A Dynamic Context Management Infrastructure for Supporting User-driven Web Integration in the Personal Web

Norha M. Villegas¹,², Hausi A. Müller¹, Juan C. Muñoz², Alex Lau³, Joanna Ng³, and Chris Brealey³

¹University of Victoria, Victoria, British Columbia, Canada  
²Icesi University, Cali, Valle del Cauca, Colombia  
³IBM Canada Ltd., Toronto, Ontario, Canada

Abstract

Most web applications deliver personalized features by making decisions on behalf of the user. Thus, the user’s web experience is still a fractionated process due to a lack of user-centric web integration. In contrast, smarter web applications will empower the user to control the integration of web resources according to personal concerns. Moreover, as the user’s situation and web resources continuously evolve, web infrastructures supporting smarter applications require dynamic and efficient mechanisms to represent, gather, provide, and reason about context information. Aiming at optimizing the user’s web experience, this paper proposes a self-adaptive context management infrastructure, and an extensible context taxonomy based on the resource description framework (RDF). Our context manager is able to deploy new context management components to keep track of changes in the user’s situation at run-time. Our taxonomy includes a set of inference rules for supporting dynamic context representation and reasoning. Using a smarter commerce case study, we illustrate the application of feedback loops and semantic web, to the realization of dynamic context management in the personal web.

1 Introduction

Adaptive software systems adapt their behavior at run-time to address functional and non-functional requirements under changing environmental conditions [12]. The smart Internet, also known as the Internet of things, is a good representative of adaptive software systems. In the smart Internet, online services are discovered, aggregated and delivered dynamically, interactively, fully or semi-automatically in response to evolving users concerns, and under heterogeneous system infrastructures and dynamic execution conditions [5].

The personal web, an application of the smart Internet, involves complex and highly dynamic functional and non-functional requirements. Its objective is to empower the user to control the integration of information, web services, and web resources into her personal web context sphere (the user’s information repository). This integration is performed according to the user’s matters of concern, without requiring technical skills, and with the goal of personalizing and exploiting her web experience [13]. Different approaches have been proposed to enable general Internet users to pull resources from the web to an integrated and personalized platform. An instance of such approaches is the Mashup model. Mashups enable users to create new functionalities based on existing web resources [9, 11]. Nevertheless, the application of this model is still limited by the usage of tools that require special programming skills that most Internet users do
not have. In contrast, the personal web proposes an approach where web resources, described using resource description framework (RDF)\(^1\) based linked data [1], are integrated into the user’s context sphere through simple web interactions such as “tagging” or the popular Facebook “Like” button.\(^2\)

Considering that the user’s interest and web entities are highly heterogeneous and dynamic, the personal web requires context-aware adaptive instrumentation to support the user in the integration of web resources under changing environmental conditions. Moreover, this instrumentation must ensure adaptation properties and non-functional requirements to guarantee its operation while improving the user’s experience [19]. For instance, the *settling time* adaptation property, which in this case may correspond to the time required for the adaptive instrumentation to deploy a new context sensor, affects important performance quality factors such as latency, throughput, and capacity. These measures are critical to guarantee desirable levels of response time in user-centric web environments.

Under this complex setting, we identified two main research challenges for the realization of user-driven web integration. The first challenge concerns the representation of relevant information within the user’s sphere of context. Context models within this sphere represent the user’s interests and her current situation. This information must be processed to understand the way it affects the interactions between the user and web entities [17]. Moreover, these models must be flexible enough to support context representation and reasoning in different problem domains, and without demanding manual changes in the instrumentation from one domain to another. The second challenge is the management of the context information life cycle (i.e., context acquisition, processing, reasoning, provision, and disposal). In order to manage the user’s context sphere life cycle in the personal web, context management infrastructures must adapt themselves to address changing requirements. For instance, the discovery of new context entities at run-time would require the deployment of new sensors to acquire context information. To address these challenges we investigate: (i) the application of context-aware computing and semantic web technologies for representing and reasoning about context information as the first level entity the user interacts with [17]; and (ii) the application of feedback loops [8, 12] to the realization of dynamic context management infrastructures as required in the personal web [18].

