A. G. B. "An ontological representation of documents and queries for information retrieval systems"'Proc. of the 23rd International Conf. on Industrial engineering and other applications of applied intelligent systems', Springer, Berlin, Heidelberg, 2010, pp. 555--564.

Dubois, E., Hagelstein, J., Lahou, E., Ponsaert, F. and Rifaut, A. "A knowledge representation language for requirements engineering," *Proc. of the IEEE* (74:10), 1986, pp. 1431 - 1444.

Ducatel, K., Bogdanowicz, M., Scapolo, F., Leijten, J. and Burgelman, J. C. "Scenarios for Ambient Intelligence in 2010", Technical report, IST Advisory Group, 2001.

Eickhold, J.; Fuhrmann, T.; Saballus, B.; Schlender, S. & Suchy, S. “AmbiComp: A platform for distributed execution of Java programs on embedded systems by offering a single system image” AmI-Blocks'08, Workshop at the European Conference on Ambient Intelligence 2008, Nuremberg, Germany, 2008

Filho, O. F. F. and Ferreira, M. A. G. V. "SEMANTIC WEB SERVICES: A RESTFUL APPROACH"'Proceedings of IADIS International Conference WWW/Internet 2009', 2009.

Foster, M. E. " State of the art review: Multimodal fission"(D6.1), Technical report, COMIC Project, Edinburgh, 2002.

Fraser, N. "Assessment of Interactive Systems", *in* D. Gibbon, R. M. and Winski, R., ed.,'Handbook on Standards and Resources for Spoken Language Systems', Germany, Berlin, 1997, pp. 564--615.

Friedewald, M., Costa, O. D., Punie, Y., Alahuhta, P. and Heinonen, S. "Perspectives of ambient intelligence in the home environment," *Telemat. Inf.* (22:3), 2005, pp. 221--238.

Fuchs, F., Hochstatter, I. and Henrici "Assisting the User in Selecting Devices for Media Content," *GI Jahrestagung* (1), 2006, pp. 241-247.

Gangemi, A. "Ontology Design Patterns for Semantic Web Content" 'The Semantic Web ISWC 2005', Springer, 2005.

Gangemi, A. and Mika, P. "Understanding the semantic web through descriptions and situations."'DOA/CoopIS/ODBASE 2003 Confederated International Conferences DOA, CoopIS and ODBASE, Proceedings', Springer, 2003.

Gangemi, A. and Presutti, V. "Ontology Design Patterns" 'Handbook on Ontologies, 2nd Ed.', Springer, 2009.

Gárate, A.; Lucas, I.; Herrasti, N. & López, A. “Ambient Intelligence as paradigm of a full Automation Process at Home in a real application” Proceedings of IEEE International Symposium on Computational Intelligence in Robotics and Automation, 2005 (CIRA 2005)

Groza, T., Schutz, A. and Handschuh, S. "SALT: a semantic approach for generating document representations"'Proc. of the 2007 ACM symposium on Document engineering', ACM, New York, NY, USA, 2007, pp. 171--173.

Gruninger, M. and Fox, M. S. "The role of competency questions in enterprise engineering"'Proc. of the IFIP WG5.7 Workshop on Benchmarking - Theory and Practice', 1994.

Hinchman, L. and Hinchman, S. K. *Memory, Identity, Community: The Idea of Narrative in the Human Sciences*, State University of New York Press, 1997.

Hofer, T., Schwinger, W., Pichler, M., Leonhartsberger, G., Altmann, J. and Retschitzegger, W. "Context-Awareness on Mobile Devices - the Hydrogen Approach" 'HICSS '03: Proceedings of the 36th Annual Hawaii International Conference on System Sciences (HICSS'03) - Track 9', IEEE Computer Society, Washington, DC, USA, 2003, pp. 292.1.

Janzen, S., Filler, A. and Maass, W. "Designing Ubiquitous Information Systems based on Conceptual Models (to appear)" 'Proc. of Workshop "Sozio-technisches Systemdesign im Zeitalter des Ubiquitous Computing (SUBICO 2011)" at Informatik 2011, Berlin', 2011.

Janzen, S., Kowatsch, T. and Maass, W. "A Methodology for Content-Centered Design of Ambient Environments" 'Proc. of DESRIST 2010', 2010a, pp. 210-225.

Janzen, S., Maass, W., Kowatsch, T., Filler, A., Romanelli, M., Germesin, S., Becker, T., Laleci, G. B., Kilic, O., Ocalan, C., Alpay, E. and Dogac, A. "IKS Deliverable. D2.1 Report: AMI Case Requirements Specification", Technical report, EU project IKS, 2010b.

Janzen, S., Kowatsch, T., Maass, W. & Filler, A., Linkage of Heterogeneous Knowledge Resources within In-store Dialogue Interaction, 9th International Semantic Web Conference (ISWC 2010), Shanghai, China, 2010c.

