A Pattern-based Ontology Building Method for Ambient Environments

Wolfgang Maass[1,2] and Sabine Janzen[1]

[1] Research Center for Intelligent Media (RCIM)
Furtwangen University, Robert-Gerwig-Platz 1
D-78120 Furtwangen, Germany
{wolfgang.maass,sabine.janzen}@hs-furtwangen.de
http://im.dm.hs-furtwangen.de
[2] Institute of Technology Management, University of St. Gallen
Dufourstrasse 40a, CH-9000 St. Gallen, Switzerland
wolfgang.maass@unisg.ch
http://www.item.unisg.ch

Abstract. Ambient environments are characterized by an ever increasing amount of information that needs to be selected and organized in order to make correct assumptions about users, entities, etc. within a specific context. This issue can be addressed by using ontologies that meet the specific requirements of such environments. In this paper, we survey ontology engineering methods that represent an adequate approach to creating adequate ontologies. Because unprecedented, we introduce a Pattern-based Ontology Building Method for Ambient Environments (POnA) and exemplify this method through the development of a domain-specific ontology for cosmetic products within ambient shopping environments.

Key words: Ambient environments, design patterns, ontology engineering methods

1 Introduction

In recent years, research in intelligent environments and entities has increased. The embedding of adaptive information and communication services into everyday physical things characterizes ambient environments (Ambient Intelligence (AmI)). A number of prototype systems for ambient environments have been designed and implemented, such as [1–4]. The challenge is to interpret sensor data so that adaptive behavior of ambient environments can be generated which naturally supports users in, for instance, situations of problem solving, relaxing, informing, and communicating with other users. Available information needs to be selected and organized so that ambient environments are able to make correct assumptions about users and physical entities within a specific context [5]. However, a major shortcoming of these AmI systems is weak support of knowledge sharing [6], i.e., AmI systems are typically based on proprietary and fixed sets of

representations that are designed for particular AmI applications. This, in turn, poses hurdles for integrating external information that is stored in the infosphere of digital environments, such as the World Wide Web (WWW). Bridging the gap between sensor-data-driven applications and infospheres requires interoperable knowledge representations. Through this, information can be reused in both directions. Because both AmI environments and infospheres can become quite complex, a more principled, ontology-driven approach for the creation of appropriate knowledge representations is required [6]. Initial approaches were top-down approaches based on foundational ontologies that drilled down conceptual structures until they reached a conceptual level compatible with information derived from sensor data (e.g., [7]). For complexity reasons, these systems neglected most of the ontological structure that are given by the overarching ontology, which meant that the ontology was only helpful at design time. In these systems, semantics coded into ontological structures were typically not used at run-time.

Some applications that used ontologies indicated the value of smaller pieces of more abstract ontological structures, called design patterns, that could reused. The idea of design patterns has several origins, such as Christopher Alexander's work on design patterns in architecture, Kevin Lynch's work on mental models of city structures, and computational knowledge patterns [8, 9]. Ontology design patterns are conceptual macros that are repeatedly used by experts for solving problems in their particular domains (cf. [10]). They are a means for extracting knowledge from experts that can be used for designing applications with a holistic understanding of a domain. Furthermore, machine-processible representations of design patterns can be used for the generation of a system behavior itself.

AmI environments try to support natural behavior. Therefore ontology design patterns must combine domain knowledge and enhancements using information services grounded in sensor data and Web-based infospheres. In general AmI environments are required to be context-oriented, user-centered, and network-enabled [7]. Context-orientation is achieved by adaptating to particular situations, to objects within these situations, and to domain constraints such as contractual restrictions. Furthermore, AmI environments adapt to users within situations in a personalized and proactive manner. When an AmI environment is network-enabled, any object can in principle establish relationships with any other object or service within and beyond a particular situation. In summary, ontology design patterns for AmI environments can be defined on various layers: (1) users, (2) objects, (3) services, (4) physical space, (5) infosphere (information space), and (6) social space [11]. (4) through (6) establish networks among entities from (1) through (3). A detailled discussion is beyond the scope of this article.