This paper proposes an extensible RDF-based taxonomy of context information. It also discusses the design and implementation of a dynamic context management infrastructure (CMI). Our approach aims to support the vision of the personal web, by assisting the user in the management of her context information life cycle. While the user is surfing the web, our CMI gathers from *personal web-enabled* (PWE), web sites acting as context providers and consumers, the context information that characterizes web entities the user interacts with. Then, it integrates this information into the user’s context sphere. Therefore, our approach is to treat web resources as *context entities* that provide context information related to the user’s preferences along her web experience. The integration of new context entities enriches the user’s personal web sphere with new context information that is then provisioned by the context manager to PWE web sites. These PWE web sites exploit the provisioned context to deliver web resources and services to the user according to her interests and current situations. Our proposed CMI self-adapts by deploying new RDF context sensors and providers, thus enabling context exchange with PWE web sites integrated at run-time. The proposed infrastructure is built on top of IBM WebSphere technologies, where feedback loops control the adaptive behavior of the service component architecture (SCA) implementation.

The remainder of this paper is organized as follows. Section 2 describes the smarter commerce case study used to guide the validation of our approach. Section 3 presents the role of context management in realizing the vision of the personal web, describes our proposed taxonomy for context representation, and explains

---

\(^1\) RDF Primer—http://www.w3.org/TR/2004/REC-rdf-primer-20040210/

\(^2\) Open Graph Protocol (OGP)—http://ogp.me
the application of semantic web technologies to the management of the context life cycle. Section 4 details the software architecture of our proposed CMI. Section 5 summarizes our work and reviews related research. Finally, Section 6 presents important conclusions and future work.

2 Case Study: a personal shopping web sphere

In order to illustrate how user-driven web integration can be realized by empowering the user to manage her personal context sphere, we use the following smarter commerce scenario [13]. Imagine a woman who is a frequent online shopping user. For surfing her personal web, the first activity she must perform is the initial configuration of her personal shopping web sphere. For this, the front-end provided by the personal web infrastructure enables the user to complete her personal web profile (e.g., city, country, and preferred payment methods). The user may decide also to register her favorite PWE web sites, and other PWE entities such as her personal calendar, email account, and social networks. PWE entities can be deployed either locally or remotely. By registering a PWE web site, the user agrees to provide it with context information about her preferences and situation, and to acquire context information from her interactions with web resources exposed by this web site. The user also configures the desired privacy settings (e.g., always ask for providing or acquiring context information to or from the PWE web site). As soon as the user registers a new PWE web site, our CMI acquires the web site identification and its context requirements specification (e.g., a web site offering technology products may be interested in product categories such as the user’s favorite mobile device brand).

2.1 Interacting with PWE web sites

Whenever the woman uses a PWE browser to access a specific web site, our CMI receives this request to identify whether this web site corresponds to a registered PWE web site, and whether it is already registered into the user’s personal context sphere. If the web site corresponds to a registered PWE web site, the CMI, taking into account the user’s privacy configuration, provides the site with relevant context information about the user’s situation and preferences (e.g., the user’s shopping preferences according to the categories of products offered by the PWE web site). Then, the PWE web site exploits the provisioned context to offer products and services accordingly, thus enhancing the user’s web experience. As the user interacts with web entities exposed by the site (e.g., by ranking products, creating wish lists, or generating purchase orders), the instrumentation of the CMI processes these interactions to identify context information. Instances of this identified context are: category of products the user interacted with, her preferred payment methods, and favorite store locations. Once the user finishes her interactions with the PWE web site, the CMI acquires and integrates this context information into the user’s context sphere. The CMI also supports the user in registering a new PWE web site when required. Context acquisition and provision is not feasible with PWE sites not registered as part the user’s context sphere.

2.2 PWE web sites requirements

PWE web sites are web sites the user interacts with in the personal web. To be enabled for the personal web, they must support four fundamental requirements: (i) to expose the services required to interact with the CMI (for context gathering and provision); (ii) to provide user-centric interactions according to the problem domain (e.g., rankings on web entities); (iii) to apply linked-data (RDF) to describe itself, its web entities, and corresponding user interactions; and (iv) to provide the user with personalized products and services according to the user’s context. For the implementation of PWE web sites in this case study, we used the open source version of Magento, a feature-rich platform for building e-commerce applications. We selected this platform because of its support for the integration of semantic web technologies, and its scalability and performance.

