CCMF, Computational Context Modeling Framework – An Ontological Approach to Develop Context-Aware Web Applications

Luis Paulo Carvalho[1] and Paulo Caetano da Silva[2]
[1]UNIFACS-University of Salvador/IRT-Instituto Recôncavo de Tecnologia
[2]UNIFACS-University of Salvador
Brazil

1. Introduction

The purpose of software is to help people to perform their activities and fulfill their objectives. In this regard, the human-software relationship could be enhanced if software could adapt to changes automatically during its utilization (Brézillon, 1999). Context is defined by (Dey, 2001) as any type of information which characterizes an entity. An entity is any person, place or object that is relevant to the interaction between users and software. According to (Dey & Abowd, 1999) context-awareness is the capability of software to use context to offer services to users. For instance, a context-aware system may trigger an alarm in a device near to a user to remind him of the departure time of a planned trip (Yamato et al. 2011). In this example, context is used to provide a service to the user: the reminding of a personal activity. It is not, however, a trivial task to associate context with software in the same level of abstraction as humans do when they communicate with each other. (Dey & Abowd, 1999) considers that such ability is naturally inherited from the richness of the human languages and the common understanding of how the world works and from an implicit understanding of daily situations. Thus, with the intention of enhancing the offering of services from software to humans, it is important to transmit these capacities to computational environments.

As maintained by (Sheng & Benatallah, 2005), web services have become a promising technology for the development of internet-oriented software. Services are autonomous platform-independent software that executes tasks ranging from providing simple answers to users requests to the execution of complex processes. Web services are services that utilize the internet and its open technologies, e.g. WSDL, Web Service Description Language, SOAP, Simple Object Access Protocol, UDDI, Universal Description, Discovery and Integration, and XML, eXtensible Markup Language, to supply functionalities to other applications (Berbner et al. 2005). (Kapitsaki et al. 2009) assure that the handling of context is of vital importance to web services, since it promotes dynamic behavior, content adaptation and simplicity of use to end users. However, the association between web services and context is not easy to achieve because adequate mechanisms are not offered to developers in order to support the description and representation of context-related information and its later utilization by web services.

Considering the lack of appropriate technologies to extend software to integration of context and web services, this work proposes:

1. A framework, CCMF (Carvalho & Silva, 2011), Computational Context Modeling Framework, which relies on the reuse of artifacts and tools to automate analysis and development activities related to the making of context-aware web applications;
2. The instantiation of CCMF considering 2 different case studies: (a) Integration of CCMF and CCMD (Carvalho & Silva, 2011), Computational Context Modeling Diagram (to be presented in Section 3.1); (b) The embedding of ontologies in CCMF. Both cases intending to enable the development of context-aware web applications;
3. The analysis of the coupling between the framework and the 2 targeted technologies (CCMD and Ontologies) in order to evaluate the advantages and disadvantages of each approach.

This work is organized as follows: Section 2 presents related works. The framework is described in Section 3. In Section 4, CCMF is coupled with CCMD and ontologies to develop a context-aware web application (both approaches are compared). Section 5 enfolds conclusions and future works.

2. Related works

(Carvalho & Silva, 2011) gathered requirements from related literature - e.g. (Topcu 2011), (Hoyos et al. 2010), (Dey et al. 2001), (Vieira, 2008), (Bulcão, 2006), (Schmidt, 2006) - with the intention of enumerating important characteristics for the modeling of the information which influences the utilization of software and to provide guidelines to base the creation of the framework. Table 1 lists the requirements.

	Requirement	Purpose
I	It must support the development of context-aware software as a two-phase task: specification of context information (structure) and adaptation	The separation into two phases promotes the decoupling of aims, allowing designers to focus on specific activities related to each development phase
II	It must categorize the information into context dimensions	By modeling the context focus and dimensions, designers are able to orderly identify and structure context information, promoting the readability of the model
III	It has to identify the context focus of a task	
IV	It must support the transfer of context information and artifacts between development phases	The effort required to perform next development steps (e.g. modeling of adaptation) is lessened by the input of artifacts from previous phases (e.g. modeling of structures)
V	It must promote the diversity of context information in a domain-independent manner	So that designers can model context-aware systems to automate tasks of a variety of scenarios
VI	It has to support the reuse of distributed computing systems such as services	To base context adaptation on web services API, e.g. Google Agenda API (GAgenda, 2011)

Table 1. Requirements for developing context-aware software (Carvalho & Silva, 2011)

The following works were evaluated against the requirements (listed in Table 1):