Here, we discuss a tentative design method for ontology design patterns (POnA) and its application to the design of AmI environments with a focus on Natural Language Processing (NLP). A review of existing methods has revealed (cf. Section 2&3) that a dedicated method suitable for ambient environments

is not yet available. Therefore we introduce a *Pattern-based Ontology Building Method for Ambient Environments (POnA)* and exemplify this method through the development of a pattern-based ontology for an AmI shopping environment. Next, we will discuss existing ontology building approaches and their applicability in AmI environments. In Section 3, we illustrate POnA through the development of an exemplary ontology. We then discuss some findings (Section 4) that exemplify patterns (Section 5). Finally, we conclude with a summary and an outlook on future work.

2 Related Work

In general, there are diverse approaches for the design and development of ontologies [12]. Systematic methodologies concentrate on the ontology development process and are independent of particular languages. Patterns are light-weight versions of methodologies that include useful hints. Svatek argues that repositories of such reusable patterns might improve the accuracy, transparency and reasonability of an ontology [12]. In the following, systematic methodologies and pattern-based approaches for ontology engineering are briefly described.

2.1 Systematic Ontology Building Methods

There are diverse ontology building methods for designing ontologies (for a review of earlier methodologies cf. [13] NeOn 5.4.1.). For instance, Uschold and King describe a methodology for building ontologies anew [14] while METHONTOLOGY considers reuse of other ontologies [15]. The more recent NeOn methodology focusses on the reuse and combination of distributed ontologies with a special emphasis on ontology design patterns [16]. The Unified Process for Ontology building (UPON) [17] applies a phase-structured software engineering viewpoint by following the Unified Process model. All these methodologies follow a basic pattern. First, a scope is defined by a domain of interest. Second, scenarios are defined, which are used to identify competency questions (CQ) to be answered by the ontology [18]. Thereafter, terms are gathered and translated into formal concepts that are connected. The final result is called a formal ontology. The NeOn methodology defines ontology design patterns as best practices for more efficient ontology engineering processes. Additionally it describes generic processes for ontology evaluation, evolution, and localization [16].

Even though the NeOn methodology is rich with respect to complete ontology life cycle, it lacks depth with regard to the application of design patterns. At the moment, matching problems with ontology design patterns and reusing and composing of ontology design patterns are complex tasks still left to the knowledge engineer's expertise [16]. In the following, we describe a methodology that combines early phases of the NeOn methodology and the UPON methodology for designing ontologies for AmI environments. Special emphasis is given to a problem solving viewpoint often found in AmI scenarios.

2.2 Building Pattern-based Ontologies

Christopher Alexander introduced the term "design pattern" for shared guidelines that help to solve architectural design problems [19]. He argued that a good design can be achieved using rule sets, i.e. patterns. The potential for reusing ontological structures through a pattern-based approach was first developed by Clark et al. [20]. They emphasized the importance of combining concepts within a vocabulary using "knowledge patterns", i.e., frequently recurring, structurally similar patterns of axioms. The notion of ontology design patterns was used by Gangemi [10] when presenting Conceptual Ontology Design Patterns (CODePs) as a useful resource for engineering ontologies for Semantic web infrastructures. CODePs are represented by textual, semiformal, and formal descriptions similar to Alexander's initial approach.

3 Pattern-based Ontology Building Method for Ambient Environments (POnA)

Our goal is the development of an ontology design pattern library specific to AmI environments that considers all six layers (cf. Section 1). These patterns are grounded in the patterns of the Ontology Design Pattern ODP library.[3] The PoNA methodology reuses UPON's detailed engineering approach and combines it with an approach proposed by the NeOn methodology that is centered on ontology design patterns. Currently, we focus on early development phases of ontology engineering. Rigorous ontology evaluation and evolution phases will be considered in our future work.

Ontological design patterns have several advantages. They improve explicit modularizations of knowledge bases and enable separation of abstract theories from real-world phenomena [20]. Their usage ensures that better explications concerning structure and modeling decisions are made when constructing a formal axiom-rich ontology [20, 10]. Furthermore, ontology design patterns provide new opportunities for ontology integration [21]. Contrary to systematic methodologies, pattern-based approaches do not dispose of structured methodologies that consist of specific activities and outcomes. Our hypothesis is that a combination of systematic methodologies (cf. Section 2.1) and ontology design patterns (cf. Section 2.2) constitute a more detailled and thus efficient approach to designing ontologies for ambient environments.