3Magento—http://www.magentocommerce.com/
3 Context Life Cycle Management

We claim that user-driven web integration can be realized by assisting the user in the management of relevant context information within her personal context sphere. Thus, we treat web entities as context sources. This context information is integrated into or eliminated from the user’s context sphere according to the user’s web interactions. For this, we need to provide the user with suitable instrumentation for the representation of context information and the management of its life cycle [17]. In this paper the management of the context life cycle includes: (i) the secure acquisition of context information from PWE web sites according to user interactions (e.g., a product added to an on-line wish list will become a context entity within the user’s context sphere); (ii) the integration of context information into the user’s context sphere (e.g., the integration of the product’s category in the user’s context sphere); (iii) context reasoning to infer new context facts from the information in the user’s context sphere; (iv) the secure provision of context information to PWE web sites; and (v) the disposal of context information whenever it is no longer relevant for the user.

3.1 An extensible RDF-based taxonomy for context representation

Context representation is crucial for managing the context information life cycle. Context models represent the relevant aspects of entities that affect the interactions between users and systems, as well as the situations that trigger dynamic changes in such interactions. Linked data uses RDF to make typed statements (predicates) that link arbitrary things in the world (objects and subjects) [1, 2]. In the same way our approach exploits RDF as a machine-readable specification for context information. We apply RDF to represent web entities and the way the user interacts with them. URIs identify context entities (web resources); HTTP provides the vehicle to access these entities; and RDF/XML, Jena [4], and SPARQL [16] support context acquisition, provision, and reasoning. Previously we proposed a context classification useful to characterize context information for any problem domain [17]. Based on this classification, this paper proposes SmartContext, an RDF-based representation useful to characterize the situation of web entities in the smart Internet domain (cf. Fig. 1). SmartContext provides a set of semantic RDF and OWL-lite rules, and an extensible taxonomy for context representation and reasoning. This way, our approach supports knowledge sharing via context gathering and provision between the user’s context sphere and PWE entities; context reasoning based on existing logic mechanisms; and knowledge reuse via the integration of domain specific ontologies [20]. Figure 1 presents the core elements of SmartContext. These elements are defined in three different layers depending on whether they define concepts for the smart Internet, the personal web, or a specific problem domain in the personal web.

**The generic context taxonomy (gc).** The top layer in Fig. 1 defines the foundational elements to represent context information in any problem domain of the smart Internet. ContextEntity is the main type required for context representation. Any other context type is derived from it. Individual context concerns anything that can be observed about an isolated web entity (e.g., a PWE web site). Two of its possible classifications are Human and Artificial context. Human context characterizes the user’s preferences and behavior (e.g., the woman’s personal profile). Artificial context describes virtual (e.g., a user’s personal calendar) and physical web entities (e.g., the user’s preferred payment method). Location represents the place of settlement of an object and can be classified as physical (e.g., a hotel where a conference takes place) or virtual (e.g., the URI of a web service that supports the user’s preferred payment method). Activity context answers questions regarding future, current, and past goals, as well as actions and tasks of humans or web entities. In the personal web, activity context is important to rep-
represent the interactions between the user and web entities (e.g., annotating a product using tags). Finally, Time context not only provides information about a specific date and time, but also categorical information such as holidays, working days, and meeting schedules. Definite time context represents time frames with specific start and end points (e.g., the interval of time scheduled for a conference trip in the user’s calendar).

The personal web context ontology (pwc). The middle layer of SmartContext defines the personal web base vocabulary. The User type, derived from the Human category, represents information about the user. Web-Resource, a sub-classification of Artificial context, represents web elements the user interacts with such as PWE web sites, web services, and products offered on-line. GeoLocation and EndPoint are sub-categories of Physical and Virtual location context respectively. ScheduledEvent, PerWebConcerns, and User-Interaction represent subcategories of ActivityContext. ScheduledEvent may correspond to any calendar event registered in the user’s context sphere (e.g., a musical concert). PerWebConcern characterizes matters of concern (e.g., the shopping concern described in this case.
study). This category is useful for the classification of PWE web sites. Finally, the User-Interaction represents interactions between the user and web entities. Currently, the infrastructure supports three interaction types: tagging, liking, and ranking. These interactions enable the CMI to infer context facts from integrable web entities (e.g., product categories the user likes will be integrated into the user’s context sphere).

**Domain specific ontologies.** The bottom layer of SmartContext, shopping context ontology (sc), defines the context types in a particular problem domain of the personal web. For the shopping scenario introduced earlier, this layer corresponds to eClassOWL, a web ontology for describing products and services. This ontology is also used by GoodRelations, the standardized vocabulary for describing web offerings in e-commerce domains. The categories in the sc layer represent the information that may be integrated into the user’s context sphere. For our shopping concern, potential relevant web entities are instances of PaymentMethod, DeliveryMethod, Currency, and ProductServiceCategory. The interactions correspond to WishList and ShoppingCart, as well as the ones defined in SmartContext’s middle layer.

