Context Management for Adaptive Information Systems

Cinzia Cappiello, Marco Comuzzi, Enrico Mussi and Barbara Pernici

Department of Electronics and Information
Politecnico di Milano
Milan, Italy

Abstract

Modern technologies enable users accessing services using multiple channels. In the service design phase, this poses additional requirements for high software adaptivity along different technical requirements and different user expectations. During execution, services are usually dynamically selected; this service selection phase requires the identification of the most suitable service along the context that characterizes the users in the time instant in which they send the service request. This paper focuses on the selection phase and it aims at providing a framework to define and manage the context in a general environment characterized by adaptivity and multichannel access. An example to demonstrate the suitability and feasibility of the framework is provided referring to the MAIS (Multichannel Adaptive Information Systems) architecture and considering services related to the tourism domain. The MAIS architecture aims at providing automatically and efficiently services with the appropriate features by choosing among many provider offerings.

Keywords: Web services, Context ontology, Context management

1 Introduction

Modern economies focus on the customers’ satisfaction and thus the importance of the evaluation of the context and consequently of the user profile characteristics is increasing. Indeed, organizations tend to offer different types of services in order to meet and satisfy different user requirements.

1 Email: [cappiello,comuzzi,mussi,pernici]@elet.polimi.it
The personalization of contents to user expectations is a particularly relevant research theme in the Web Information Systems (WIS) design literature. Indeed, in the WIS systems literature the Profiling technique was introduced for the first time to represent user requirements and use this information for adapting the Web pages content. Profiling is the technique through which data are collected and manipulated with the goal of identifying and describing the profile of an entity, such as a user, an object, a product or a process [24]. A profile is generally a structured representation of the information that describes the users and their preferences along the services and the data they access. Users’ preferences are considered implicit when they are inferred by the analysis of personal characteristics and behavior, whereas explicit requirements are elicited directly from the interaction with users themselves.

This classification is limited to the distinction between all the information that is provided by the user and the information that can be deduced from the user characteristics or behavior. If we extend the user profile characteristics to the interactions that users have with the environment in which they are located in a certain time instant and with the device through which they are accessing services, we refer to a context.

The context definition issue is addressed also in Service Oriented Computing. Indeed, this area depicts a landscape in which different users dynamically invoke services provided by different providers using a heterogeneous set of devices and technologies. This picture is enriched by the opportunities provided by adaptive and multichannel information systems. This kind of applications, in fact, enforces the vision of customized and context-aware service provisioning: the service provided and its characteristics have to meet users’ profile, in terms of preferences expressed on the service behavior and content and to user context, intended as the features of the environment in which the user interacts with the service, such as his device in use or network access characteristics.

The need for expressing preferences on a well-defined domain has introduced the problem of defining domain concepts and relations between them using a machine-readable language, in order to let users of context-aware platforms express their preferences and platforms themselves to create a correspondent context. The Semantic Web and, in particular, ontologies have helped researchers in finding this language [3]. Nowadays, ontologies are the most recognized mean to let applications share, exchange and compose concepts and relations from different domains, and OWL, the Web Ontology Language, is the machine-readable language used to describe them [26].

This paper proposes a novel framework to define and manage context in general environments. The framework highlights a set of functionalities that are needed to manage context in information systems. The context is specified
using a flexible and extensible ontology-based model. The model is flexible, since it provides a general framework that can be applied to almost every application in which it is needed to manage user context; the model is extensible, since it provides mechanisms to add new domain specifications on which new aspects of the context can be instantiated.

The approach proposed for modeling context relies on the use of ontologies to describe context. Ontologies and OWL, in fact, can be used to connect different domain specifications in order to define a common context ontology. They can also help designers and analysts in developing applications that are able to reason about the context definition for supporting, for instance, dynamic and adaptive service selection or user profiling.

The architecture for service provisioning of the MAIS (Multichannel Adaptive Information Systems) Project [27] is proposed as a concrete application for the context model and the context management framework described in the paper.

The paper is organized as follows. Related work is shown in Section 2, Section 3 describes the structure of the context model, Section 4 discusses a framework for managing context information retrieved from the model. The MAIS Project example of context definition and management is provided in Section 5 and conclusions are finally drawn in Section 6.

