A Conceptual Model for Context-aware Web Engineering

J. Wolfgang Kaltz and Jürgen Ziegler

IIS, Universität Duisburg-Essen (Campus Duisburg), 47057 Duisburg, Germany
Email: {kaltz,ziegler}@interactivesystems.info

Abstract. This paper presents a conceptual model which aims to account for context-sensitivity in a comprehensive and integrated fashion for Web engineering processes. The model permits the combination of a domain ontology with context-relevant parameters, with a degree of relevance. In the subsequent development of the Web application, the Web engineer can then choose to make use of the context-sensitive model where it is deemed useful, and at varying levels: navigation, content display and services presented to the user.

Keywords: Web-engineering, context model, domain ontology, service integration

1 Introduction

The next generation of Web software holds the promise of "mass customization" ([1]). If we look at customization of software as its ability to fulfill an individual user's needs, we realize that the software must be aware of several factors: the user's profile, her current task or goal, and possibly additional factors such as location, time, or device used. The combination of all relevant factors can be termed the "context", and thus a Web software which takes them into account is a context-aware application.

Several approaches to integrating such context-awareness within software applications have been used in the past. Hypermedia applications have typically focused on the areas of learning and information retrieval, with the goal of guiding the user's navigation and presenting relevant information. An overview of this field of research, usually called adaptive hypermedia, is given in [2]. In this field, the system adapts its offering according to the user model, where this user model is generally constructed from the user's behavior. Adaptivity in this sense takes into account user categories, sometimes also the types of tasks ([3],[4]). Another recent field of development in Web applications with respect to context-sensitivity has focused on providing context-relevant information to mobile users, usually based on their location as context information. A typical scenario is a touristic visit in which the user is guided by a mobile device such as a PDA (see e.g. [5]). Chen et al. ([6]) provide a survey of context-aware mobile computing research. These applications use a context model which is specific to mobile scenarios. Research such as [7] does suggest that other types of context, such as *domain context*, are also relevant to mobile applications, but does not explore this issue in detail.

We believe that context modeling is potentially useful in all types of applications, and that therefore a more general model, including mechanisms for context integration and usage, which would cover a wide variety of application domains, would be beneficial. Attempting to integrate context and adaptation capabilities in a comprehensive approach poses a number of interesting challenges. An integrated conceptual model based on a multitude of context factors is needed. This raises the question of how to structure and systemize the analysis and design processes. There is the risk of high complexity resulting from a large number of interactions between context factors. Generally speaking, modeling context is a difficult and time-consuming task, so an important question is how it can be made more effective, with the help of an appropriate base model and tools.

Our specific focus is the area of Web engineering, which advocates a process and a systematic approach to development of Web-based systems ([8]). We will aim to provide for context modeling in a more general way than for mobile scenarios only, as we believe the benefits of context models can apply in general to Web applications, whether they are used by mobile devices or not. In fact, future Web applications should attempt to provide services for all sorts of devices, the device being just an additional context factor. In this article, we will first present an approach to modeling context on a broad base, integrated with other elements of the application domain; subsequently we will discuss how this model may be used within a Web engineering process. We will also discuss design issues of a corresponding context system. Finally, we will discuss related work and present an outlook.

2 Context Modeling in Web Engineering

Recent studies have shown that a systematic engineering approach is often lacking in the development of Web software [9], leading to negative effects such as limited reusability and difficult maintainability. The focus of current research (see, e.g., [10]) is thus to provide a methodology and tools for systematic engineering of Web applications. Such efforts are often based on establishing an ontology, which represents the terms and relations of the domain relevant to a specific application.

In our modeling approach, the domain ontology plays a central role and may already have been developed in the overall engineering process. The goal is then to connect the domain ontology with context information. Context parameters are classified and connected with a certain weight to elements of the domain ontology, resulting in a *relevance space*. This approach is detailed in this section; the subsequent section will focus on how this model may be used by a Web application.