### 3.2 Realizing user-driven web integration

**Context reasoning with semantic web technologies.** Context reasoning is crucial to characterize the user’s situation and provide PWE web sites with context information accordingly. Thus, a CMI for the smart Internet must be able to infer relevant context facts from context information gathered from web entities. In our implementation, context reasoning is based on the vocabulary defined by SmartContext (cf. Fig. 1), RDF(S), OWL-lite, and user-defined rules at different levels of our taxonomy. The context inference engine is provided by Jena [4]. Jena is a Java framework and programming environment for RDF, RDF(S), OWL, and SPARQL. The Jena’s inference API enables applications to process RDF graphs enriched by assertions entailed from relevant ontologies.

Table 1 presents selected examples of the semantic rules defined for context reasoning in our CMI. Rules are defined as a set of premises, a list of conclusions, and an optional name and optional direction. Each term corresponds to either a triple pattern, an extended triple pattern, or a call to a built-in function. Row 1 in Table 1 corresponds to an RDF(S) rule, Rows 2 and 3 are OWL-lite rules, and Row 4 presents an instance of our user-defined rules. User-defined rules can be either defined at design-time or loaded at run-time.

As an illustration, suppose the RDF graph presented in Fig. 2 represents the initial model of our user’s context sphere partially (without the discontinuous edges and the gray nodes). According to this model, the woman is interested in products of type Music, in particular in the album Shakira Hits classified into the Pop-Music category. In the same way, she scheduled a calendar event, CASCON2011, for a specific date interval. The event will be located in HiltonMarkham, and has been tagged as a Shopping Concern. From this model, which is compliant with SmartContext, the context engine infers new context facts by applying rules such as the ones presented in Table 1. New context facts in the figure are represented by the discontinuous edges—new predicates, and gray nodes—new context entities. Context reasoning works as follows. First, the application of the owl:transitiveProperty to the gc:locatedIn predicate enables the engine to infer that: HiltonMarkham is located in Markham; Markham is located in Toronto; and thus the conference is also located in Markham. Then, by applying pwc:isNearTo, the context manager infers that CASCON2011 is near Toronto, as Markham is located within a distance set by the user to define nearby places (e.g., 40 mm). Finally, through owl:inverseOf, the engine infers that as the property locatedIn is the inverse of the property hosts, and CASCON2011 is located in HiltonMarkham, then

---

4http://www.heppnetz.de/projects/eclasiowl/
5http://www.heppnetz.de/projects/goodrelations/
6RDF Semantics— http://www.w3.org/TR/rdf-mt/
7http://www.w3.org/TR/owl-ref/
8Jena—http://jena.sourceforge.net/inference/
Table 1: Examples of context reasoning rules

<table>
<thead>
<tr>
<th>Axioms</th>
<th>Jena Rule Definitions and Examples</th>
</tr>
</thead>
<tbody>
<tr>
<td>owl:transitiveProperty</td>
<td>(?P rdf:type owl:TransitiveProperty), (?A ?P ?B), (?B ?P ?C) → (?A ?P ?C) (locatedIn rdf:type owl:TransitiveProperty), (CASCON2011 locatedIn HiltonMarkham), (HiltonMarkham locatedIn Markham) → (CASCON2011 locatedIn Markham)</td>
</tr>
<tr>
<td>pwc:isNearTo</td>
<td>(?A rdfs: subclassOf gc:GeoLocation), (?B rdfs: subclassOf gc:GeoLocation), lessThan(distance(?A, ?B), ?D) → (?A pwc:isNearTo ?B) (Markham rdfs: subclassOf gc:GeoLocation), (Toronto rdfs: subclassOf gc:GeoLocation), lessThan(distance(Markham, Toronto), 40) → (Markham pwc:isNearTo Toronto)</td>
</tr>
</tbody>
</table>

HiltonMarkham hosts CASCON2011.

Figure 2: Initial RDF model of the user’s context sphere (partial view)

**Context gathering, provision and integration.** Context gathering enables our CMI to acquire context information about relevant web resources from PWE web sites. For this, PWE web sites, based on SmartContext and its domain specific ontologies (e.g., eClassOWL), generate an instance of an RDF subgraph with information regarding the web entities the user interacted with. The CMI gathers this graph as an RDF/XML serialization using SPARQL endpoints [16]. In order to realize context provision, RDF endpoints provide PWE web sites with RDF graphs. PWE web sites process the provisioned context to offer products and content to the user accordingly. Before completing either gathering or provision, the context manager checks the compliance of RDF graphs with SmartContext and the corresponding domain specific ontologies.