1. CMS-ANS (Bonino et al. 2007), Context Management Service/Awareness and Notification Service, a framework that allows context sources to publish information and client software to be notified when specific contexts are acquired;
2. CONON (Wang et al. 2004), CONtext Ontology, a two-layered ontology intended to promote the sharing of context structures among agents and services;
3. CMP (Simons, 2007), Context Modeling Profile, uses stereotypes to extend the class diagram of UML to model context. In the same way, ContextUML (Sheng & Benatallah, 2005) adds stereotypes to the UML (in specific, to the class diagram) to model context-aware web services;
4. CEMantTIKA Case (Patrício, 2010), composed of a set of customized diagrams - based on the Eclipse platform and JBoss Drools (JBDrools, 2009) - that model context structures and adaptations of context-aware software;
5. (Bastida et al. 2008) proposes WSBPEL (WSBPEL, 2007), Web Service Business Process Execution Language, to model adaptation based on context information extracted from software requirements;
6. (George & Ward, 2008) modify the WSBPEL engine to support the addition of context variables and sources (i.e. the sources are used to fill information in the variables);
7. CAMEL (Grassi & Sindico 2009), Context-Awareness ModEling Language, composed of UML-oriented diagrams made specifically to model context structures and adaptation;
8. (Yamato et al. 2010) proposes dynamic adaptations of composite web services utilizing semantic context metadata to select equivalent functionalities from clusters of web services.

Table 2 shows the result of the evaluation of the works against the proposed requirements (filled cells indicate that the requirement was fulfilled).

	I	II	III	IV	V	VI
CMS-ANS		■				■
CONON		■			■	
CMP	■				■	
ContextUML	■			■	■	■
CEManTIKA Case	■		■	■	■	
(Bastida et al. 2008)	■			■	■	■
(George and Ward 2008)	■			■	■	■
CAMEL	■			■	■	
(Yamato et al. 2010)				■	■	■

Table 2. Evaluation of related works against the requirements (Carvalho & Silva, 2011).

Since the aforementioned related works do not fulfill all of the requirements (described in Table 1), CCMF is proposed as set of activities intended to automate the analysis and development of context-aware web applications. In Section 3, the framework is described and it is discussed its association with reusable development languages, tools and artifacts. A case study in which CCMF is applied is presented in Section 4.

3. CCMF, Computational Context Modeling Framework

CCMF is composed of a set of analysis and development activities which are intended to lessen the effort demanded by the development of context-aware web applications. This is achieved from: (a) the reuse of artifacts and third-part tools, modeling languages and technologies and (b) the automation of the execution of targeted activities, which are described in next paragraphs.

As shown in Figure 1, the framework is composed of two layers containing specific activities to be carried by developers. Its upper layer comprises the activities related to the definition of structures of context information and its externalization to the adaptation modeling mechanism, a WSBPEL diagram. Such externalization is made possible by the transformation of the context structures into a context medium. Provided that XML-based languages enable the interoperation of computational agents, e.g web services (Alboaie et al. 2003), XSD (XSD, 2001), XML Schema Definition, documents are used by the framework to enable the utilization of context by web services integrated to the WSBPEL diagram. Respectively, the definition of context structures and the transformation of these structures into context mediums are performed by the activities identified by numbers 1 and 2.

Once the context medium is created, it can be transformed into language-specific context classes. Considering, for instance, JAVA as the development language, XMLBeans API (XMLBeans, 2009) can be used to transform the XSD schema into JAVA serializable classes. The resulting classes can be instantiated as objects that are capable of encapsulating their attributes into XML documents. Later on, web services can rely on such documents to exchange complex context data between each other, i.e. the serialization via XML documents is necessary to interoperate web services in a manner that the information about context is used to parameterized the adaptation. The generation of the serializable classes is performed by the activity number 3.

After having modeled the context structures, the developer must define how the context information must be used to automate adaptations. This activity (identified by number 4), the first one of the framework´s lower layer, depends on WSBPEL to base the context adaptation on web services. In this case, web services must be gathered and integrated to a WSBPEL diagram in order to utilize context information to parameterize responses to situations of use. Along with the deployment of the composite context-aware web service (by activity number 5), a WSDL document is created. This document describes the web service with the purpose of allowing the remote calling of its functionalities by other computational agents (e.g. handheld-embedded applications, other web services). To ease the effort required by the creation of these agents, the WSDL document can be transformed into language-specific source code (by executing the activity identified by the number 6). The source code is intended to provide ways to client software to access the composite web service in order to be served by adaptation functionalities.

