CROCO: Ontology-Based, Cross-Application Context Management

Stefan Pietschmann Annett Mitschick Ronny Winkler Klaus Meißner
Chair of Multimedia Technology
Technische Universität Dresden, Germany

Abstract

Context-awareness is a key issue for future applications within heterogeneous and networked environments. In terms of efficiency and reusability, such applications should be separated from the problems of context gathering and modeling, but should instead profit from and contribute to cross-application context information. For this purpose, an ontology-based, cross-application context modeling and management service is needed to provide appropriate support for the variety of conceivable application scenarios. Although there have been numerous approaches dealing with application-independent context management, none of them sufficiently supports the vision of cross-application context handling. Therefore, we present CROCO, an ontology-based context management service that allows for cross-application context gathering, modeling, and provision. We verified and tested the practicability of our concept within two different projects with disparate application scenarios.

1. Introduction

Context-awareness and context-aware applications have been the focus of extensive research work in recent years. Accordingly, a number of solutions for context gathering and modeling have been suggested [1, 3] or already found their ways into practice. A context managing solution has to take account of appropriate support for context providers (e.g. software agents, applications, or sensors, which supply context data) and context consumers (e.g. applications, which utilize context data for specific purposes). The majority of the proposed approaches exclusively fulfil the requirements of the according research projects and application scenarios, i.e. they support only specific context providers and context consumers or use a specialized domain model. Gathering, modeling, and supply of context information across application boundaries has still not been designed, implemented, and tested for and within heterogeneous scenarios. Thus, the vision of cross-application context-awareness based on semantic technologies has not been fulfilled, yet.

In this paper we therefore present CROCO, an ontology-based, cross-application context management service. In Section 2 we discuss relevant work related to ontology-based cross-application context modeling. The design of CROCO is presented in Section 3. After a discussion of the implementation details, Section 4 illustrates the practicability of our approach with the help of two “orthogonal” application scenarios: personal document management and adaptive co-browsing applications. Section 5 concludes this paper and suggests future research directions.

2. Related Work

A lot of research has been carried out regarding context modeling and management, especially in the fields of Adaptive Hypermedia [9] and, more recently, Ubiquitous Computing [2]. In this research process the means for context representation have changed from basic key-value models to logic- and ontology-based models, which provide much better support for aspects like partial validation, disambiguity or applicability. We will not discuss these advantages in detail here, as this has already been extensively done, e.g. in [18]. In the following we will shortly present related concepts for ontology-based context management.

Context infrastructures are usually similar in the components they consist of, which resemble the basic tasks of the system (e.g. management, inferencing and storage). However, they differ in important additional features, such as a Context History and Security support, as well as regarding their architectural style.

CoBrA (Context Broker Architecture) [4] and DAIDA-LOS [16] are two powerful approaches that – similar to our concept – comply to the Blackboard model using a central component for storage, management and provision of context data [1]. Both are designed to support smart spaces, though, and are therefore not suited for our application scenarios. Furthermore, they either do not consider privacy as-
pects (DAIDALOS) or provide a context history (CoBrA), which is crucial for sophisticated context reasoning.

In [6] a context framework for MobiLife is presented that is similar to our concept in using a provider-consumer metaphor and facilitating easy access via a service interface. However, its distribution among a network implies that there is no central context history and additional effort regarding security needed. Another promising distributed context management concept proposed in [14] differs from common approaches by not using a specific ontology. Instead, during registration, Context Services (providers) exchange their domain models and corresponding mapping rules. When context data is requested, these rules are used to match the request with the correct provider and map its data automatically to the requested model. In contrast to our approach, to benefit from cross-application context authors here need to define mapping rules and are therein limited to previously known concepts.

SOCAM (Service-Oriented Context-Aware Middleware) [8], based on a hybrid architectural style with distributed components and central server, seems to be the closest to our approach, as it allows for cross-application context management and exchange by subclassing a domain-independent ontology. Opposed to our concept, this does not include time-related concepts for describing context dynamics and hence does not provide a context history.

3. CROCo: An Ontology-based, Cross-Application Context Service

In this section we present our concept of an ontology-based, cross-application context modeling and provision service, called CROCo. First of all, we specify preconditions and requirements for its design, followed by a presentation of the overall concept.

3.1. Requirements

Cross-application context management must allow diverse applications, application plug-ins, software agents, and sensors to act as context providers or consumers. Therefore, a corresponding context management service needs to offer a generic and flexible mechanism for dynamic registration of such agents. Registered context consumers should be able to fetch (i.e. query) relevant data from the service at any time (pull), or be notified if relevant data managed by the service has changed (push). Furthermore, context consistency needs to be ensured, e.g. with the help of appropriate reasoning mechanisms based on a confidence and reputation system for context providers, as well as a context history. Moreover, dynamic cross-application context management necessitates special care to be taken of security and privacy issues.