Continuing with our illustration, suppose now the user interacts with a shopping PWE web site that offers on-line tickets for music concerts. The CMI provides this web site with the user’s context presented in Fig. 2. Once the user finishes her interactions with the web site, the CMI gathers from the PWE web site, and integrates into the user’s context sphere the sub-graph presented in Fig. 3. Gathered context is represented by white nodes. In this case the user interaction was a purchase of an e-ticket for a concert. From gathered context, the context engine infers new context information (gray nodes and their corresponding edges): the user not only bought a new product but now has a new calendar event for a concert that will take place in Toronto on November 10. The integration of new con-
text entities into the user’s personal context sphere is realized through programmable operations on RDF-based context models. For instance, we use the Union operation supported by Jena’s API for merging data from different RDF graphs. A merge between the graphs presented in Figs. 2 and 3 will generate the new user’s context sphere after the purchase.

Figure 3: An RDF sub-graph instance with gathered and inferred context information

4 The Context Management Infrastructure

This section describes the software architecture of our dynamic context management infrastructure (CMI) (cf. Figs. 4-7). The architecture and its realization are based on the assembly model specification for SCA version 1.0 [14]. SCA defines a programming model for building software systems based on service oriented architecture (SOA) design principles. It provides a specification for both the composition and creation of service components. SCA is independent from implementation technologies and communication mechanisms. For components, it includes different programming languages and environments. For communication mechanisms, SCA supports various communication and service access technologies such as web services, messaging systems, and remote procedure calls (RPC). SCA implementations support the realization and execution of SCA-based architectures such as the one presented in this paper. Examples of these implementations are Apache Tuscany, Fabric, IBM WebSphere, and FraSCAti. Implementations details described in this paper correspond to WebSphere technologies.

Figure 4 presents the legend for the main SCA artifacts used in Figs. 5-7. Components are the basic artifacts that implement the program code for managing the context life cycle. They also implement the feedback loop components that control the deployment of new RDF sensors and new RDF context providers at run-time. Services are the interfaces that expose these functions to be consumed by other components. References enable components to consume services. Composites provide a logical grouping to deploy components. Promoted interfaces are endpoints that must be resolved by services or references outside the composite. Composites are deployed within an SCA domain that generally represents a same processing node. Although the deployable annotation is not defined by the SCA specification, we use it to indicate that the corresponding component is deployed at run-time.

Figure 4: Artifacts of the SCA assembly model specification [14]

As presented in Fig. 5, the architecture of our CMI defines three types of composites deployed in three different SCA domains. The top part of the figure corresponds to the Client composite deployed in the User’s client domain. This composite represents the components in the workstation the user is navigating from (e.g., a laptop, a desktop or a mobile device). The bottom left part corresponds to the PWEnabledSite composite deployed in the PersonalWebEnabledSite domain. This composite represents the required infrastructure for PWE web sites to interact with the CMI. The bottom right part corresponds to the ContextManagementInfrastructure composite deployed in the PersonalWebInfrastructure domain. This composite constitutes the core components of the CMI. In the following subsections we use the terms CMI and context manager to refer to the
4.1 The user’s client domain

The client domain hosts the client composite and its three main components (cf. top part of Fig. 5). The first component is the ContextManagerExecutor. This component, deployable at run-time, controls the deployment and operation of RDF sensors. The second component is the ClientRDFSensor. This component, also deployable at run-time, gathers context information represented as linked data (e.g., the CASCON conference event tagged as a shopping concern) from PWE entities available in the user’s client (e.g., a personal calendar or agenda). The third component, the PersonalWebEnabledBrowser, corresponds to a web browser instrumented with partial functions of the CMI. These functions are deployed in the form of a context management plug-in. The plug-in is responsible for identifying PWE web sites user requests. The browser enables user interactions with the interface of her personal web sphere (e.g., to modify her user profile, preferred PWE web sites, and privacy settings).