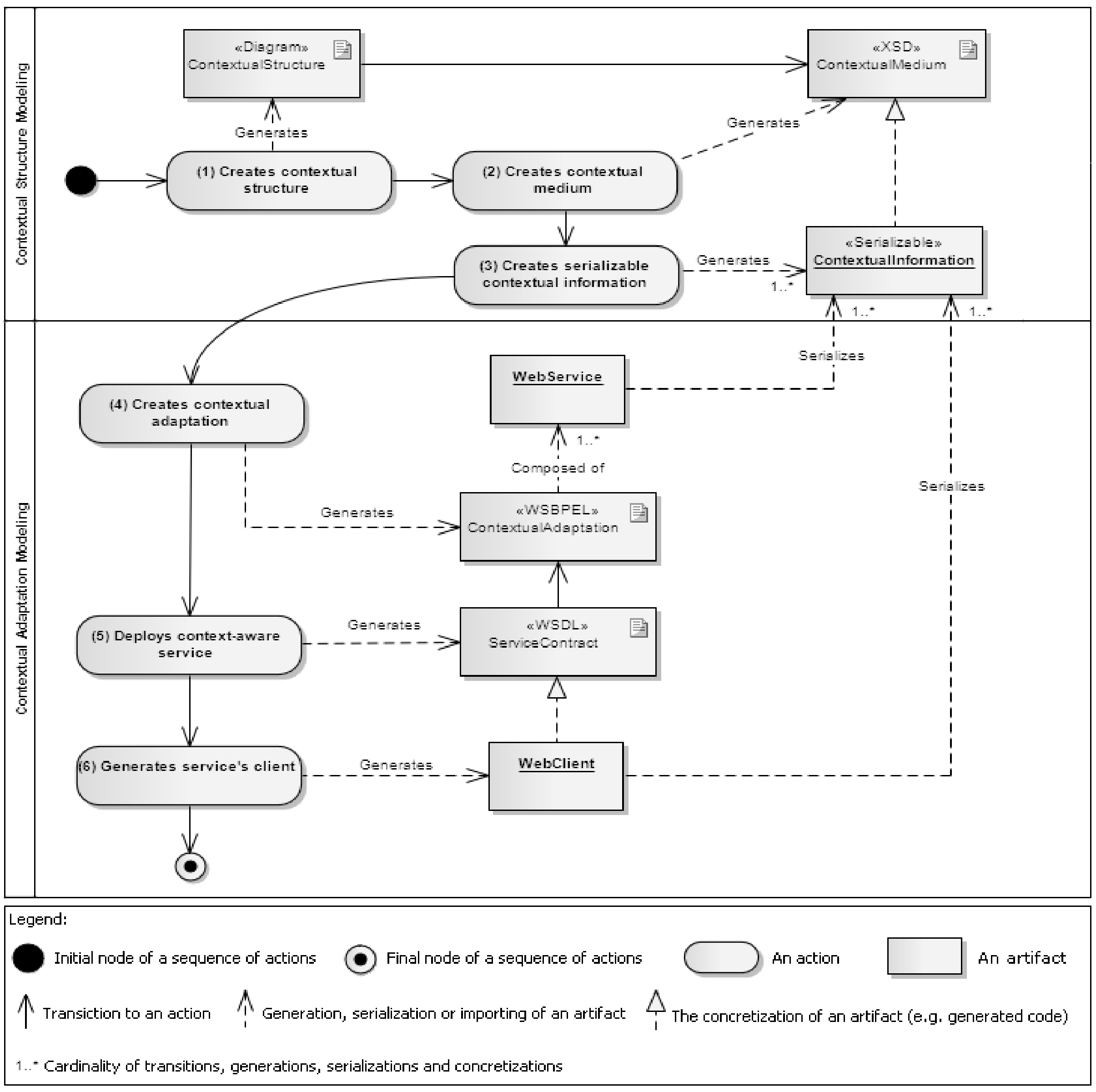

Contextual Structure Modeling
Contextual Adaptation Modeling
«Diagram» ContextualStructure
«XSD» ContextualMedium
Generates
(1) Creates contextual structure
(2) Creates contextual medium
(3) Creates serializable contextual information
«Serializable» ContextualInformation
1..*
Serializes
WebService
(4) Creates contextual adaptation
Composed of
«WSBPEL» ContextualAdaptation
(5) Deploys context-aware service
«WSDL» ServiceContract
(6) Generates service's client
WebClient
Legend:
Initial node of a sequence of actions
Final node of a sequence of actions
An action
An artifact
Transiction to an action
Generation, serialization or importing of an artifact
The concretization of an artifact (e.g. generated code)
1..* Cardinality of transitions, generations, serializations and concretizations

Fig. 1. CCMF – Computational Context Modeling Framework (Carvalho & Silva, 2011)

Provided the activities depicted in Figure 1, CCMF is capable of:

1. Modeling structures of context information (upper layer of the framework) and the adaptation (lower layer of the framework);
2. Enabling the reuse of development artifacts, e.g. by transforming a context medium (a XSD document) into XML-based classes in order to serialize complex context data between web services;
3. Supporting the reuse of distributed computing systems such as web services. In this case, the adaptation mechanism is placed "in the cloud" and it can be reused by other computational agents over the internet.

In Section 3.1, it is described a diagram, CCMD, Computational Context Modeling Diagram, which automates the execution of activity number 1 of CCMF, the modeling of context structures, i.e. CCMD can be coupled with the upper layer of CCMF to define the sets of information which may interfere in situations of use of software. In Section 3.2 ontologies are introduced as a replacement for CCMD, being analyzed what are the necessary adaptations to be applied to the framework in order to enable the use of ontologies to model context-aware web applications.