Janzen, S. & Maass, W., Smart Product Description Object (SPDO), Poster Proceedings of the 5th International Conference on Formal Ontology in Information Systems (FOIS2008), Saarbrücken, Germany, 2008.

Kamis, A., Koufaris, M. and Stern, T. (2008). Using an Attribute-Based Decision Support System for User-Customized Products Online: An Experimental Investigation. MIS Quarterly, 32 (1), 159-177.

Kim, E. and Choi, J. "An Ontology-Based Context Model in a Smart Home" 'Computational Science and Its Applications - ICCSA 2006, International Conference, Glasgow, UK, May 8-11, 2006, Proceedings, Part IV', Springer, 2006, pp. 11-20.

Kopecký, J.; Gomadam, K. & Vitvar, T. hRESTS: “An HTML Microformat for Describing RESTful Web Services” WI-IAT '08: Proceedings of the 2008 IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology, IEEE Computer Society, 2008, 619-625

Kuechler, Jr., W. L. & Vaishnavi, V. K. “An expert system for dynamic re-coordination of distributed workflows” Expert Syst. Appl., Pergamon Press, Inc., 2008, 34, 551-563

Lamb, R. & Kling, R. “Reconceptualizing Users as Social Actors in Information Systems Research” MIS Quarterly, 2003, 27

Larsson, S. and Traum, D. "The information state approach to dialogue management", *in* Smith, R. and van Juppevelt, J., ed.,'Current and New Directions in Discourse and Dialogue', Kluwer Academic Publishers, Dordrecht, 2003, pp. 325-353.

Lashina, T. "Intelligent Bathroom" 'Proceedings of European Symposium on Ambient Intelligence (EUSAI'04)', Technical report, Philips Research, 2004.

Lassila, O. "Applying Semantic Web in Mobile and Ubiquitous Computing: Will Policy-Awareness Help?"'Proc. of the Semantic Web and Policy Workshop, 4th International Semantic Web Conf.', 2005.

Lechner, U. and Schmid, B. F. "Communities and Media - Towards a Reconstruction of Communities on Media" 'Proc. of HICSS 2000', IEEE Press, 2000.

Luz, S. "State-of-the-art survey of dialogue management tools"(D2.7a), Technical report, Disc Project, 1999.

Lyytinen, K. and Yoo, Y. "Issues and Challenges in Ubiquitous Computing," *Commun. ACM* (45:12), 2002, pp. Commun. ACM, Vol. 45, No. 12. (December 2002), pp. 62-65.

Lyytinen, K. and Yoo, Y. "Research Commentary: The Next Wave of Nomadic Computing," *Inform Syst Res* (13:4), 2002, pp. 377-388.

Maass, W. and Janzen, S. "Pattern-Based Approach for Designing with Diagrammatic and Propositional Conceptual Models" 'Proc. of DESRIST 2011', 2011, pp. 192-206.

Maass, W., Storey, V. C. and Kowatsch, T. "Effects of External Conceptual Models and Verbal Explanations On Shared Understanding in Small Groups, (to appear)" 'Proc. of 30th International Conf. on Conceptual Modeling (ER 2011), Brussels, Belgium', 2011b.

Maeda, E. and Minami, Y. "Steps towards Ambient Intelligence," *NTT Technical Review* (4), 2006, pp. 50-55.

Martin, D., Domingue, J., Brodie, M. L. and Leymann, F. "Semantic Web Services, Part 1," *IEEE Intelligent Systems* (22:5), 2007, pp. 12--17.

May, M. "Research Challenges in Ubiquitous Knowledge Discovery" 'Next Generation of Data Mining', Chapman & Hall, 2008.

McTear, M. F. "Spoken dialogue technology: Enabling the conversational user interface," *ACM Computational Surv.* (34:1), 2002, pp. 90-169.

Moore, G.C. and Benbasat, I. (1991). Development of an instrument to measure the perceptions of adopting an information technology innovation. Information Systems Research, 2 (3), 192-222.

Nunnally, J.C. (1967). Psychometric Theory McGraw-Hill, New York.

Nuzzolese, A. G., Gangemi, A. and Presutti, V. "Gathering Lexical Linked Data and Knowledge Patterns from FrameNet"'Proc. of the 6th International Conf. on Knowledge Capture (K-CAP)', 2011.

Orlikowski, W. J. and Barley, S. R. "Technology and Institutions: What Can Research on Information Technology and Research on Organizations Learn from Each Other?," *MIS Quarterly* (25:2), 2001, pp. 145-165.

Ou, S., Georgalas, N., Azmoodeh, M., Yang, K. and Sun, X. "A Model Driven Integration Architecture for Ontology-Based Context Modelling and Context-Aware Application Development"'Model Driven Architecture -- Foundations and Applications', Springer, 2006.