4.2 The PWE web site domain

PWE sites are deployed in PersonalWebEnabledSite domains. As explained in Sect. 2.2, these PWE web sites must expose instrumentation for supporting context acquisition and context provision. This instrumentation corresponds to the PWEEnabledSite composite and its components, PWRDFContextBroker and PersonalWebInstrumentation (cf. bottom left part of Fig. 5). These components are not necessarily required to be SCA-based artifacts as long as they expose communication interfaces compatible with the CMI. The PWRDFContextBroker component exposes the userContextGathering service required by the CMI to push context information from the user’s context sphere to the PWE web site (context provision). It also exposes the RDFSensor reference used to push context information related to relevant web entities from the PWE web site to the CMI (context gathering). The PersonalWebInstrumentation component exposes three main services as a mediation mechanism for the context manager to access the functionality provided by the PWRDFContextBroker. The first service, PWEIdInfo, enables the CMI to obtain the identification and classification of the PWE web site. The second service, getRDFContextEndpoint, provides the context manager with the corresponding identification of the service userContextGathering exposed by PWRDFContextBroker. The getRDFContextEndpoint service also provides the SmartContext-compliant context requirements specification of the PWE web site. The third service, setRDFSensorEndpoint, is consumed by the context manager to provide PWRDFContextBroker with the identification of the service PWSiteContextGathering exposed by PWRDFSensor in the ContextManagementInfrastructure composite. The PersonalWebInstrumentation component also identifies the relevant context information from the entities the user interacted with. This information is sent to PWRDFContextBroker through the relevantContext service.

4.3 The personal web infrastructure domain

This domain contains the core of the CMI, the ContextManagementInfrastructure composite. This composite is defined by two components deployable at run-time, RDFContextProvider and PWRDFSensor; and two composites, ContextManager and PersonalWebIntegrator (cf. bottom right part of Fig. 5).

The context manager composite. Whenever a new PWE web site is integrated into the user’s personal web sphere, and the site is not been already served by the CMI, this composite deploys and configures one instance of both RDFContextProvider and PWRDFSensor at run-time. These components are configured dynamically using the proper endpoints to communicate the context manager with the PWE web site. Once the RDFContextProvider is deployed and executed, it uses its PWSiteContextEndpoint reference to provide the PWE site with the context information that describes the user’s interest and situations. Then, the PWE web site uses this information to pro-
Figure 5: SCA architecture of our dynamic CMI

provide the user with information and services according to her matters of concern. Similarly, the \textit{PWSiteContextGathering} service, provided by the \textit{PWRDFSensor}, is consumed by the \textit{PWRDFContextBroker}, thus enabling the CMI to acquire context information from web resources.

To control the adaptation of the CMI and orchestrate the management of the context life cycle, the \textit{ContextManager} composite implements a feedback loop defined by the four components presented in Fig. 6. This feedback loop controls the dynamic deployment of RDF context providers and RDF context sensors in the \textit{PersonalWebInfrastructure} domain. For this, the \textit{PersonalWebEnabledSite} domain must (i) provide its context provision endpoint: the target URI of the \textit{userContextGathering} service exposed by the \textit{PWRDFContextBroker}; and (ii) expose a service to receive the CMI’s context gathering endpoint: the target URI of the \textit{PWSiteContextGathering} service exposed by the \textit{PWRDFSensor}. These endpoints are dynamically manipulated by feedback loops that control mediation mechanisms supported by WebSphere Enterprise Service Bus (ESB) and WebSphere Service Registry and Repository (WSRR) Version 7.0. [6].

The first component of the feedback loop is \textit{ContextMonitor}. This component receives PWE requests generated by the user from the PWE browser through \textit{PWEEnabledService}. Then, it identifies whether the request corresponds to a PWE web site. These two activities occur at the same time the user browses the web site. If the request corresponds to a PWE web site, and the user accepts its integration into her personal web sphere, the CMI
asks the user for the privacy settings. The supported values for this configuration are: (i) always provide and acquire context information; (ii) always ask for privacy settings before starting the first web interaction; (iii) always provide but never acquire context information; and (iv) never provide but always acquire context information. Then, the ContextMonitor component, through the promoted reference getRDFEndpointRequest, obtains the endpoint that must be used to dynamically configure and deploy RDFContextProvider (cf. Fig. 5). The second component of the feedback loop, ContextManagerAnalyzer, correlates the information that identifies and classifies the PWE web site with the information within the user’s context sphere (cf. Sect. 3.2), in order to understand the web site’s context requirements. The ContextManagerPlanner component uses SmartContext to generate the adaptation plan that defines the sensors, and the context providers that must be deployed to enable context gathering and provision. Plans are dynamically generated as scripts that state the sequence of commands required in order to configure and deploy SCA composites using WebSphere APIs. Finally, the ContextManagerDeployer component, acting as an executor, configures and deploys RDFContextProvider and PWRDFSensor in the PersonalWebInfrastructure domain according to the plan. A similar process is applied for deploying instances of ContextManagerExecutor and ClientRDFSensor in the user’s client domain (e.g., in a personal PWE mobile device).