2 Related Work

The need to specify context in computer-based applications has been underlined in three principal fields of research: ubiquitous and mobile computing, context-aware applications, and pervasive computing.

In ubiquitous and mobile applications, users can access data and information from a great variety of devices and applications with different capabilities and computational resources; the context, in this case, is used to deliver the appropriate content using the presentation that is best-suited for the user characteristics. Therefore, in order to define the context information, such as the user’s location, environmental conditions and device characteristics are needed. This information is, in most cases, implicit and gathered from sensors and plugins that are deployed in the environment surrounding the user. The UWA (Ubiquitous Web Applications) Project [9], for instance, proposes many different models and methodologies for developing ubiquitous web applications starting from user requirements. The customization of web applications, in this case, is obtained from user profiling based on requirements analysis rather

\[^{2}\] MAIS project web site: http://www.mais-project.it
than from the management of an actual user context. Moreover, [6] proposes a reflective architecture for mobile devices and an approach to deal with conflicts between context features and user requirements. In [7], business process executions’ exceptions can be generated by a context manager. The context model, in this case, is very simple and limited to static user personal data and a small set of mobile features, like GSM cell notification, GPS position or available wireless networks.

Context-aware applications use context to provide relevant information and services to the user [18]. The problem, in this case, is not only related to adapt content to the user computational capabilities but other elements of the context, such as user behavior or preferences, have to be considered in order to choose the most relevant information and service for the user. Typical examples of context-aware applications are virtual traveller’s guides [5] or services for disabled users [10]. In these environments, in fact, the most important factor in defining the context is related to the user preferences in terms, for example, of artistic interests or disabilities. Other aspects typical of ubiquitous computing, such as user location or device in use become second order concerns that appear only at the moment of providing the content using a customized presentation layer.

The founding idea of pervasive computing is the human-centered computation in which computational resources and services are freely available everywhere and cooperate seamlessly to support users’ tasks [15]. In order to fulfill this vision, context is not only used to identify the environment and user characteristics for executing tasks, but it is also exploited to infer new knowledge about users to help them in a transparent and proactive manner. Pervasive computing underlines the need to provide formal context descriptions and rules by which new context features can be extracted.

Academic research on user profile and context in the field of service oriented computing is still developing and, although people agree on the concepts that characterize the notion of context, it is difficult to give a formal definition of it. This paper exploits the definition in [12], by which context can be considered as any information that can be used to characterize persons, places or objects that are considered relevant to the interaction between a user and an application, including users and applications themselves.

The definition and consequent usage of context for ubiquitous, mobile or context-aware applications and pervasive computing is strongly limited and applied to specific domains. Many applications related to the tourism domain, i.e., virtual traveler’s guides, treat the context as a set of information on users’ location and preferences [5], group belonging, or users’ behavior [18], to give tourists information and services closely related to their interests.
In other applications, context is often intended as a collection of specific user characteristics, such as disabilities in disabled user domains [10], or device and channel features in mobile applications.

Besides an extensible and cross-domain context definition, the literature lacks the presence of frameworks and architectures to manage context definition and expansion. The Context Toolkit framework contains guidelines to help developers in building context-aware applications quickly and from scratch without limiting the complexity of the context space [13]. Although this proposal represents a great effort in trying to formalize context structure and context-aware computing aspects, it still views the context as a collection of data coming from sensors surrounding users, without considering users’ preferences or other context features. [16] discusses a framework for pervasive context-aware applications using the Context Modeling Language (CML). The framework is oriented toward the use of logics to reason on the context instead of the description of mechanisms to extend and manage context information. A context model for service selection is described in [19], the model is applied to a middleware for gathering context information for a guidance board PDA application.

Many tools and languages for describing context have been developed. The Context Modeling Language represents an ad-hoc language to specify types of information, their classification, dependencies and quality metadata [16]. Furthermore, Contextual Graphs [23] are the elements of a graphical language that can be used to specify context and rules that have to be exploited to take decisions related to the value of different context features.

The use of ontologies as a way to specify context, perhaps, has emerged as a proper approach to deal with context and rules derived from context specification [15]. Ontologies, in fact, enable the definition of concepts and relations between context elements and provide flexibility and extensibility because they can be combined in order to include descriptions of different domains such as quality and security metadata, in the context specification.