It should be noted that, although we shall aim at having as big a part as possible of the model generated by the system, we consider it indispensable for an expert user to be able to validate and extend the model. Therefore the model will need to be flexible with regard to representing context information, and yet remain comprehensible by an expert user. The goal of attaining this compromise will motivate the decisions presented here.

2.1 A General View of Context

Our general notion of context is that elements of the conceptual design space of an application, such as concepts of the domain ontology, can be contextualized through relations to elements of the context space. A certain information object may be, for example, particularly relevant for a specific geographic region or location. This relation is reciprocal: the information object is relevant if the current context is determined by the specific location; conversely, a specific location may be relevant if the current context focus is on a particular information object.

Context is thus determined by a set of context factors and

- a current perspective on these factors,
- an integrated ontology,
- a structuring of the context factors.

2.2 Categorization of Context Elements

The question of how to integrate context elements, or parameters, within an application model can be approached in two different ways. First, it is conceivable to allow for arbitrary definition of context parameters, and arbitrary combination with elements of the domain ontology. This would provide for maximum flexibility in the linking of context parameters and domain elements, and ensure that the model has a theoretically unlimited expressive power.

The second approach is to define a categorization of context parameters, and require specific values relevant to the application domain at hand to be assigned to one of these classes. This would seem to be a major restriction for the conceptual model, however, for the practical usage, we postulate that such a grouping of parameters is unavoidable, from the modeling perspective and from the usage perspective. Since we assume that some form of human interaction will be required to complete and adjust the model, the grouping of parameters is indispensable for the user to maintain an overview of the model. On the other hand, we assume that not all context elements will be linked with the domain ontology. This means that there will be a concept of "neighborhood" between context elements, such that certain elements will be relevant in a given context, even though they are not directly linked to the domain. In order to achieve a neighborhood, some form of categorization is required. Furthermore, such a categorization will allow for a large degree of re-use, through the possibility of context catalogues.

Given that we wish to classify context elements, we now should ask which, and how many, classes are useful. Existing approaches in mobile scenarios, as described above, usually focus on one or two classes, which are key to the scenarios they wish to cover: these are typically *Location* and/or *Device*, sometimes *Time*. The limitation to one or two classes has the advantage of ensuring that the model will remain understandable, however is less expressive and limits the combination possibilities. For context modeling representing a broad variety of Web application scenarios, we propose the following categories of context parameters:

- User & Role: a categorization of users according to their role, such as various types of customers, or different types of employees.
- Process & Task: the functional context, such as work items for employees.
- Location: a categorization of locations relevant to the application, in the desired granularity: for some applications, the country may be sufficient location information, for others, the city, and so forth. These locations are not to be confused

with the locations *sensed* by the system, such as by GPS positioning, user input, or network address (IP address or the like). Furthermore, this location categorization does not refer to locations in the information space (i.e. where is a certain Web media located) nor their proximity location-wise to other elements in this space: in our terminology, this type of relationship is part of the domain ontology, not of the context dimensions.

– Time: different types of time information may be relevant, such as the time-zone of the client, the actual time, a virtual time, etc.
– Device: device information may be a relevant context parameter (such as in mobile scenarios), e.g. the device type, display properties, etc.

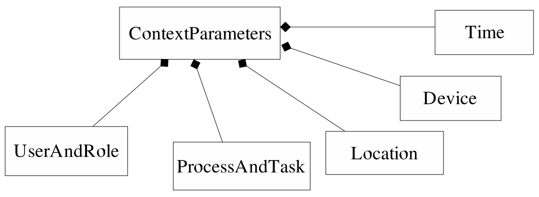

Fig. 1. Context parameter classes

2.3 Modeling Activity

The modeling for a specific application domain will consist of linking elements of the domain ontology with elements of the context model, and assigning a relevance factor to this link. We term the resulting model the *relevance space* (an example is provided in the next section).

Elements of context are also linked to services which are available to the Web application, or, more precisely, to service descriptions. By service, we understand a mechanism through which the application will provide the user with a dynamic offering: e.g., a reservation service through which the user can book a flight. Typically, the Web application would offer such a service to the user, and to implement the service would rely on an actual, back-end service, which is either locally implemented or is a third-party offering. These services could in turn be Web services, for which the Web application would act as consumer, but could in fact be supported by various back-end technology. Thus, in our model, service descriptions are used for describing a service which is relevant in the given context.