In the following, we present the ***P**attern-based **On**tology Building Method for **A**mbient Environments (POnA)*. Following UPON [17], POnA consists of four engineering phases: *Requirements, Design, Implementation* and *Test*. Each phase is subdivided into activities, contains decision points, and provides clearly defined outcomes. The re-use of design patterns is integrated into the design phase. We will exemplify POnA for an AmI application in the cosmetics domain [22].

[3] http://ontologydesignpatterns.org

3.1 Requirements Phase

The requirements phase aims at the identification of business needs for modeling a domain of interest and specification of the environmental aspects of the ontology, e.g., users and scenarios. This phase consists of the following activities: *(1) defining the domain of interest and scope, (2) identifying objectives, (3) defining scenarios, (4) defining terminology* and *(5) identifying competency questions.*

Defining Domain of Interest & Scope According to [14] and [17], defining a domain of interest is an important step in focusing on a particular fragment of the world to be modeled. Therefore, prospective users and the type of ontology to be used are circumscribed. In line with the scope of a particular ontology, the most important concepts and their characteristics are identified. Consequently, some parts of a domain of interest are brought into focus. Within our cosmetics domain, we focus on ontological representations of the following product concepts: *perfume and fragrances, make-up for eyes, makeup for lips, hand care, nail care, foot care, make-up for facial skin, liquid hair care, hair spray, hair color, dental care, shower gel, skin care, and sunscreen.* Target groups of resulting ontologies are manufacturers, retailers and customers, which might use the ontology for two different real-world situations:

- **In-store purchase situation** - The customer communicates with a cosmetic product. She might search for an individual solution, e.g."I have dry skin. Which vanishing creme suits me?"
- **Usage situation at home** - A product advises a customer on correct application procedure and initiates re-purchases through communication with appropriate web-based shopping services.

Identifying Objectives In this step, motivations for an ontology are collected together with associated problems. This step is important for later reuse of ontologies. It indicates to other knowledge engineers the importance of this ontology and the problem types that are targeted. We found that physical products are mainly described in a non-semantic way; their descriptions exist in terms of static databases or XML structures (e.g., BMEcat®, ETIM/eCl@ss and GS1). Modeling of enterprises or processes is generally sophisticated, but the description of products rarely exceeds the scope of classification. We intend to integrate physical products into communicative situations in ambient shopping environments. Therefore, semantically annotated product information is required to realize personalized communications between different stakeholders and products.

Defining Situations Situations are textual descriptions of integrated performances of a particular interaction type. Situations conceive different entities (objects, subjects, information, and services) and their interactions (for a discussion cf. [11]). Situations are prototypes that reflect characteristic features of a corresponding class of situations, e.g., shopping situations. Thus, situations

resemble frames [8], schemas [9], and use cases. For instance:

"Anna has dry skin and searches for a vanishing creme matching her skin in a shop. She wants to take a look at the cremes that are right for her, so she asks all products in the store for a solution to her specific skin problem. Six vanishing cremes give notice that they want to solve her problem because they are suitable for dry skin. Anna goes to the creme closest to her and initiates a dialogue with the intelligent product. She asks whether the creme is a gel-based moisturizer. Furthermore, she wants to know about the ingredients. Then, Anna asks for the price as well as current discount campaigns and matching products. The creme informs Anna about a bundle price with a 5% discount for the vanishing creme and the corresponding eye care. Anna decides to buy both products."

Defining Terminology The terminology was extracted from situation reports gathered in workshops with experts and was automatically extracted based on Web-based product descriptions. For the cosmetics domain, 130 terms were extracted and categorized into five term categories (for an excerpt see Table 1): *Actors and Roles, Products and Features, Environment and Situation, Problems* and *Solutions*.

Actors and Roles	**Products and Features**	**Environment and Situation**	**Problems**	**Solutions**
User	Name	Shop	Colored hair	Protecting lips
Manufacturer	Price	Services	Oily skin	Treating skin
Father	Bundle	Retailer	Dry skin	Coloring hair
Desire	Perfume	Time	Oily hair	Painting nails

Table 1. Extract of the cosmetics terminology

Identifying Competency Questions Competency questions (CQs) are conceptual questions that the ontology must be able to answer [23]. They were

CQ1	Does the product fit to me?
CQ2	Does the product solve my problem?
CQ3	How long does the product last?
CQ4	Is my purchase decision correct?
CQ5	Which product can I use for protecting my lips (in winter)?
CQ6	How can I apply the product?