The Italian MAIS Project has developed some initiatives to deal with the problem of context specification and context-aware applications at different architectural levels. In [4], a methodology to support very small databases usage in context-aware applications is proposed. A preliminary model for Web service adaptive selection and provisioning is proposed in [2]. Other groups inside the project are developing methodologies for modeling users’ context and profile [1]. These efforts are oriented toward the problem of adaptive interfaces generation for the presentation layer of mobile and portable devices and to user profiling. The objective of this paper is to provide a general and extensible users context definition that can be used at different architectural
levels, showing its application in the adaptive service selection scenario and discussing its impact on the MAIS architecture.

3 Context Top Ontology

Generally, a user profile describes user requirements with a list of \(<\text{attribute-value}>\) pairs, where the value describes a specific user or user class. Each requirement can be classified as:

- **Explicit**: the user specifies the value.
- **Implicit**: the value is not specified explicitly by the user.

The implicit requirements are usually inferred by the analysis of the user characteristics and the identification of a user’s class to which the user belongs.

In the literature, as discussed in Section 2, there are not contributions that describe the elements that can be contained in a general context. Indeed, all the contributions describe contexts within a specific domain. The goal of this paper is to provide the elements and guidelines to define and create a user profile in a generic domain. In order to describe the general context, we use an approach based on ontologies. This is because ontologies allow a flexible and extensible definition of the concepts characterizing a specific element, that in this case is a general context. In order to formalize the proposed ontologies, we have used the OWL language [26]. In Figure 1 the top ontology is presented. The top ontology specifies the general concepts defined independently of a domain of application and that can be used in different application domains [25]. The context here defined is composed of three fundamental elements: the user profile, the environment and the channel.

The **user profile** describes the properties associated with the user and it is distinguished in domain-dependent user profile and domain-independent user profile. The former contains all the characteristics that describe the users along their social distinctive properties. It includes personal data, their physical abilities and general interests. They are usually expressed as \(<\text{attribute-value}>\) pairs. The latter refers to the specific domain in which the context is considered and it usually contains the preferences along specific services. They are usually expressed as both \(<\text{attribute-operator-value}>\); the specification of operators, such as \(=\), \(>\), or \(<\) allows the definition of complex constraints along the service characteristics. An example of constraint in the tourism domain is “hotel category \(>\) 3”; this indicates that in the booking service the user wants to select only hotels belonging to the 4th and 5th categories. Figure 2 shows an example of the modelling of a part of the context, namely the User profile, using OWL.
The environment collects the information about the geographical position, the ambient condition, the temporal details and the actions that characterize the interaction of the users with the surrounding space.

Finally, the channel describes the elements that characterize the interaction of the users with the platform used to access services. The considered model characterizes the channel through device features, network, network interface and application protocol characteristics.
For each subclass that is included in the user profile, environment, and channel classes, there are ontologies that specify the admissible terms and concepts. Figure 3 illustrates an example of ontology for location specification.

As described above, a large variety of data is stored and used to provide a significant context. These data have to be associated with metadata, both for distinguish them in their heterogeneity and for the need to formalize their usage and management policies. In the reminder, this procedure is referred to as labeling.

Metadata can be classified according to some basic criteria. In the literature, metadata are usually distinguished in three categories: build-time metadata, control metadata and usage metadata [11]. Along these categories, we define the metadata that will be associated with the context data. The build-time metadata are created and used during the system design and construction phase. Along this definition, it is necessary to define metadata that distinguish between persistent data and time-related data. The former highlight data that are not modified frequently, and thus they have to be permanently stored, while the latter identify data that are frequently updated and thus have to be inferred in a real-time way. The time-related data are generally provided by device sensors able to detect, for example, the location or the ambient conditions that characterize the context.
Control metadata are metadata that are used in the execution phase. In this category we define the security utilization metadata that provide notes about the security and access rights. Indeed, in the context data can be associated with different levels of security. For example, personal data are in general managed with mechanisms that assure high security levels, while other information, such as the domain preferences, can be managed without any control if not clearly specified and requested by the user.