Oviatt, S. "Multimodal Interfaces", *in* Sears, A. and Jacko, J. A., ed.,'The Human Computer Interaction Handbook', 2008, pp. 413--432.

Padilla, N., Iribarne, L., Asensio, J. A., Mucoz, F. J. and Ayala, R. "Modelling an Environmental Knowledge-Representation System"'Proc. of the 1st world summit on The Knowledge Society: Emerging Technologies and Information Systems for the Knowledge Society', Springer, Berlin, Heidelberg, 2008, pp. 70--78.

Peters, S. and Shrobe, H. E. "Using Semantic Networks for Knowledge Representation in an Intelligent Environment" 'Proc. of the Int. Conf. on Pervasive Computing and Communications', IEEE Computer Society, Washington, DC, USA, 2003.

Alexander Pfalzgraf, Norbert Pfleger, Jan Schehl and Jochen Steigner. ODP – Ontology-based Dialogue Platform. Technical Report, SemVox GmbH 2008.

Presutti, V., Daga, E., Gangemi, A. and Blomqvist, E. "eXtreme Design with Content Ontology Design Patterns"'Proc. of WOP 2009, collocated with ISWC-2009', CEUR Workshop Proceedings, 2009.

Purao, S.; Han, T. & Storey, V. “Improving Reuse-based Design: Augmenting Analysis Patterns Reuse with Learning” Information Systems Research, 2003, 14, 269-290

Ramos, C., Augusto, J. C. and Shapiro, D. "Ambient Intelligence---the Next Step for Artificial Intelligence," *IEEE Intelligent Systems* (23:2), 2008, pp. 15--18.

Riechert, T., Lauenroth, K., Lehmann, J. and Auer, S. "Towards Semantic based Requirements Engineering"'Proc. of the 7th International Conf. on Knowledge Management (I-KNOW)', 2007.

Rogers, E.M. (2003). Diffusion of innovations. (5 ed.) Free Press, New York.

de Ruyter, B. and Aarts, E. "Ambient intelligence: visualizing the future" 'AVI '04: Proceedings of the working conference on Advanced visual interfaces', ACM, New York, NY, USA, 2004, pp. 203--208.

Searle, J. R. *Speech acts: An essay in the philosophy of language*, Cambridge University Press, London, 1969.

SemVox GmbH. 2009 (02). Dokumentation: Das ODP-Flex Grammatik- Framework. SemVox GmbH, Saarbrücken.

Shadbolt, N. "Ambient Intelligence," *IEEE Intelligent Systems* (18:4), 2003, pp. 2--3.

Sheth, A. P., Gomadam, K. and Lathem, J. "SA-REST: Semantically Interoperable and Easier-to-Use Services and Mashups," *IEEE Internet Computing* (11), 2007, pp. 91-94.

Sonntag, D. *Ontologies and Adaptivity in Dialogue for Question Answering*, AKA and IOS Press, Heidelberg, 2010.

Sonntag, D.; Reithinger, N.; Herzog, G. & Becker, T. (2010), A Discourse and Dialogue Infrastructure for Industrial Dissemination 'Spoken Dialogue Systems for Ambient Environments: Second International Workshop, IWSDS 2010, Gotemba, Shizuoka, Japan, pp. 132-143.

Suarez-Figueroa, M. C. and Gomez-Perez, A. "NeOn Methodology for Building Ontology Networks: a Scenario-based Methodology" 'Proceedings of the International Conference on SOFTWARE, SERVICES & SEMANTIC TECHNOLOGIES (S3T 2009)', 2009.

Sun, H., Florio, V. D., Gui, N. and Blondia, C. "Promises and Challenges of Ambient Assisted Living Systems"'ITNG '09: Proceedings of the 2009 Sixth International Conference on Information Technology: New Generations', IEEE Computer Society, Washington, DC, USA, 2009, pp. 1201--1207.

Veres, C., Sampson, J., Bleistein, S. J., Cox, K. and Verner, J. "Using Semantic Technologies to Enhance a Requirements Engineering Approach for Alignment of IT with Business Strategy", *Complex, Intelligent and Software Intensive Systems, International Conference* 'Proc. of International Conf. of Complex, Intelligent and Software Intensive Systems', IEEE Computer Society, Los Alamitos, CA, USA, 2009, pp. 469-474.

Wang, X. H., Zhang, D. Q., Gu, T. and Pung, H. K. "Ontology Based Context Modeling and Reasoning using OWL," *Pervasive Computing and Communications Workshops, IEEE International Conference on* (0), 2004, pp. 18.

Wixom, B.H. and Todd, P.A. (2005). A Theoretical Integration of User Satisfaction and Technology Acceptance. Information Systems Research, 16 (1), 85-102.

Zukier, H. “The Paradigmatic and Narrative Modes in Goal-Guided Inference” Handbook of Motivation and Cognition: Foundations of Social Behavior, Guilford Press, New York, 1986, 465-502.