Note that not all parts of the context model need be explicitly relevant in the modeling process. However, they remain available to the application, and may become a part of the decision-process even if they are not directly linked to the domain resources.

The actual context parameters might come from various sources:

– predefined context catalogs, such as categories of time like spring, summer, fall and winter; geographic zones; user roles typical to certain business domains; etc.

- the domain ontology, as it may already contain elements pertaining to context. For instance, geographic zones might already be represented in the ontology, if inherent to the business domain.
- custom additions by experts of the domain.

Our aim is that much of the model will be automatically generated using the available sources, and then fine-tuned by a business expert. This generation activity is however not the focus of this article.

2.4 Definition of Context

Given the above categorization of context parameters, the context space C may be defined as the combination of context parameters, domain ontology elements and service descriptions:

$$C = \{U, P, L, T, D, I, S\} \tag{1}$$

where U is the set of user & role factors, P the processes & tasks, L the locations, T the time factors, D the device factors, I the available information items, and S the available services (or service descriptions).

The relevance space R can then be defined as the product of the context space with a relevance factor:

$$R = C \times \mathbb{R} \tag{2}$$

Relevance factors are associated with items in the hierarchy, i.e. context parameters and elements from the domain ontology, including service descriptions. Sub-items in the hierarchy implicitly carry the same factor, unless overridden by a more specific value.

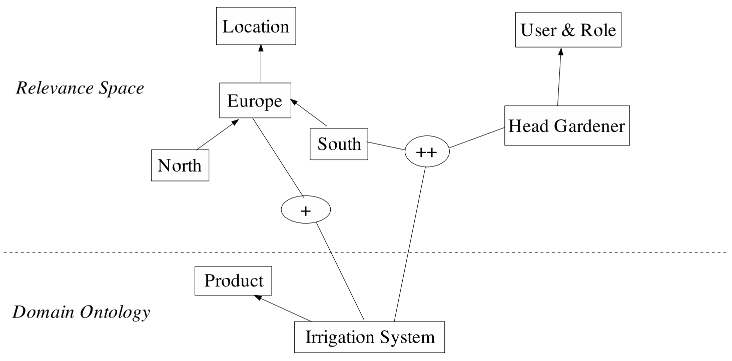

Fig. 2. Relevance space: an example

Figure 2 provides an example of a context model and the associated relevance space. Consider a construction market servicing various geographical areas (as an online store and/or as actual outlets). A product from the domain ontology is modeled to be relevant (marked as $+$) in a geographical area. All sub-areas, i.e. sub-items of the hierarchy are implicitly of the same relevance. Furthermore, within a more specific geographic area, and for a specific user profile, the element of the domain ontology is modeled to be even more relevant (marked as $++$), making it more susceptible in appearing e.g. in a customized offering.

As shown in this example, the business user may model the relevance factors in the granularity of her choosing; not all elements need be explicitly linked with relevance factors. However, the elements' presence is useful even if not explicitly linked: such elements can nonetheless carry a relevance factor, by their proximity in hierarchies to elements which *are* linked. Note also that although our definition of context allows for arbitrary relevance values, we presume that a higher-level definition as shown in the example will yield more meaningful results, given that the relevance factors will need to be verified by a business expert in a sensible manner.

At this stage of context modeling, the services to be integrated are viewed as simple domain elements, in analogy to elements of the information space. A service can be associated to the context parameter hierarchy, with a certain relevance, just as the other elements. A more detailed specification of the service will later be necessary in order to provide a meaningful mapping of application parameters (such as a GPS location) to a context parameter used in the model (such as a geographical zone relevant to the current business domain). [11] gives an approach in particular for specifying Web services such as to integrate context information, focusing on the environmental context. The hierarchy of location parameters might also be modeled by an approach such as ”semantic location” ([12]).