Usage metadata are defined to support users in using and understanding of data. In the reference scenario, it is fundamental to introduce metadata that provide technical details about data storage and maintenance. Specifically, in order to manage both centralized and distributed approaches, we will use metadata that specify the effective data storage support and its location.

All the defined metadata are used inside an architecture in order to define the operations and procedures to perform. The correspondences between metadata values and the actions to undertake are defined using management rules that have to be stored in a specific repository.

4 Context Management Framework

The objective of the framework described below is not to provide a software system that can be used by context-aware information systems for managing their context, but it is an attempt to highlight the essential functionalities needed by a generic context-aware information system that aim to manage its context using the approach described in Section 3. Once a system designer decides to use the context top ontology, the Context Management Framework modules can be mapped into the information system architecture and implemented from scratch. With this approach, designers can extend the implementation of their information systems including a set of modules that can facilitate the management of the context, without having to integrate a third party software. Its realization and architectural impact will be discussed in the following section while, in Section 5, will be described an example of how this framework can be used within a multichannel adaptive information system, describing the context management in the MAIS Project.

Figure 4 shows the framework; it comprises three main components:

- The Repository, which contains the context top ontology, the metadata definitions and the rules for the context management;

- The Global Context Manager, which provides the functionalities exploited by information systems for reasoning on the context and managing the communication with the agents that they coordinate;
• A set of one or more **Agent Context Managers**, which provide agent-side functionalities to manage the context and communicate with the information system.

### 4.1 Repository

This component comprises the **Ontologies Repository** and the **Metadata and Rules Repository**, which are the repositories employed for storing all the information needed by the framework. The first repository contains the context top ontology and the domain specific ontologies, while the second stores the metadata and the respective rules. The context top ontology, the metadata, and the rules are defined during the design phase and are stored into the registries by the system designer. Domain specific ontologies can also be added and stored at runtime.

### 4.2 Global context manager

The **Global context manager** is the core of the framework and has the task of managing and reasoning on the context and sharing context information with
the agents. These functionalities are provided by three modules:

- **The Context Broker** manages the context during the runtime phase. Given a context top ontology, metadata, a set of rules, and an initial labeling, the Context Broker creates an initial instance of the context retrieving the context information from every agent. Once this instance is created, it is saved into the *Runtime Context* repository, which is used for storing, coherently with the labeling, every instance of an agent’s context and it is modified every time that an agent changes its context. The initial instance of the context is also stored into the *Historical Context* repository, which contains the information employed for tracing the evolution of the context over time. At runtime, the Context Broker interacts with the agents for gathering information about the context and keeping the registries up to date. This interaction is performed following the guidelines that are obtained combining the set of rules with the results that can be acquired using both the runtime and the historical context reasoner.

- **The Runtime Context Reasoner** performs reasoning over the Runtime Context. This module can be used for analyzing the context during the runtime phase and allowing the context broker to perform actions when a particular configuration of the context is encountered. The runtime reasoner can even be useful for performing inference on the context.

- **The Historical Context Reasoner** performs reasoning over the Historical Context. For instance, the historical reasoner can be used for analyzing the variations of the context over the time and retrieving information about the behavior of agents [21,17].

### 4.3 Agent context manager

The term *agent* identifies a person or a software component that interacts with the context-aware information system. The *agent context manager* represents the component used for managing the context on the agent-side. The agent context manager bases its behavior on a combination of context top ontology, domains specific ontologies, metadata, and rules. For example, the storing policies employed by an agent context manager are based on the *usage metadata*. Anyway, the agent context manager does not have a view on the overall context ontology but only on a subset of it. This subset is provided to the agent by the context broker that decides which part of the context ontology share with an agent.
5 Service Selection Based on Context

This section describes an example of how the framework presented in Section 4 can be integrated into a multichannel adaptive information system. The example consists in adding functionalities for managing the context to the architecture developed for the MAIS Project and observing how this addiction affects a particular MAIS functionality, the concrete service selection.

The MAIS project studies a flexible highly adaptive environment for delivering services through different distribution channels in a context-aware manner. From a user perspective, such an architecture allows the invocation of services registered and published in a public UDDI Registry owned by the MAIS architecture. A user interacts with the services through several devices (e.g., Smartphone, PDA) and a variety of wired and wireless networks. In MAIS, the device represents the user end point of the channel, i.e., the element that allows the communication with the service provider [20].