3.2. Concept

We have designed our ontology-based, Cross-application Context management Service (CROCo) as a generic system allowing arbitrary context providers to submit, and context consumers to request context data via specific service interfaces. Clients may act as providers and consumers at the same time. With the help of consistency and inference rules, higher-level information that may not be provided by any sensor is derived by CROCo (inferring contextual information from basic sensor data) and serves as a general context supplier for the external context-aware applications (cf. Section 4). As an example, we can adapt web content directly the users’ age, so that inappropriate content is filtered out. However, the age of consent differs from country to country, so, as a matter of course, we need to bind the adaptation to the semantic concept of adulthood, which can be dynamically reasoned from other contextual information of a user, such as his age and location.

Figure 1 shows the overall architecture of CROCo. As already mentioned in section 2 it follows the Blackboard model, which promotes a data-centric approach in which external processes can post context information on a blackboard, or subscribe for change notifications. Thus, it is rather easy to add new context providers and consumers.

CROCo has three responsibilities: (context) data management, consistency checking and reasoning, and context data update and provision.

Data management is based on three layers: the Context History (CH), which contains the history of updates to the context model, the Consistent Context (CC), representing the currently valid, consistent contextual data, and finally the Inferred Knowledge (IK) - a layer which encapsulates all information derived, i.e. reasoned from the current context information. Consumers always request data from the Context Store, which manages CC and IK, so their division is transparent to external components. The CH is completely hidden, though, and only used internally for statistical analysis by context reasoners that include the time-variance of context data into their inferencing process. It may as well be extended to predict future context changes.

Consistency checking is done by the Consistency Manager (CM), which is triggered every time new contextual data is added to the context model. An arbitrary number of Consistency Enforcers, each responsible for ensuring consistency of a certain aspect (e.g. data type or cardinality), can register at the CM to carry out consistency checks and conflict detection.

Reasoning is carried out by a Reasoning Manager (RM). Similar to the CM, arbitrary Reasoners can register at the RM, which invokes them to start the reasoning process once relevant data changes. Context reasoning is based on facts stored in a Facts Database and its result is, of course, sub-
Figure 1. Conceptual architecture of CroCo

object to subsequent consistency checks. Since inferred data must not overwrite existing data – simply because it would interfere with the Confidence mechanism – it is stored separately in the IK base. This also facilitates the dynamic integration of additional reasoners at run time without negative impact on the Consistent Context.

Finally, to support context data update and provision, CroCo provides two services: the Update Service and Query Service. The former facilitates updates and changes to the context model, while the latter provides both a synchronous and asynchronous way to retrieve context information from CroCo. Context consumers, i.e., systems that are interested in context changes can register for specific context data, so that they are notified by a callback handler once it changes. Of course, consumers can also request data synchronously. These two different approaches facilitate the use of CroCo in different application scenarios. In any case, the Privacy Enforcers ensure that consumers have the necessary authorization to access data.

3.3. Functionality

In the following we discuss the functionality of CroCo, i.e., how its components work together at run time during context updates and requests.

Figure 2 presents the main components of CroCo and illustrates an example workflow which is discussed in more detail in the following. At first, a context provider, e.g., a sensor, sends new data in the form of regular RDF triples to the Context Update Service \( \circ \). Additional information, such as a timestamp, provider id, and confidence value, are transferred as well, which we will discuss later on.

The context information is forwarded to the Context Store \( \circ \), which internally manages the consistent and inferred context. It uses CroCoON (cf. section 3.4 or a domain-specific ontology profile derived from it to model persons (users), groups, hard- and software on client devices, time, geographic information, preferences, and much more. To ensure that the submitted context information is consistent with the internal model, it is validated by the Consistency Manager \( \circ \), which internally triggers all registered Consistency Enforcers. A consistency report is returned to the Context Store, which is added to the Context History together with the submitted context data \( \circ \). If the data is validated as consistent, the reasoning process, which is controlled by the Reasoning Manager and carried out by several registered Reasoners, is started \( \circ \). The knowledge inferred by the reasoning process is then again checked for consistency \( \circ \) and stored in the Context History \( \circ \). Finally, the Callback Handler \( \circ \) is informed that the context model has changed \( \circ \). If there are context consumers, which had previously registered as change listeners, they are notified of the context data changes \( \circ \) once authorization for access to this data has been granted by the Privacy Enforcer \( \circ \). Context consumers can alternatively request data from the Context Update Service \( \circ \), i.e., synchronously.