2.5 Practical Considerations

The availability of several dimensions in the context model obviously raises issues of practicability. Potentially, an item of the relevance space may need to be displayed in as many dimensions as there are parameters, making this display very difficult to interpret. Yet, a useful interpretation for a user is necessary, as we believe user adjustments to the model should be possible at any time. We postulate that the modeling in such a manner is nonetheless useful: a given relevance link will usually not be a link between all dimensions, but more likely between two or at most three, and recent research offers new tools such as [13] which assist in visualizing a multi-dimensional information space. Appropriate visualization is a topic of current work.

The issue of precision of the relevance setting is also important, and was introduced by the example in Fig. 2. In principle, it is necessary to maintain the possibility of arbitrary precision for a relevance setting: a relevance setting need not be explicitly specified by the user, but could either be the result of automated extraction from company resources, or be a result of an item's proximity to other items which do have explicit relevance factors. For the latter, an algorithm will compute the actual relevance factor. For such items for which the *user* will want to set a relevance factor, the possibility of entering a high precision value (such as a probability value) may be counter-productive, as it is more likely to result in confusion and errors. Instead, the business expert might specify the relevance factor at a ”high-level”, perhaps in five categories: $++$ meaning very highly relevant, $+$ meaning highly relevant, no marking meaning somewhat relevant, $-$ meaning rather irrelevant, $--$ meaning totally irrelevant. A $-$ or $--$ setting would indicate items which are inappropriate in a given context, and thus may bother or even offend the end-user if presented in that context.

Modeling relevance relations between context parameters and domain ontology elements is conceivably a lengthy task, which we believe can be considerably supported by the system, through mining of the data stock available at the customer's site. One approach would be to look for co-occurrences and set relevance factors according to statistical values. The expert user would then be asked to verify the accuracy of the model. For an applied example of generation of a concept network through data mining, see [14]. Orthogonal to data mining, catalogues of context parameters can be established, some domain independent (such as e.g. geographic locations), others business domain specific, such as typical user roles for a specific business domain. One approach which may be useful in this regard, particularly if business rules are also to be represented, is "adaptive object-models" (see [15]).

3 Model Use

Once defined, the contextual model is available to the remainder of the Web engineering process. One can distinguish between two types of approaches: complete integration or part-wise integration. By complete integration, we understand that all parts of the Web application would be context-sensitive, i.e. all menu navigation and information presented would be generated according to context. In a part-wise integration, the application would in some parts be context-independent, and in other, selected parts, present context-sensitive menu navigation and information.

We believe that, typically, not all parts of a Web application will require a context-sensitive model. On the contrary, for some aspects adaptation may be counter-productive. Therefore it is an issue of the Web engineering process to determine in which parts, which types of context-sensitive resources will be useful. It should be noted that, in any case, context parameters will need to be modeled in an integrated fashion with the application domain if a useful usage of context factors is to be achieved: in this light, the decision whether a Web application should be fully or partly context sensitive is secondary.

3.1 Context-sensitive Resources

Given specific context parameters (such as described in the previous section), the system will provide resources which are deemed to be relevant in this context - these are what we term the context-sensitive resources. Reciprocally, given a specific resource, the system can provide context factors for which this resource is deemed to be particularly relevant - this information could then be used e.g. to search for other resources relevant in the context.

We distinguish three types of context-sensitive resources (see Fig. 3):

– $Navigation$ refers to navigation possibilities which are relevant to this context, through a menu or through other types of links. For instance, the context-model can

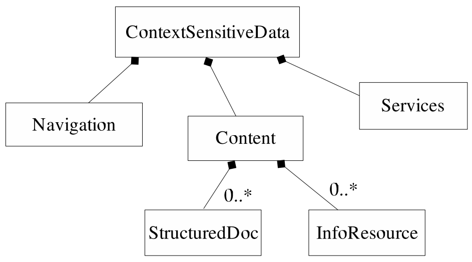

Fig. 3. Classes of context sensitive data (output)

provide links which are probably of interest, thus permitting adaptive navigation, where it is desired.