The context-aware behavior is enabled by the particular service categorization adopted in the MAIS architecture. MAIS gives prominence to the distinction between abstract services, that is, non invocable services, and concrete services, that is, invocable services. In brief, an abstract service represents an interface that is implemented by one or more concrete services. Users select abstract services and invoke them. The decision of which concrete service has to be selected is performed by the MAIS architecture (i.e., the concrete service selection) that chooses the concrete service that best fits the constraints imposed by the context in which the invocation is carried out. The selection is performed matching services description with context information [22].

The context management framework can be easily integrated into the MAIS architecture [22] and Figure 5 shows how the modules of the framework are deployed into the MAIS architecture. The MAIS architecture provides an infrastructure for managing adaptive orchestrated Web services. The user and the environment execution context can be managed with the support of the MAIS reflective architecture. The reflection mechanisms support storing and updating, in a distributed environment, information about interacting parties, based on guess and updates mechanisms [14]. The Agent Context Manager is associated with the Platform Invoker module, which allows user interaction with the service orchestration environment. The other modules proposed in the context management framework are inserted in the service provider environment, to support dynamic service selection in the Concrete Service Invoker (Context Broker) and service recommendation in the recommendation environments: the historical reasoner for the recommender system and runtime reasoner to support service negotiation.
Figure 6 illustrates the example scenario, where two users (i.e., agents) interact with the MAIS architecture. The first user, located in Italy, is equipped with a PDA and a smartcard used for storing data, while the second, located in Germany, interacts with the architecture using a standard web browser. Both users want to reserve a seat on a train and choose the same abstract service, the TrainReservation service.

This service is described with a WSDL [8] document, which represents the functional interface, and a WSOL [28] document, which describes the quality of service. For this abstract service, two concrete services are available. The first, called TrainSocietyIT, offers reservations in Italy and requires the CircuitA credit card circuit while the second, called TrainSocietyDE, requires the CircuitB circuit and accepts reservations only in Germany.

The architecture is equipped with our context top ontology that contains the ontology for location specification illustrated in Figure 3, a domain specific ontology (i.e., tourism ontology) that describes concepts related to the train reservation, and a labeling that marks credit card circuit as sensitive data. There is also a rule stating that every sensitive data cannot be stored by the architecture but have to be requested to users every time it is needed. This rule constrains the communication between the architecture and users, but does not make commitments on methods used by agents for storing their sensitive data. In our example the first user stores its sensitive data on the
smartcard, while the second one reinserts data every time it is needed. The management of this information is performed by the respective Agent Context Managers.

When a user invokes the selected abstract service, the architecture picks out the best concrete service according to the user’s location and credit card circuit. The location can be automatically extracted from the context while the information about the credit card have to be requested explicitly to users because it is a sensitive data. In our example the concrete service \textit{TrainSocietyA} is assigned to the Italian user while the concrete service \textit{TrainSocietyB} is assigned to the German one.

![Fig. 6. Scenario](image)

6 Conclusions and Future Work

The formalization of user context and profile is a fundamental concern in ubiquitous, mobile and pervasive computing. Due to the heterogeneity of user context characteristics, that can span from domain dependent preferences, i.e., hotel class preferences, to metadata expressed on context features, the literature contains examples of frameworks and context-aware applications
only for specific domains, such as virtual traveler’s guide or disabled users domain.

The paper has proposed a flexible and extensible ontology-based model for defining and managing user contexts. Indeed, ontologies allow the definition of concepts and relations of different domains, and permit the inclusion of metadata labeling definition for context content management.

The general framework described for managing context is instantiated in the architecture of the MAIS Project for adaptive service provisioning and it has been shown how it could exploited for adaptive service selection.

Future work will deal with the formalization of rules needed to reason on the context and metrics to evaluate the benefits obtained by the use of a formal context definition and a context management framework in context-aware applications. It will be also studied how this context specification can be used in the phase of adaptive and multichannel service design, in particular how it can be exploited for requirements driven user profiling in order to let service designers create service offerings that correspond directly to users’ classes extracted from the context definition.

Acknowledgement

Part of this work has been supported by the Italian Ministry of University and Research, MIUR, under FIRB grant MAIS.

References