Several mechanisms facilitate sound and useful consistency checks and reasoning in CroCo: For one, each context provider is assigned a Confidence value (cf. [17]) representing its accuracy and reliability, which is also reflected in the Context History. Based on this value it is possible to include the quality of context updates into consistency checks. Furthermore, the Aging Knowledge Base stores information about the Variability of contextual data. As an example, the birth date of a user is static, while his location may be highly dynamic. This implies that context information can get outdated over time and may need to be updated with new, even

Figure 2. Example workflow of CroCo
low-confidence data. A last additional parameter used in CroCo is the Reputation of context providers, which depends on their data quality. If a provider constantly sends inconsistent data, its Reputation decreases, ultimately resulting in lower Confidence values for context data sent by this provider. So, consistency checks and conflict detection are influenced by the age and Variability of context data, as well as the Reputation and Confidence of its providers.

### 3.4. The CroCoON Context Model

In contrast to most prevalent, static XML-based approaches, we use an ontology-based model similar to [4] and [10]. This allows for the integration of external ontologies describing relevant contextual aspects (preferences, device characteristics, location, time, etc.) as well as domain-specific knowledge. Traditional approaches either directly store sensed contextual data, or they incorporate a static mapping from sensor data to context parameters. In contrast to this, an ontological basis provides machine-processable semantic metadata, domain-inherent integrity constraints and inferencing rules to derive higher-level contextual knowledge from basic sensor data. Thus, it becomes possible to model implicit knowledge and to establish a semantic context (meaning) on top of the purely technical context (parameters).

We have developed a generic, ontology-based context model, called CroCoON (Cross-application Context Ontology)\(^1\), for the use within CroCo. As can be seen in Figure 3, it consists of several sub-ontologies that model different aspects of context, e.g. time, place, the user and his device. Therefore, we reused concepts from several well-known ontologies, such as SOUPA [5], PROTON\(^2\) and the W3C Time Ontology\(^3\).

Based on this Upper Ontology it is possible – and intended by all means – to extend the model and integrate domain-specific knowledge to facilitate the usage of CroCo in diverse application scenarios. We call these extensions Ontology Profiles. In addition to CroCoON, they describe a certain type of activity (application scenario) in more detail, e.g. a co-browsing session, a video conference, etc. Thus, they represent the common conceptualization of context providers and consumers for a particular application scenario. An example for the co-browsing domain is given in section 4.2.

### 4. Implementation and Usage of CroCo

Based on the concept explained in Section 3, we have implemented CroCo in Java. It heavily relies on the Jena Semantic Web Framework\(^4\) for RDF and OWL processing tasks. The reasoner to date is based on the GenericRuleReasoner provided by Jena inference support. It processes Jena rules which can be stored in separate files and can thus be updated and adjusted very easily. The context history has been implemented as logger that allows for easy future extension. Also, basic privacy mechanisms, i.e. consumer-based access control is provided by the Privacy Enforcer. The service interface was built with Apache CXF. The Update Service and Query Service both support SPARQL requests [15] - the former to allow consumers to register for relevant data, the latter to query the current context model.

In the following we give two examples of application domains, i.e. scenarios that heavily depend on contextual knowledge and may benefit as well from the use of a joint context management facility. Namely, these are personal multimedia management and context-aware co-browsing. The context management service explained above was used within both domains to prove its independence and cross-application usability. Therefore, it has been deployed in the projects K-IMM (Knowledge through Intelligent Media Management) [12] and VCS (Virtual Consulting Services) [13], both using their own domain models.

### 4.1. Personal Document Management

Managing a considerable quantity of documents involves administration efforts and certain strategies for ordering and arrangement to keep track of content and structure of the collection - esp. over a long period. With the help of Semantic Web technologies, which ensure machine-processability and interchangeability, it is possible to apply semantic knowledge models and paths to organize and describe heterogeneous multimedia items. In this application area context plays a very important role and can be applied within annotation and retrieval scenarios. Retrieval tasks

---

\(^1\)The according OWL files can be found at [http://mnt.inf.tu-dresden.de/crocoon/](http://mnt.inf.tu-dresden.de/crocoon/)

\(^2\)[http://proton.semanticweb.org/](http://proton.semanticweb.org/)

\(^3\)[http://www.w3.org/TR/owl-time/](http://www.w3.org/TR/owl-time/)

\(^4\)http://jena.sourceforge.net
can be supported with the help of context-dependent adaptation and personalization techniques. Furthermore, to generate rich semantic annotation and description of multimedia content the context of its creation, modification, usage, sharing, etc. (i.e. the whole life cycle of a document) is highly valuable. Thus, a system for personal multimedia document management is primarily a context consumer and profits from cross-application context modeling. Relevant context concerning a document’s life cycle comes from desktop applications, like e-mail clients (transmission of a document via e-mail), authoring or editing tools (e.g. photo correction software), or browser information (e.g. file upload for online photo printing). The variety of context providers necessitates a flexible solution capable of cross-application context gathering and modeling.