3.1 CCMD – Computational Context Modeling Diagram

CCMD is composed of a set of stereotypes which allows the creation of diagrams to model concepts related to computational context, e.g. the context focus and dimensions. According to (Brézillon & Pomerol, 1999) the context focus corresponds to a step in the solution of a problem or in a decision making process. (Vieira, 2008) states that the focus can represent a relationship between an agent and a task, in which the agent is responsible for performing the task. For instance, referring to the modeling of software that bases the scheduling of meetings on computational context, the focus might indicate that a secretary (the agent) must perform the task "prepare meeting". The focus is important to context modeling because it enables developers to identify specific sets of context information in relation to the task executed by an agent. Once the focus is identified, the related information can be grouped as context dimensions. As indicated by (Brézillon & Bazire, 2005) the context dimensions enables the categorization of context information and have the main purpose of helping software designers to specify, model and fill information into adequate structures (Bulcão, 2006). CCMD models the following context dimensions identified by (Abowd & Mynatt, 2000):

1. "Location" represents spatial characteristics of the context information;
2. "Temporal" ("Time") comprises any date/time-related information of the context;
3. "Participant" represents entities (other than the agent) which participates in the execution of the task;
4. "Motivation" ("Preferences") is related to the objectives of the agent and participants;
5. "Activity" corresponds to activities performed during the execution of the task.

The task, the context focus and dimensions are illustrated in Figure 2. Number 1 points to the task. The context focus is represented by the stereotype linked to number 2. Number 3 is associated with the stereotypes that represent the context dimensions.

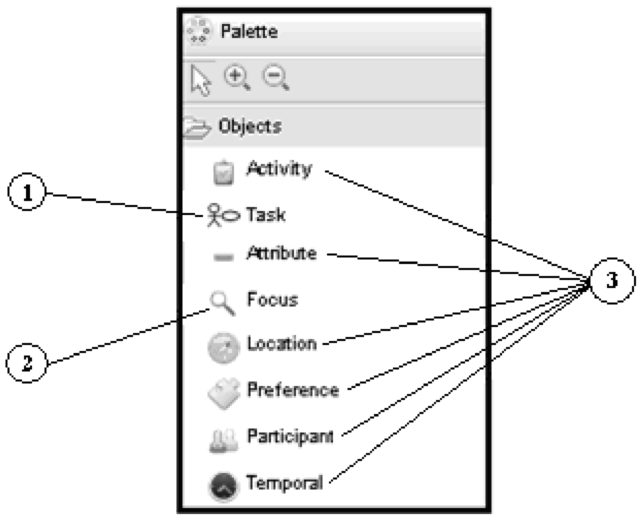


Fig. 2. Stereotypes of CCMD (Carvalho & Silva, 2011).

A concrete implementation of the stereotypes of CCMD can be generated via EuGENia (EuGENia, 2011) plugin to embed the modeling of context in the Eclipse IDE. EuGENia defines a declarative language which abstracts away the details of the coding of diagrams. The declarative metadata of CCMD is transformed by EuGENia into artifacts which input graphic-related data into Eclipse's GMF (GMF, 2010), Graphical Modeling Framework, and EMF (EMF, 2009), Eclipse Modeling Framework. As a result, it is generated a diagram (CCMD) to be used via Eclipse IDE to model context information structures. The graphical elements of CCMD are shown in Figure 3. Next to number 1, it is represented the "Task" from which the context focus is extracted. The "Focus" is placed next to number 2. The "Preference" is modeled by the element next to the number 3. Each "Participant" is symbolized by the element pointed by number 4. The "Location" is positioned near to number 5. The "Activity" is represented by the element next to number 6. The "Temporal" ("Time") dimension is situated nearby the number 7.

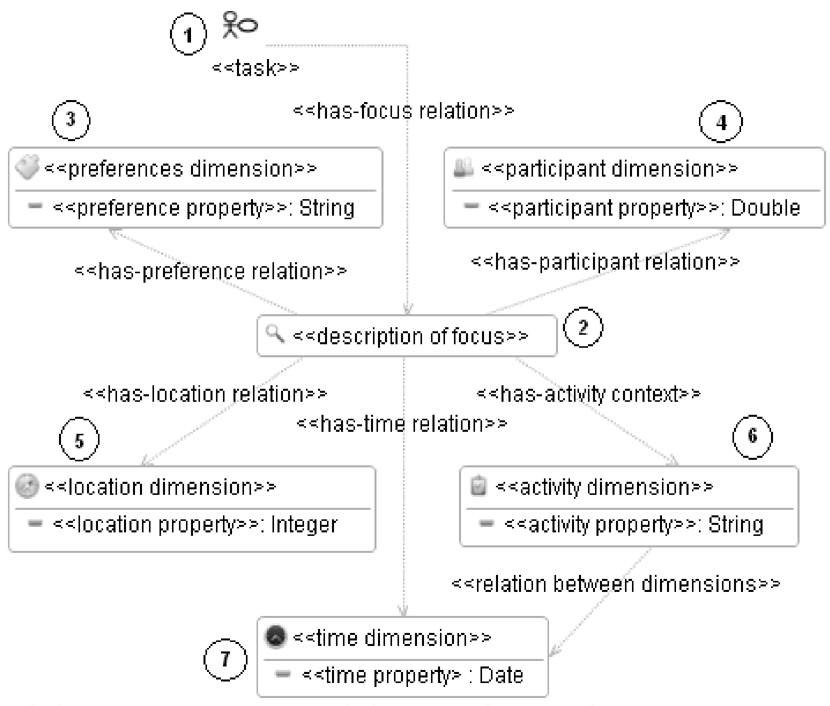

1
<<task>>
<<has-focus relation>>
3
<<preferences dimension>>
<<preference property>>: String
4
<<participant dimension>>
<<participant property>>: Double
<<has-preference relation>>
<<has-participant relation>>
<<description of focus>>
2
<<has-location relation>>
<<has-activity context>>
<<has-time relation>>
5
<<location dimension>>
<<location property>>: Integer
6
<<activity dimension>>
<<activity property>>: String
<<relation between dimensions>>
7
<<time dimension>>
<<time property> : Date