**The personal web integrator composite.** The components of this composite integrate the context information gathered from PWE web sites into the user’s personal web sphere, reason about the context information to be provisioned to these sites, and manage context disposal (cf. Fig. 7). For this, in light of the RDF-based context model that describes the user’s context sphere, the RDFAnalyzer component processes the context specification provisioned by the PWRDFSensor component through the integratePWSContext service (cf. Fig. 5). Then, if the analyzer identifies the necessity of adapting the user’s personal context model, the ContextModelIntegrator component, acting as part of the planner and the executor, uses the information received from the analyzer to integrate new context entities or eliminate those that are no longer interesting to the user, or have completed their life cycle (context disposal). New context entities are integrated as explained in Sect. 3.2, using the integratePWSContext and integrateLocalContext services. For supporting context provision, RDFAnalyzer correlates the information in the user’s context sphere as explained in Sect. 3.2.

**Figure 6: The ContextManager composite**

**Figure 7: The PersonalWebIntegrator composite**

5 Discussion and Related Work

In this paper we proposed a novel approach to dynamic context management based on the combination of self-adaptive instrumentation
Linked data provides suitable mechanisms to support context representation and discovery on the web. The semantic web offers programming platforms suitable for implementing context gathering and provision. However, due to the dynamic nature of context in web environments, the effectiveness of our approach depends on the efficiency of the adaptive instrumentation. Moreover, controlling the vast extension of inferable context facts is also a challenge to guarantee context management efficiency. In the current version of our implementation this is addressed by disposing unused context information according to the user’s preferences.

Security and privacy are also necessary features in the management of the context information life cycle [17]. We address privacy through the integration of the user in the context management loop. Thus, the user is the one who decides whether or not to share and store her context information. User-driven web integration comprises diverse sources of context information, which might be selectively accepted or rejected by context consumers based on their origin. In light of this, we consider data provenance and trust management as required security features for our CMI. Provenance information must therefore accompany the provisioned context. We plan to use RDF Named graphs [3] as the mechanism to maintain and communicate provenance information. Named graphs can also be exploited for realizing access control on context information as mentioned in Sect. 4.3. Detailed access control policies can be attached as signatures to portions of the user’s context using named graphs as a delimiter. These signatures would be then evaluated by the context manager.

In the initial stage of this work, we surveyed different approaches for context management in different application domains [17]. Other dynamic approaches to context management exist. For instance, Hu et al. proposed a context management system that exposes autonomic behavior to support the replacement of failing context sensors at run-time [10]. A remarkable feature of their approach is the integration of sensor description standards to facilitate the discovery of sensing services. However, the context model supporting context management in Hu et al.’s approach is constrained by specific context types suitable only for pervasive environments. In our approach SmartContext provides extensible context models applicable to any domain of the smart Internet.

Another example is the context management framework proposed by Euzenat et al. [7]. They also use RDF and semantic web technologies for context representation and context discovery. Another feature in common with our CMI is the integration of new context sensors at run-time. Nevertheless, in their approach relevant context information must be fully specified at design-time. This approach is not suitable for smarter web applications where the user is the most important actor of the web experience. Our approach is different. It integrates the user in the context management loop. This way we not only enable users to modify their context preferences at run-time in a transparent way, but also provide a context management infrastructure with self-adaptive capabilities to support changes in the user’s context model. Finally, among all the research works surveyed in our systematic review [17], our approach is the only one that manages context information dynamically as a first level entity, while integrating the user as an essential part of the context life cycle.

6 Conclusions and Future Work

Dynamic context management is absolutely crucial for enhancing the user web experience. In order to advance user-centric web integration and hence supporting the vision of the personal web, we propose to treat context information about web entities and users as first level entities. These entities are explicitly represented and managed according to user’s interests. Furthermore, in order to deal with the dynamic nature of context, we apply adaptive software instrumentation to the gathering, provision, and reasoning of context under changing requirements. In this paper we proposed an innovative approach to optimize web users experience based on dynamic context management techniques supported by feedback loops and the
semantic web. Based on an exhaustive analysis and literature survey allowed us to confirm the novelty of our contribution. Our research is a promising direction to advance the state-of-the-art of context-aware web applications. Implementing dynamic context management as a level of indirection between web systems and the user’s environment is an important contribution toward context-aware user-centric web platforms.