Table 2. Examples of CQs for the cosmetics domain

identified through analysis of scenarios and terminology as well as through brainstorming with domain experts. CQs were used during the test phase of POnA to evaluate the quality of resulting ontologies [17]. Some examples of CQs are

presented in Table 2. The output of the requirements phase consists of CQs, scenarios, and the terminology with corresponding term categories.

3.2 Design Phase

The design phase aims at the identification of semantic structures, more precisely the definition of design patterns for answering CQs. First, relations between terms are identified, which results in coarse term structures. Based on descriptions of situation types, CQs and term structures, prototypical ontology design patterns (PODPs) are derived and formally modeled by reusing ODPs grounded in DOLCE [10]. Complex prototypical ontology patterns require the combination and adaptation of ODPs. Resulting ontology design patterns are discussed with domain experts again. PODPs are informal conceptual structures that are derived from analysis of situations, terms, and CQs. Therefore, PODPs resemble more mental models of real world perceptions [24] than formal logic representations and fit very well to the requirements of AmI environments. PODPs consist of conceptual entities, called scopes, and relations. Scopes are themes that frequently occur in situations, terms and CQs and share a common meaning. The design phase consists of the following activities: *(1) identification of prototypical ontology design patterns, (2) terminology setup*, and *(3) mapping PODPs onto ODPs*.

Identification of Prototypical Ontology Design Patterns In contrast to UPON [17], CQs are mapped onto PODPs that in turn are mapped onto formal ODPs [19, 20, 10]. PODPs and ODPs provide levels of granularity that support discussions with experts much better than discussions of isolated concepts and relationships. Generally, it is proposed that an accurate domain ontology specifies all and only those conceptualizations that are required for answering all the CQs formulated within the requirements phase [10]. The most general CQ for product-centered situations is classified as a problem-solution situation, i.e., "Which solutions exist for this problem?" Through analysis of the 55 CQs, seven POSPs are derived: *Product-Pattern (P), Product-Product-Pattern (PP), Context-Pattern (C), Product-User-Pattern (PU), Product-Information-Pattern (PI), Product-User-Context-Pattern (PUC)*, and *Product-Context-User-Information-Pattern (PUIC)*. Prototypical design patterns conceive a general conceptualization of a set of situations, dominant terms, and corresponding CQs. Furthermore, they highlight candidates for key concepts, called scopes. PODPs have a conceptual or rather architectural nature [25] and arrange the answering of the CQs by scopes according to the categories of the terminology (Actors and Roles, Products and Features, Environment and Situation, Problems and Solutions) (cf. Fig. 1).

The P-Pattern represents solutions to problems concerning the product itself, e.g., its price, whereas the PP-Pattern covers semantics on product bundles and their features. The C-Pattern semantically describes the current context, e.g., time and space of a situation, present objects, and individuals. The PU-Pattern describes information that relates products with user attributes. The