– $Services$, in analogy to $Navigation$, refers to available services relevant to this context. Service description for relevant services can be used to dynamically search for available, relevant services. Such a service can be presented to the user, leaving it up to her whether to execute this service; but such a service could also be transparently executed, e.g. to dynamically retrieve information without requiring user interaction.
– $Content$ refers to documents or other types of structured information which are relevant in a given context and can be retrieved. The structuring could consist of Doc-Book or similar XML mark-ups, thus enabling a straightforward adapted presentation through the use of stylesheets. Another type of content elements would be arbitrary information resources, such as office documents, which the user could then choose to retrieve and open as a whole.

3.2 Precision and Filtering

Given that there is a categorization of context parameters in the model (see above) and thus a concept of neighborhood, a request to the context model for relevant resources may have different results. The request may be more or less granular; typically not only exact matches will be wanted, but also neighboring resources. Since the relevance space is linked with a relevance factor, any element of the context space will have a certain relevance factor. We propose the use of a *context filter* or *context lens*, which will have a certain relevance setting; according to the setting, those elements of the context space will be matched which have a relevance factor equal to, or greater then the lens setting. Once the conceptual model has been established as described above, several inference mechanisms for the generation of the adaptation model might be used; however, the issue of inference is beyond the scope of this paper and a subject for future work.

4 The Context System

In our Web-engineering scenario, the context system is an independent module, which may be “plugged” into the Web software where deemed useful by the engineering process. Indeed, experience, most notably in the adaptive hypermedia field, has shown that adaption, respectively context-sensitive behavior, is not useful in all situations and can in fact be counter-productive. The context system we propose is meant to integrate with a Web engineering process which includes the creation (or usage) of a domain ontology. The context system also uses this domain ontology, upon which the relevance space is based (see above). The application design process will then determine at which points context-sensitive resources are useful. The application would thus provide the context system with current parameters, and receive in return the desired context-sensitive resources.

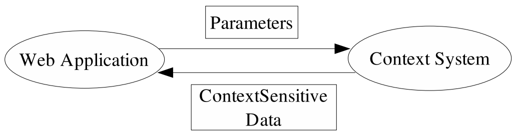

Fig. 4. Context system (black-box view)

Figure 4 gives a simple, black-box type view of the context system. What is noteworthy here is that context-relevant parameters as given by the Web application are not identical to the context parameter modeling - the useful matching is a task of the context engine. The context engine returns to the Web application the desired context-sensitive resources (see above).

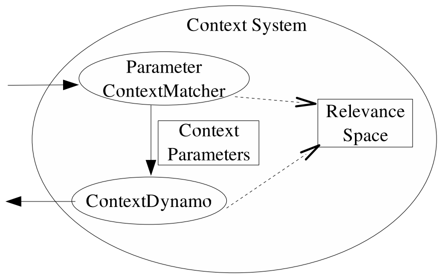

Fig. 5. Context system - interior overview

Figure 5 provides an overview of the context system's interior workings. Parameters provided by the Web application can be of two types. The first type are elements of the domain ontology: given that the Web application is built upon a domain ontology, it ”knows” which elements of the ontology are currently relevant. One basis for this knowledge is ontology-based navigation, where parts of the menu navigation structure were constructed from an ontology, and the user has navigated through such a structure. This information is provided as parameters to the context system. The second type of parameters are classical Web application parameters, such as client device parameters. Such parameters are transformed into context parameters as known by the context engine (defined here as $ContextParameters$). For instance, the Web application may provide a time context in date, hours and minutes; however in the relevance space only higher-level distinctions of time, such as spring, fall etc. are used, so this parameter is transformed accordingly. The same mappings may be needed for the other types of parameters as well. The context parameters are then passed on to the $ContextDynamo$, which determines which resources are relevant in this context, according to the $RelevanceSpace$ and the current context-filter setting. These resources are then returned to the Web application, which may then present them to the user.

5 Related Work

The goal is to provide an environment for the development of Web software which takes into account context in a comprehensive and integrated fashion; to achieve this goal, one can distinguish two types of approaches.