As a practical application scenario, we utilized CROCO within the K-IMM project5 to access context information about a document’s usage. To show the applicability and benefit of context acquisition for K-IMM, we implemented a plug-in for the e-mail client Thunderbird6 as context provider for CROCO. It gathers and sends contextual information about outgoing and incoming e-mail with attachments (i.e. documents) to CROCO. The K-IMM System, acting as context consumer, registers at CROCO for notification about changes of the CROCoN based context model regarding the following SPARQL request:

```sparql
SELECT ?x WHERE {
?x rdf:type activity:MailTransfer.
?x activity:transfers ?mail.
?mail doc:hasAttachment ?document.}
```

Being notified with a MailTransfer event, K-IMM sends a query for further information about sender, receiver, attachment, time, etc., and processes the result considering whether the attachment is a document which belongs to the document collection managed by the K-IMM System. In this case, the context information about the document’s transmission is used to extend the existing semantic information about the document, stored within the K-IMM System, and to adapt the documents representation to the user (e.g. highlighting).

### 4.2. Adaptive Co-Browsing Applications

Co-browsing can be defined as an extension of the traditional web browsing. It allows for synchronous view and interaction with web pages and heavily includes means for collaboration and mutual awareness. The recent development of the Web towards a software platform has further pushed the need for such tools, e.g. for consulting and support purposes, but it also poses some serious problems. Time- and location-independent access to the Web implicates that applications are used in extremely different contexts, so that the standard “one-size-fits-all”-approach of early web applications does no longer meet today’s requirements. Even more, co-browsing can not simply follow the WYSIWIS (“What You See Is What I See”) paradigm [7], because it fails in typical, highly heterogeneous scenarios with different browsers, screen sizes and application-specific roles involved. Therefore, a co-browsing system needs to provide a common ground for all participants by adapting to single-user as well as group context parameters.

While early Adaptive Hypermedia approaches included their own context modeling algorithms, this is no longer appropriate – regarding both performance and complexity. A co-browsing system needs access to diverse contextual data it can not necessarily sensor itself. Examples include the users’ location (provided by a device-internal GPS sensor), their contacts (provided by a web-based CRM software) or topics they are interested in or currently work on (provided by their email client installation). Hence, a flexible, independent, cross-application context management system is needed.

To support co-browsing scenarios with CROCO, we developed the ontology profile CoCAB (Context-Aware Co-Browsing) – a domain model containing co-browsing-specific concepts, some of which are shown in Figure 4 and extend CROCoON. As an example, it allows us to model that a user has a specific “SystemProfile” representing his hard- and software, e.g. his browser (“WebBrowserInstallation”), and that he is participant of a co-browsing session (“CoCAB-Session”). These sessions represent the group context of all participating members and may contain information on their lowest common denominator (e.g. regarding end device capabilities) as well as higher-level information that directly results from the combination of all users’ parameters (e.g. that “all project members are participating in the session”), either by properties or relationships to such concepts.

In the project VCS7 we have implemented a co-browsing

---

5<http://mmt.inf.tu-dresden.de/K-IMM/>
6<http://www.mozilla.com/products/thunderbird>
7<http://mmt.inf.tu-dresden.de/VCS/>
system that uses CrOCo as a context modeling and storage service. Therefore, we have added server-side components to VCS, which act as context providers and consumers. A CRM client may provide context data as well, e.g., the project membership of certain uses as described above. CrOCo handles user-specific context data from the providers and infers additional context data like the “lowest common denominator” explained above. The Adaptation Engine – a DOM-based adaptation pipeline in VCS – represents a context consumer that requests contextual information from CrOCo for inclusion into the adaptation process. Thus, the views of co-browsing participants are adapted based on the contextual data modelled with CrOCo so that they have a common ground.

5. Conclusion

In this paper we have presented the design and implementation of an ontology-based, cross-application and domain-independent context management service called CrOCo. It handles arbitrary context data which is supplied by context providers and requested by context consumers via a service interface. In contrast to existing solutions, CrOCo allows for the cross-application exchange of context data and further separates the modeling and management process from context-aware applications themselves. We did validate the practicability of our approach by using it successfully within two “orthogonal” application scenarios, namely personal multimedia document management and adaptive co-browsing.

In the future we will concentrate on the development of additional ontology profiles for different domains. Moreover, we are interested in improving the context history, e.g. by realizing a more sophisticated logger or by integrating mechanisms to predict future context changes as proposed in [11]. Another challenging task is the extension of security and privacy mechanisms in CrOCo to provide a more fine-grained access control to contextual data.

References