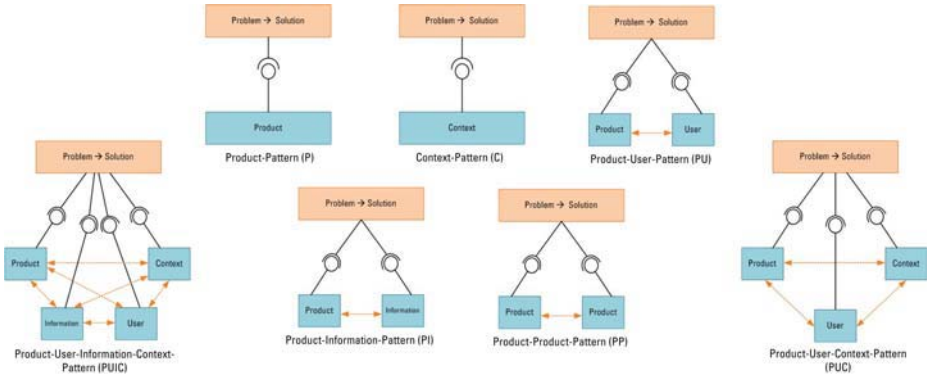

Fig. 1. Design patterns for ambient environments

PUC-Pattern enhances the PU-Pattern with contextual information. Solutions for problems concerning additional information about the product, e.g., user reviews, images, videos, etc. are represented by the PI-Pattern. The most complex pattern, the PUIC-Pattern, represents solutions to problems concerning the matching of information about products to user-specific attributes and needs within a context. All PODPs combine scopes that were extracted from the terminology. The product scope covers the term category *Product and Features* whereas the user scope represents the term category *Actors and Roles*. The context scopes contain terms subsumed by the category *Environment and Situation*. The lollipop relationship between pattern P and pattern C indicate that they conceptually contribute to what is a problem solving situation (cf. Figure 2). More complex PODPs are derived from simpler ones, e.g., pattern PI is an extension of pattern P. Patterns can be composed, which creates new patterns, e.g., the PUIC pattern. By discussion of PODPs, the requirement for an independent information scope appeared that supports CQs such as *"How can I apply this product?"* The information scope refers to information *about* a product, e.g. product images, application videos, and user-generated or professional product reviews.

PODPs dispose a template-based and compact visualization [10], which consists of the following slots:

(a) Graphical visualization of the pattern [19, 10]
(b) Name of the pattern [19, 10]
(c) Description of the intention of the pattern [19, 26]
(d) CQs that are addressed by the pattern [26]
(e) Terms that characterize the pattern
(f) Situations that exemplify a pattern [26]
(g) Consequences, side effects, references to other patterns [19, 10, 26]
(h) Components of the pattern, e.g., product scope and information scope [19, 26]

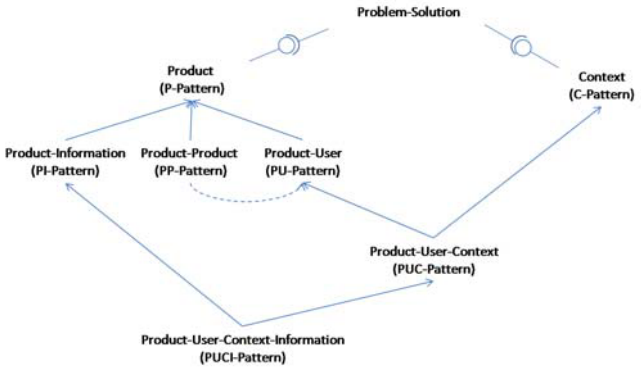

Fig. 2. Tree of design patterns for ambient environments

Slot (e) represents the linkage of the different patterns. For example, the PP-Pattern evolves from the P-Pattern when more than one product is part of a communicative situation within an ambient environment. On the other hand, the PU-Pattern has a strong reference to the P-Pattern representing one product as well as the PP-Pattern focusing on solutions to problems concerning product bundles (cf. Fig. 2).

Terminology Set-up Next, scopes within patterns are refined and structured by arranging relevant terms (cf. Section 3.1) and further analyzing the CQs. For concept identification, a middle-out approach is used that starts with salient concepts and proceeds with generalization and specialization [27]. This seems to be the most effective approach because concepts "in the middle" are more informative about the domain. Table 3 shows the four scopes and an extract of their corresponding terms. The product scope covers all necessary information

Product Scope	**User Scope**	**Context Scope**	**Information Scope**
Name	Characteristic	Space	Animation
Price	Problem	Time	Video
Ingredients	Feeling	Temperature	Image

Table 3. Pattern scopes and exemplary terms

that is part of the product itself, e.g., information about price and material. In contrast, external product information, such as manuals or brochures, is covered by the information scope. The user scope covers terms and questions related to a user whereas the context scope conceives characterizations of a general context.

Mapping PODPs onto ODPs Operations that support the reuse of formal ODPs try to find patterns that optimally cover CQs [16]. In PoNA, PODPs are mapped onto ODPs. Within the cosmetics domain, ODPs proposed by [26] fit well with requirements given by PODPs (cf. Fig. 2). Typically several ODPs are required, which leads to pattern composition. For instance, the ODP mapping of the PI-pattern uses the ODP mapping of the P-pattern based on the *Description and Situation* ODP in conjunction with the information-object ODP [10]. Currently, mapping tasks are executed manually. Our goal is to reduce the complexity of ODP mapping tasks by automatically mapping situations, terminologies, and CQs onto PODPs. With a library of predefined PODP-ODP mappings, we will test ways in which the reuse of formal ontologies can be improved.