Fig. 3. Concrete implementation of CCMD stereotypes (Carvalho & Silva, 2011)

The instantiation of CCMF and its coupling with CCMD is described in Section 4.1. The framework is used to implement a context-aware web application. The modeling activities enumerated in Figure 1 are exemplified as well as the reuse of tools and artifacts.

3.2 Ontologies

As stated by (Noy, 2004; Chen et al. 2004), ontologies are believed to be a key feature in the making of context-aware distributed systems due to the following reasons:

1. Ontologies enable the sharing of knowledge by open dynamic agents (e.g. web services);
2. Ontologies supply semantics for intelligent agents to reason about context information;
3. Ontologies promote the interoperability among devices and computational agents.

Considering that the abovementioned advantages ensure that ontologies side with the purpose of CCMF, which is to promote the reuse and interoperability of distributed computational agents (e.g. web services) with the intention of automating the creation of context-aware web applications, it is proposed the coupling of CCMF with an ontological approach. In this case, it must be surveyed how ontologies are capable of supporting the representation of context structures and the generation of context-aware adaptations mechanisms.

(Bontas et al. 2005) defines ontology reuse as the process in which available (ontological) knowledge is used to generate new ontologies. By reusing existent ontologies the cost of implementation is reduced, since it avoids the manual codification of a new one. Moreover, two different ontologies can be bound together as one to represent concepts of broader domains, i.e. a given ontology can be associated with others with the intention of modeling concepts of a domain in order to represent the sum of the information represented by each of the combined ontologies. Therefore, the framework must be evolved as to allow the collecting and binding of ontologies with the intention of supplying structures of information to model computational context.

Another aspect of ontologies that motivates modifications on the development activities of CCMF is related to their capability of having a dual purpose (Reinisch et al. 2008): ontologies are able to represent knowledge and also to store and generate instances of such knowledge to interoperate agents. In comparison with the set of activities of the former framework (Figure 1), the handling of CCMD and the XSD/XML-based artifacts can be replaced by ontologies, because they can be accessed directly by the web services of the context-aware composition in order to enable the saving and retrieval of the context information. As a consequence, the effort required to model a context-aware composite web service is lessened, because it is not necessary to deal with the instantiation and manipulation of serializable objects and XML documents, i.e. the utilization of ontologies causes the removal and substitution of activities from the original framework (the one illustrated in Figure 1 of Section 3).

The modified framework is shown in Figure 4. The activity identified by number 1 represents the collecting and binding of existent ontologies in the making of a new one, which must be suitable to model information of the context-aware application's domain. The second activity is that of customizing the ontology to better represent context information. This activity can be exemplified by the definition of associations between natively dissociated classes and/or the addition of new classes and attributes to candidate ontologies. The third activity creates the context adaptation and bases it on the utilization of composite web services. The web services of the composition are able to add and select instantiated individuals from the ontology in substitution to serializations via XML documents. The deployment of the composite web service is performed by activity number 4. The generated WSDL contract is reused in the making of client software by activity number 5.

Thus, the adapted framework, hereafter O-CCMF, Ontology-driven Computational Context Modeling Framework, is able to (re)use ontologies through a smaller set of activities dedicated to the development of context-aware web-applications.