Using a network approach, the network in itself would hold knowledge about which information and which services are relevant in which context. This network approach is explored e.g. by ([16]), who focus on providing context-relevant information and services through a smarter network. The client, a mobile device, sends “Context-aware-packets” to the network. Within the network, nodes listen for such packets and respond to them, e.g. by presenting a relevant service to the client, which may then choose to activate it. The scenario presented is a mobile user attempting to discover a printer she may use, and has physical access to, from the present location. This type of approach has the disadvantage of requiring additional infrastructure for its realization and thus seems less suited for describing a general Web engineering approach. It is, however, potentially better suited to the specific challenges of mobile computing ([17]); it could thus be interesting to integrate such an approach for better support of mobile clients.

The other approach is data-centric, meaning that context data needs to be modeled completely at the application level, and made available to the Web application. Thus, relevant context data is modeled in the Web application environment, and within the Web application certain decisions will be based on this information. Cannataro et al. ([18]) is an example of such an approach; it defines XAHM, which has three conceptual adaptivity dimensions: user’s behavior, external environment, technology. A context in this model is a specific point within these three dimensions; although additional “dimension parameters” might be added. Another recent development is the COBRA-ONT architecture described in ([19]), which has the goal of supporting context-aware systems in smart spaces. This research, like ours, is also based on the observation that previous systems often lack a formalized model describing the contextual knowledge. The COBRA-ONT focuses on OWL ([20]) as the key element for pervasive context-aware systems.

Our approach is also data-centric, and has similarities to XAHM in that it attempts to model context information in a general, integrated manner. A significant difference in our approach is the integration within a general Web engineering activity, specifically with a domain ontology. Furthermore, our goal of service integration would allow for inclusion of context relevant services and/or information on a *discovery* basis, thus overcoming somewhat the limitation of the data-centric approach. We further agree with

([19]) that OWL is a good candidate for specifying an interchangeable format for storing contextual knowledge; however we attempt to place the focus on the definition and usage of a context model, whereas OWL to us represents a means of representation of this information.

From a modeling perspective, the approach of modeling terms in connection with a relevance value bears a resemblance to Bayesian networks, in which terms can be connected with probabilistic values. Tipping ([21]) for instance has used Bayesian networks to associate relevance values, though for the purpose of learning algorithms. Such networks may in fact provide a basis for implementation for us; however, we believe that for a business user's perspective, our - more specific - modeling approach has a higher chance of being understood, and, therefore, of being used effectively in a Web engineering process.

6 Outlook and Future Work

We believe the development of an integrated conceptual model for context-aware Web engineering to be a promising approach: from an engineering perspective, the integration with a domain ontology can enable the systematic development of a Web application for which context factors may be used; it can support the automatic generation of the context model in a significant proportion; and it can lead to the creation of context catalogs pertinent for specific business domains. It is thus reasonable to assume a meaningful degree of re-usability may be achieved not only for the domain ontologies, but also for the context models. From a usability perspective, the end-user can be presented with relevant information and menu navigation possibilities, and furthermore can interact with available, relevant services, thus providing for Web applications better tailored to the user's needs and, hopefully, raising the productivity of people working with these applications.

Future work will focus on the following issues: (1) detailed definition and prototyping of the relevance space, (2) to what degree can the context-sensitive conceptual application model be automatically generated, and with what kind of tools? (3) which representation(s) are adequate for the context network? (4) the development of useful context catalogs, depending on application domains (e.g. business catalogs, geographical catalogs, etc.)

References

1. Rheingold, H.: Smart Mobs: The Next Social Revolution. Perseus Publishing (2002)
2. P. Brusilovsky, A.K., Vassileva, J.: Adaptive Hypertext and Hypermedia. Kluwer Academic Publishers (1998)
3. Koch, N.: Software Engineering for Adaptive Hypermedia Systems: Reference Model, Modeling Techniques and Development Process. PhD thesis, Ludwig-Maximilians-University Munich (2001)
4. Staff, C.: Hypercontextr: A model for adaptive hypertext. In Jameson, A., Paris, C., Tasso, C., eds.: Proceedings of the Sixth International Conference on User Modeling (UM97). Springer, Berlin (1997) 33–39 (Chia Laguna, Sardinia, Italy).