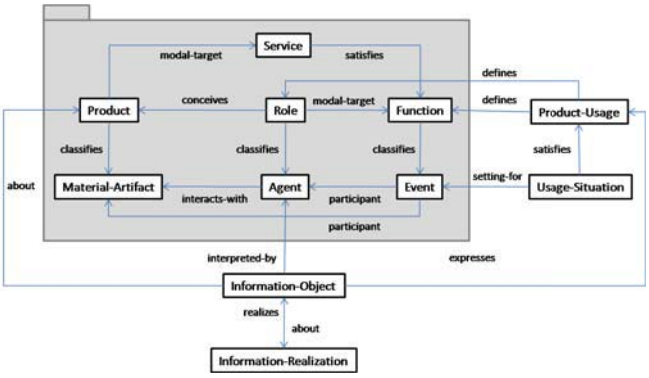

Fig. 3. Merging product and information scopes based on the *Description and Situation* ODP and information-objects ODP

3.3 Implementation Phase

The ODP-based P-pattern and PP-pattern represented in OWL-DL has been integrated into the Smart Product Description Object (SPDO) model that is used in AmI environments [7, 22]. Instances of SPDO models are used as product-centered knowledge bases for Natural Language Processing modules [28] and product reasoning [7]. Currently, we are working on the integration of the other patterns within the EU-project *Interactive Knowledge Stack* (IKS). The resulting SPDO ontology covers all information concerning the product itself. It consists of 25 classes, 86 properties, and 104 restrictions. SPDO answers all the CQs of the P-Pattern and the PP-Pattern. The linkage of two or more product scopes within the PP-Pattern is realized via reasoning based on standardized Web-based rules (SWRL[4]). Statements about alternative or matching products are generated by processing certain concepts of SPDO instantiations while each SPDO describes one particular product.

[4] http://www.w3.org/Submission/SWRL/

3.4 Test Phase

The quality of the resulting ontology is tested with respect to four characteristics: syntactic, semantic, pragmatic, and social quality [17]. First, the semantic quality is evaluated by checking the consistency of the ontology using the Pellet reasoner. Validation of pragmatic quality consists of verifying the coverage of an ontology over a domain and answering CQs. For the cosmetic domain, initial semantic checking of the SPDO showed promising results. Testing the pragmatic quality by answering CQs associated with the P-pattern and PP-pattern was straightforward because users can use the NLP component of SPDO instances for direct communications. Nonetheless, more detailed user studies are required. The syntactic quality is verified within the implementation workflow whereas the social quality can be checked only after application of an ontology in real environments, which is part of the EU-project IKS.

4 Discussion of POnA

When defining the terminology, we found a huge amount of terms with completely different origins, e.g., user or content-specific terms. We therefore decided to structure terms concerning their particular content categories. This additional step was very helpful for creating prototypical ontology design patterns based on scopes. Furthermore, we were able to clearly separate product-centered knowledge from other ontological parts, which is important for AmI environments. Thus, each product could be labled by dedicated semantic product information. Through modularization by scopes and PODPs, we were able to set up a clearly defined ontology engineering process that fits very well with the requirements of AmI environments. To ensure a better portability to other domains and a general representation of the structure of ambient environments, we decided on general scopes within patterns. Domain specifications can be realized by conceptualizations of scopes and PODPs. Additionally, we found that not all scopes can be directly derived from analysis of situations, CQs, and terminologies. At the moment, scope completion requires careful discussions with domain experts. In general, we found that joint consideration of situations, CQs and terminologies is an efficient approach for identifying requirements for ontologies because they help to "pursue the path".

5 Example

Within this section, the *Product-User-Context-Pattern* is exemplified in detail. This PODP was chosen because product, user and context scopes are quite advanced. Table 4 shows the representation of the pattern in the form of a PODP.