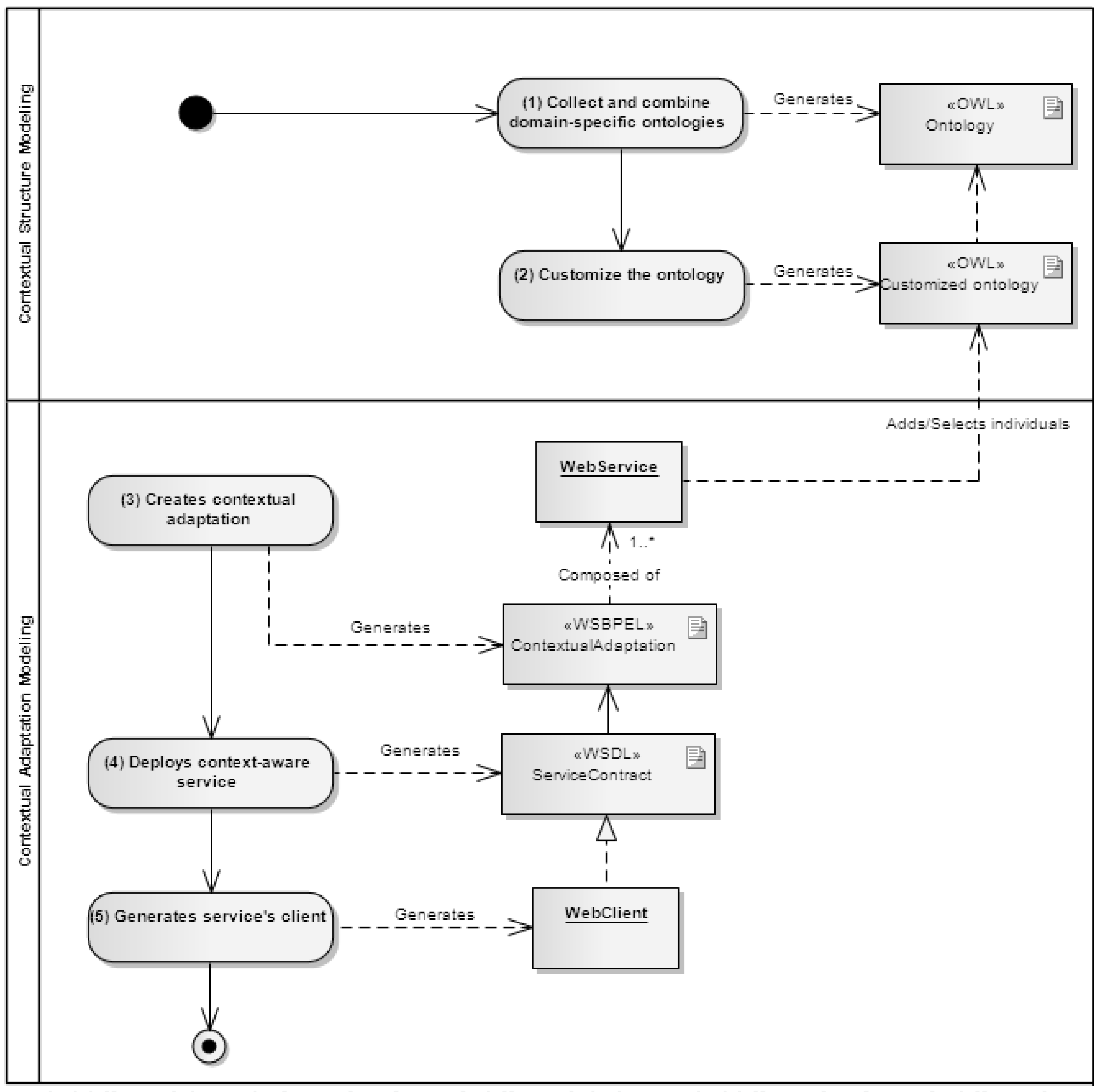

Contextual Structure Modeling
(1) Collect and combine domain-specific ontologies
Generates
«OWL» Ontology
(2) Customize the ontology
Generates
«OWL» Customized ontology
Contextual Adaptation Modeling
Adds/Selects individuals
WebService
(3) Creates contextual adaptation
1..*
Composed of
Generates
«WSBPEL» ContextualAdaptation
(4) Deploys context-aware service
Generates
«WSDL» ServiceContract
(5) Generates service's client
Generates
WebClient

Fig. 4. O-CCMF – Ontology-Driven Computational Context Modeling Framework

The two versions of CCMF, the one coupled with CCMD and O-CCMF are exemplified as study cases in Section 4. They are used to develop the same context-aware web application. Later on, both approaches are evaluated against the requirements identified in Section 2 (Table 1).

4. Case studies

As a proof of concept of the application of CCMF and O-CCMF, it is proposed the creation of a context-aware meeting alert application. The frameworks are used to model and develop an application that must send alerts to participants of meetings according to the requirements defined by (Antoniou et al. 2007):

1. If the participant is located near to the place where the meeting is going to happen, he must receive an alerting message 5 minutes before the meeting;
2. If the participant is located in a place far from where the meeting is going to happen, the message must be sent within 40 minutes;
3. If the meeting is going to happen in the rush hour, 10 minutes are added to the interval;
4. If it is raining, another 10 minutes are added;
5. If the meeting's subject is about giving a class, 30 minutes are added in order to allow the participant to prepare himself for the class.

4.1 Developing the meeting alert application with CCMF

The first activity of the framework is that of modeling the context information structures. Figure 5 shows a graphical instantiation of CCMD which is used to represent the context data that parameterizes the context adaptation of the meeting alert application. The element next to number 1 represents the task under which a context focus is identified (next to number 2). The focus aids designers in determining the specific set of context information that is necessary to enable the adaptation, i.e. the combination of tasks and focus helps designers to restrain the scope of analysis of context structures. Once the focus is identified, the datasets of context information can be added to CCMD. The meeting is symbolized by the element next to number 3. The location of the meeting is represented by the element next to number 5. The temporal dimension is represented by the element identified by number 6 and contains information about the starting and ending datetime of the meeting. The list of participants is represented by the element next to number 4. Each participant has its own geographic location (latitude/longitude coordinates) which is represented by the element next to number 7. The locations of the participants are used to calculate their distances from the location of the meeting. The preferable weather condition is represented by the element next to number 8.