Future work will focus on the assessment of our proposal. We are currently implementing experiments to validate properties such as scalability and performance. This assessment will be conducted using our previous contribution on evaluating self-adaptive software systems [19]. Finally, we are working on the application of a user study to evaluate the functional feasibility of the context management infrastructure in an industrial smarter commerce scenario.

Acknowledgements

This work was funded in part by the National Sciences and Engineering Research Council (NSERC) of Canada (CRDPJ 320529-04 and CRDPJ 356154-07), IBM Corporation via the CSER Consortium, Icesi University, Cali, Colombia and University of Victoria, British Columbia, Canada.

About the Authors

Norha M. Villegas is a PhD candidate under the supervision of Dr. Hausi A. Müller, Department of Computer Science, University of Victoria, Canada. She is a CAS student at the Center for Advanced Studies at the IBM Toronto Laboratory. Her dissertation focuses on the application of dynamic context management techniques for the optimization of self-adaptive software systems. Her research interests include control theory, autonomic computing, dynamic context management, context-awareness, semantic web, and service-oriented systems. Norha M. Villegas received a Diploma Degree in Systems Engineering and a Graduate Degree in Organizational Informatics Management in 2002 and 2004, from Icesi University, Cali, Colombia.

Hausi A. Müller is a Professor, Department of Computer Science and Associate Dean of Research, Faculty of Engineering at University of Victoria, Canada. He is a Visiting Scientist at the Center for Advanced Studies at the IBM Toronto Laboratory (CAS), CA Canada Inc., and the Carnegie Mellon Software Engineering Institute (SEI). Dr. Müller’s research interests include software engineering, self-adaptive and self-managing systems, context-aware systems, and service-oriented systems. He serves on the Editorial Board of Software Maintenance and Evolution and Software Process: Improvement and Practice (JSME). He served on the Editorial Board of IEEE Transactions on Software Engineering (TSE) 1994-2000, 2005-2009). He is Chair of the IEEE Technical Council on Software Engineering (TCSE). Dr. Müller received a Diploma Degree in Electrical Engineering in 1979 from the Swiss Federal Institute of Technology (ETH), Zürich, Switzerland and MSc and PhD degrees in Computer Science in 1984 and 1986 from Rice University in Houston, Texas, USA.

Juan C. Muñoz is a Master student in Department of Information and Communications Technology, at Icesi University, Cali, Colombia. His research focuses on the application of context management and semantic web technologies to the implementation of user-centric smarter commerce platforms. He has participated in different industrial software development projects as a software developer, system analyst, and project manager. He is a lecturer in the Faculty of Engineering, at Icesi University since 2006, where he received a Diploma Degree in Systems Engineering in 2005.

Alex Lau is a Research Staff Member at the Center for Advanced Studies (CAS) at IBM Canada Software Laboratories, Toronto. His research focuses on service-oriented architecture (SOA), BPM tools, and business events. Prior to joining CAS, he was a Software Engineer for IBM Canada, where he worked on different Websphere BPM products including Websphere Integration Developer
and Websphere Business Modeler. Alex Lau received a Diploma Degree and a Master Degree in Mathematics, from the University of Waterloo, Canada.

Joanna Ng is the Program Director of the Center for Advanced Studies (CAS) at IBM Canada Software Laboratories, Toronto. She is a Senior Technical Staff Member of IBM Software Group. She has held various senior management and architect positions in product development and software strategy within IBM. Joanna Ng is an IBM Master Inventor with a long track record of profitable innovations. She has been granted over twenty five patents from various countries in research areas related to smart Internet such as mobile commerce, voice-enabled portal, commerce portal, retail industry solutions, service-oriented architecture (SOA), asset repository, semantic and web technologies.

Chris Brealey is a Senior Technical Staff Member with IBM Software, Rational Enterprise Architecture, Design and Construction. Chris is based at the IBM Toronto Canada Lab. He is responsible for the architecture of development tools pertaining to SOA, SCA and Cloud Computing in Rational Application Developer. He is a regular collaborator in work groups and fellowship projects in the Centers for Advanced Studies (CAS). Chris previously lead the architecture and delivery of the RAD Web services tools. He joined IBM in 1989 after his graduation from the University of Victoria, Canada, with a BSc in Computer Science.

References