The PUC-Pattern represents solutions to problems concerning the matching of products to user-specific attributes and needs within a context, such as whether a vanishing creme should be matched with a specific skin type in a

Graphical visualization	*cf. Fig. 1*
Name	*Product-User-Pattern (PU)*
Description of Intention	Representation of solutions for problems concerning the matching of products to user-specific attributes and needs
Competency Questions	Extract: (a) Does the product /the application of the product solve my problem? (b) Does the product fit to me? (c) Which product can I use for protecting my lips? (d) Will I be happy applying the product?
Characterizing terms	product: price, ingredients, user: emotions, context: temperature, day of week
Situations	Anna is interested in a vanishing creme and wants to know whether it is right for her skin in humid environments.
Consequences / Side Effects / References	PP-Pattern; PUC-Pattern
Components	product scope; user scope: context scope

Table 4. Template of Product-User-Context-Pattern

humid environment. Furthermore, the PUC-Pattern disposes of cross references to other patterns. For instance, when product bundles (more than one product) have to be matched with user-specific aspects, the PUC-Pattern is extended by the PP-pattern. A mapping of the PUC-PODP onto ODPs is illustrated in Fig. 4.

6 Conclusion and Future Work

Ambient Intelligence implies modularized environments of computing and specific interfaces. The characteristic of such an environment requires systems to deal with large amounts of unstructured information from heterogeneous sources and to support dynamic knowledge sharing and reasoning. These issues imply the application of appropriate knowledge representations. We start from the outset that ontologies are an efficient means for building ambient environments because they enable efficient sharing, adding and changing of information, and inference generation [2]. By leveraging capabilities of different ontology design methodologies, we investigated *Pattern-based Ontology Building Method for Ambient Environments (POnA)* as a method with particular focus on ambient environments. We found that a combination of systematic methodologies and different types of design patterns can be an efficient approach to designing ontologies for ambient environments. The POnA process focuses on the main ontology development processes and enables an explicit pattern-based modularization and abstraction of scopes. In our future work, we will proceed with several tasks: (1)

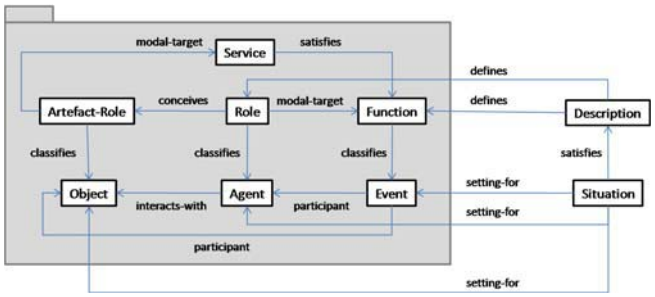

Fig. 4. ODP-based PUC-pattern

evaluation of PODPs and their formalizations based on ODPs, (2) extension of PODPs to more AmI-specific dimensions, i.e., representation of spatio-temporal information, frames of reference, and sensoric perceptions, and (3) automatic mapping of CQs, situations, and terms onto PODPs.

7 Acknowledgements

We would like to thank Andreas Filler and Tobias Kowatsch for their help on this paper. This work is part of the project SmaProN that is being funded by the Federal Ministry of Education and Research (BMBF), Germany, under the number FKZ 17 53X 07.

References

1. Dey, A.K.: Providing architectural support for building context-aware applications. PhD thesis, Georgia Tech College of Computing, Atlanta, GA, USA (2000) Director-Abowd, Gregory D.
2. Coen, M., Phillips, B., Warshawsky, N., Weisman, L., Peters, S., Finin, P.: Meeting the computational needs of intelligent environments: The metaglue system. In: Proceedings of MANSE99, Springer-Verlag (1999) 201–212
3. Schilit, W.N.: A system architecture for context-aware mobile computing (1995)
4. Want, R., Hopper, A., Falcao, V., Gibbons, J.: The active badge location system. ACM Trans. Inf. Syst. **10**(1) (1992) 91–102
5. Peters, S., Shrobe, H.E.: Using semantic networks for knowledge representation in an intelligent environment. In: PERCOM '03: Proceedings of the First IEEE International Conference on Pervasive Computing and Communications, Washington, DC, USA, IEEE Computer Society (2003) 323
6. Chen, H., Finin, T., Joshi, A.: The SOUPA ontology for pervasive computing. In Tamma, V., Cranefield, S., eds.: Ontologies for Agents: Theory and Experiences. Springer (2005) 233–258
7. Maass, W., Filler, A.: Towards an infrastructure for semantically annotated physical products. In Hochberger, C., Liskowsky, R., eds.: Informatik 2006. Volume P-94 of Lecture Notes in Informatics., Berlin, Springer (2006) 544–549