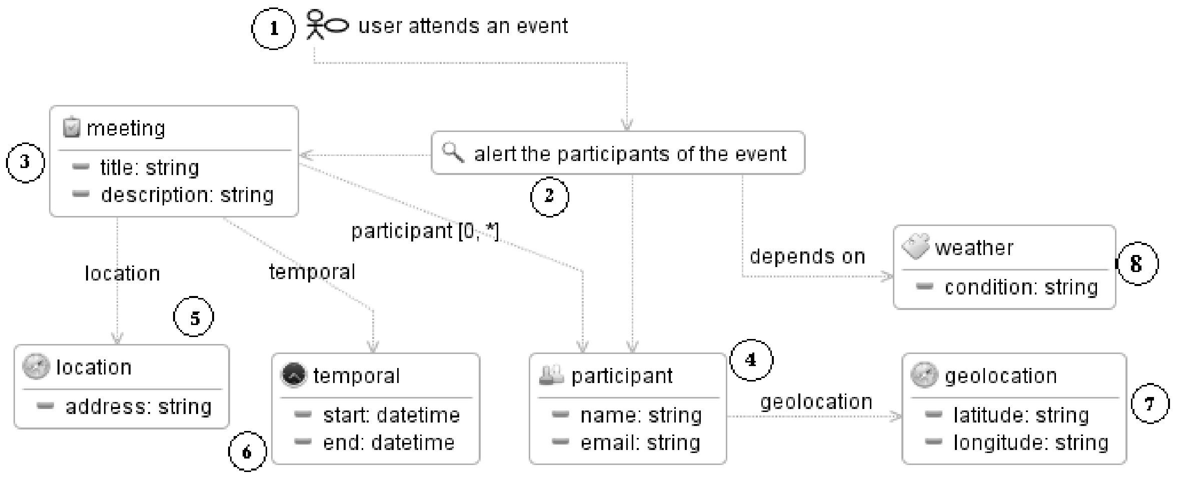


Fig. 5. Meeting alert context information modeled by CCMD (Carvalho & Silva, 2011)

After using CCMD to define context structures, a context medium must be generated (activity 2 of the framework). The purpose of the context medium is to integrate the framework with different sources of context information (e.g. CCMD). Figure 6 shows an excerpt from a XSD document generated by the CCMD that represents the meeting and its location, its participants and the date/time-related information.

```
<complexType name="meeting">
  <sequence>
    <element name="location" type="gagenda:location" minOccurs="1" maxOccurs="1"/>
    <element name="temporal" type="gagenda:temporal" minOccurs="1" maxOccurs="1"/>
    <element name="participant" type="gagenda:participant" minOccurs="0"
maxOccurs="unbounded"/>
  </sequence>
  <attribute name="title" type="string"></attribute>
  <attribute name="description" type="string"></attribute>
</complexType>
```

Fig. 6. Context medium exported by CCMD as a XSD document (Carvalho & Silva, 2011).

The context medium (XSD document) enables the framework to transform the context structure into other formats of reusable artifacts. For instance, in order to interoperate the web services, with the purpose of implementing the context adaptation of the meeting alert software, the XSD document can be converted into serializable JAVA classes (by the XMLBeans API). This transformation corresponds to the third activity of the framework and it generates a library that makes possible the exchanging of context data between web services, e.g. a certain web service fills context information into a serializable object, which automates the creation of a XML document that is serialized toward other web services. By receiving the XML document as an input, the targeted web service deserializes the context information back into a high level JAVA object. Figure 7 illustrates an example of a XML document which serializes context information related to a meeting.

```
<data:meetings title="Meeting professor Paulo Caetano"
  description="Meeting professor Paulo Caetano to talk about the dissertation"
xmlns:data="http://www.data.agenda.adapters.google.unifacs.edu.br">
  <data:location address="Rua Ponciano de Oliveira, 126, Rio Vermelho, CEP 41950-275, Salvador,
Ba, Brasil">
  </data:location>
  <data:temporal start="2011-08-23T17:30:00.000-03:00" end="2011-08-23T18:30:00.000-03:00">
  </data:temporal>
  <data:participant name="luis paulo" email="luispsc@yahoo.com.br"></data:participant>
  <data:participant name="paulo caetano" email="paulocaetano.dasilva@gmail.com">
</data:participant>
</data:meetings>
```