5. Cheverst, K., Davies, N., Mitchell, K., Friday, A., Efstratiou, C.: Developing a context-aware electronic tourist guide: some issues and experiences. In: Proceedings of the SIGCHI conference on Human factors in computing systems, ACM Press (2000) 17–24
6. Chen, G., Kotz, D.: A survey of context-aware mobile computing research. Technical Report TR2000-381, Dept. of Computer Science, Dartmouth College (2000)
7. Dix, A., Rodden, T., Davies, N., Trevor, J., Friday, A., Palfreyman, K.: Exploiting space and location as a design framework for interactive mobile systems. ACM Trans. Comput.-Hum. Interact. **7** (2000) 285–321
8. Murugesan, S., Deshpande, Y., Hansen, S., Ginige, A.: Web engineering: A new discipline for development of web-based systems. In Murugesan, S., Desphande, Y., eds.: Web Engineering. Managing Diversity and Complexity of Web Application Development. Volume 2016 of LNCS., Berlin, Heidelberg, Springer (2001) 3 ff
9. Barry, C., Lang, M.: A survey of multimedia and web development techniques and methodology usage. IEEE MultiMedia **8** (2001) 52–60
10. Wissen, M., Ziegler, J.: A methodology for the component-based development of web applications. In: Proceedings of 10th Int. Conf. on Human - Computer Interaction (HCI International 2003), Vol. 1, Crete, Greece. (2003)
11. Strang, T., Linnhoff-Popien, C., Frank, K.: Applications of a Context Ontology Language. In Begusic, D., Rozic, N., eds.: Proceedings of International Conference on Software, Telecommunications and Computer Networks (SoftCom2003), Split/Croatia, Venice/Italy, Ancona/Italy, Dubrovnik/Croatia, Faculty of Electrical Engineering, Mechanical Engineering and Naval Architecture, University of Split, Croatia (2003) 14–18
12. Pradhan, S.: Semantic location. Personal Ubiquitous Comput. **4** (2000) 213–216
13. Ziegler, J., Kunz, C., Botsch, V.: Matrix browser: visualizing and exploring large networked information spaces. In: CHI '02 extended abstracts on Human factors in computing systems, ACM Press (2002) 602–603
14. Staudt, M., Kietz, J.U., Reimer, U.: A data mining support environment and its application on insurance data. In: Knowledge Discovery and Data Mining. (1998) 105–111
15. Yoder, J.W., Balaguer, F., Johnson, R.: Architecture and design of adaptive object-models. SIGPLAN Not. **36** (2001) 50–60
16. Samulowitz, M., Michahelles, F., Linnhoff-Popien, C.: Adaptive interaction for enabling pervasive services. In: Proceedings of the 2nd ACM international workshop on Data engineering for wireless and mobile access, ACM Press (2001) 20–26
17. Forman, G.H., Zahorjan, J.: The challenges of mobile computing. IEEE Computer **27** (1994) 38–47
18. Cannataro, M., Cuzzocrea, A., Pugliese, A.: Xahm: an adaptive hypermedia model based on xml. In: Proceedings of the 14th international conference on Software engineering and knowledge engineering, ACM Press (2002) 627–634
19. Chen, H., Finin, T., Joshi, A.: An Ontology for Context-Aware Pervasive Computing Environments. Special Issue on Ontologies for Distributed Systems, Knowledge Engineering Review **18** (2004) 197–207
20. Sean Bechhofer, et al: OWL Web Ontology Language Reference. Recommendation, http://www.w3.org/TR/2004/REC-owl-ref-20040210/, XML Protocol Working Group (10 February 2004)
21. Tipping, M.E.: Sparse bayesian learning and the relevance vector machine. J. Mach. Learn. Res. **1** (2001) 211–244