8. Minsky, M.: A framework for representing knowledge. In Winston, P.H., ed.: The Psychology of Computer Vision. McGraw-Hill, New York (1975) incollection.
9. Schank, R.C., Abelson, R.P.: Scripts, Plans, Goals and Understanding. Erlbaum, Hillsdale, NJ (1977)
10. Gangemi, A.: Ontology design patterns for semantic web content. In: M. Musen et al. (eds.): Proceedings of the Fourth International Semantic Web Conference, Berlin, Springer (2005)
11. Maass, W., Varshney, U.: A framework for smart healthcare situations and smart drugs. In: SIG-Health Pre-AMCIS Workshop at the 15th Americas Conference on Information Systems (AMCIS 2009), San Francisco, USA (2009)
12. Svátek, V.: Design patterns for semantic web ontologies: Motivation and discussion. In: Proceedings of the 7th Conference on Business Information Systems, Poznan (2004)
13. Corcho, O., Fernández-López, M., Gómez-Pérez, A.: Methodologies, tools and languages for building ontologies: where is their meeting point? Data Knowl. Eng. **46**(1) (2003) 41–64
14. Uschold, M., King, M.: Towards a methodology for building ontologies. In: In Workshop on Basic Ontological Issues in Knowledge Sharing, held in conjunction with IJCAI-95. (1995)
15. Fernández-López, M., Gómez-Pérez, A., Juristo, N.: METHONTOLOGY: from ontological art towards ontological engineering. In: Proceedings of the AAAI97 Spring Symposium Series on Ontological Engineering, Stanford, USA (March 1997) 33–40
16. Suárez-Figueroa, M.C., Blomqvist, E., D´Aquin, M., Espinoza, M., et al., A.G.P.: D5.4.2. revision and extension of the neon methodology for building contextualized ontology networks. Technical report, NeOn: Lifecycle Support for Networked Ontologies (2009)
17. De Nicola, A., Missikoff, M., Navigli, R.: A software engineering approach to ontology building. Inf. Syst. **34**(2) (2009) 258–275
18. Grueninger, M., Fox, M.S.: Methodology for the design and evaluation of ontologies. In: Proceedings of the Workshop on Basic Ontological Issues in Knowledge Sharing held in conjunction with IJCAI-95. (1995)
19. Alexander, C.: The timeless way of building. Oxford University Press, New York (1979)
20. Clark, P., Thompson, J., Porter, B.W.: Knowledge patterns. In Staab, S., Studer, R., eds.: Handbook on Ontologies. International Handbooks on Information Systems. Springer (2004) 191–208
21. Gangemi, A., Guarino, N., Masolo, C., Oltramari, A., Oltramari, R., Schneider, L.: Sweetening ontologies with DOLCE. In: Proceedings of the 13th International Conference on Knowledge Engineering and Knowledge Management. Ontologies and the Semantic Web, Springer (2002) 166–181
22. Janzen, S., Maass, W.: Smart product description object (SPDO). In: Poster Proceedings of the 5th International Conference on Formal Ontology in Information Systems (FOIS2008). IOS Press, Saarbrücken, Germany (2008)
23. Grueninger, M., Fox, M.S.: The role of competency questions in enterprise engineering. In: Proceedings of the IFIP WG5.7 Workshop on Benchmarking - Theory and Practice. (1994)
24. Johnson-Laird, P.N.: Mental Models: Towards a Cognitive Science of Language, Inference, and Consciousness. Cambridge University Press (1983) book.
25. Gangemi, A., Presutti, V.: Ontology design patterns. http://ontologydesignpatterns.org (2008)

26. Presutti, V., Gangemi, A.: Content ontology design patterns as practical building blocks for web ontologies. In: ER '08: Proceedings of the 27th International Conference on Conceptual Modeling, Berlin, Heidelberg, Springer-Verlag (2008) 128–141
27. Uschold, M., Gruninger, M.: Ontologies: Principles, methods and applications. Knowledge Engineering Review **11** (1996) 93–136
28. Maass, W., Janzen, S.: Dynamic product interfaces: A key element for ambient shopping environments. In: Proc. of 20th Bled eConference, Bled, Slovenia (2007)