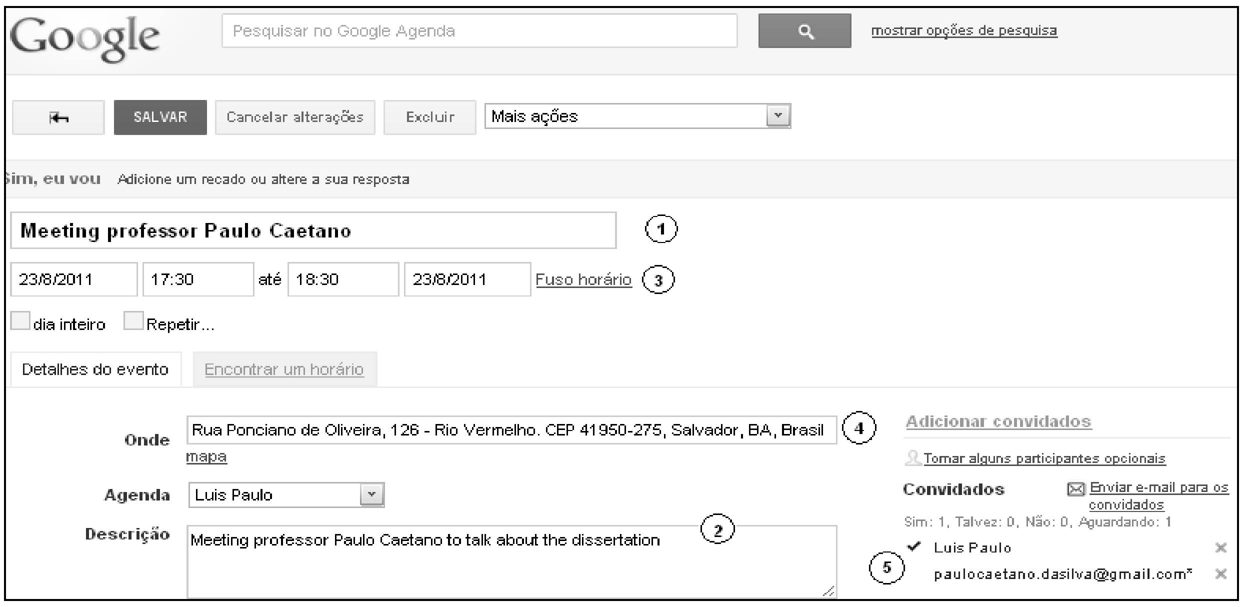
Fig. 7. XML-based serializable information (Carvalho & Silva, 2011).

Once the modeling of the context structures is made available, it must be defined how the software is adapted to situations of use. Prior to designing the adaptation, using the BPEL Visual Designer for Eclipse IDE (BPELEclipse, 2010), web services has to be found so to make possible the collecting and processing of the context data. Table 3 lists services API's used to automate the adaptation of the meeting alert application.

API/Service	Usage
Google Agenda	It supplies information about meetings
Yahoo Weather Forecast	It offers information about weather conditions
Google Geocoding (GGCoding, 2011)	It converts address-based locations of meetings to geographic coordinates
Google Geodirections API (GDirections, 2011)	It calculates the distance from each participant to to meeting's location

Table 3. Services API's used to automate the adaptation of the meeting alert application

The XML document in Figure 7 contains data retrieved from a web service based on the Google Agenda API. Figure 8 shows an example of an event added to the user's agenda. Next to number 1, it is shown the title of the event. Its description is placed near to number 2. Next to 3, the event's starting and ending date and time are shown. Event's location is identified by the number 4. The list of participants is placed near to number 5.


Google
Pesquisar no Google Agenda
mostrar opções de pesquisa
SALVAR
Cancelar alterações
Excluir
Mais ações
Adicione um recado ou altere a sua resposta
dia inteiro
Repetir...
Detalhes do evento
Encontrar um horário
mapa
Adicionar convidados
Tomar alguns participantes opcionais
Enviar e-mail para os convidados
1
2
3
4
5

Sim, eu vou

Meeting professor Paulo Caetano

23/8/2011 17:30 até 18:30 23/8/2011 Fuso horário

Onde: Rua Ponciano de Oliveira, 126 - Rio Vermelho. CEP 41950-275, Salvador, BA, Brasil

Agenda: Luis Paulo

Descrição: Meeting professor Paulo Caetano to talk about the dissertation

Convidados

Sim: 1, Talvez: 0, Não: 0, Aguardando: 1

✓ Luis Paulo

paulocaetano.dasilva@gmail.com*

Fig. 8. Google Agenda as a source of context information (Carvalho & Silva, 2011)

The WSBPEL diagram illustrated in Figure 9 models a workflow in which the web services of Table 3 are used to automate the context adaptation. The action identified by the number 1 request events from Google Agenda. The location of the event is processed to determine the weather condition in the area where the meeting is going to happen. This is performed by the action identified by number 2. Since the location of the event is not expressed as a latitude-longitude pair, Google Geocoding is executed to translate the address of the event into geographic coordinates (next to number 3). The distance from each participant to the event's location is calculated by Google Directions (near to the number 4). Once all of the context information is retrieved and processed by the web services, the WSBPEL diagram evaluates the amount of time within which the alert messages must be sent to participants. If the participant is located near to event's location[1], the message is sent within 5 minutes. If not, the message is sent within 40 minutes (conditional test next to number 5). If it is going

[1] Participants within a radius of 3.000 meters are considered near to event's location.

to rain (test placed near to number 6), another 10 minutes are added. If the event happens during the rush hour, the interval is increased in 10 minutes (condition evaluated next to number 7). If the event is about giving a class[2], another 30 minutes are added to the interval so that the participant will be able to ready himself in order to give the class.

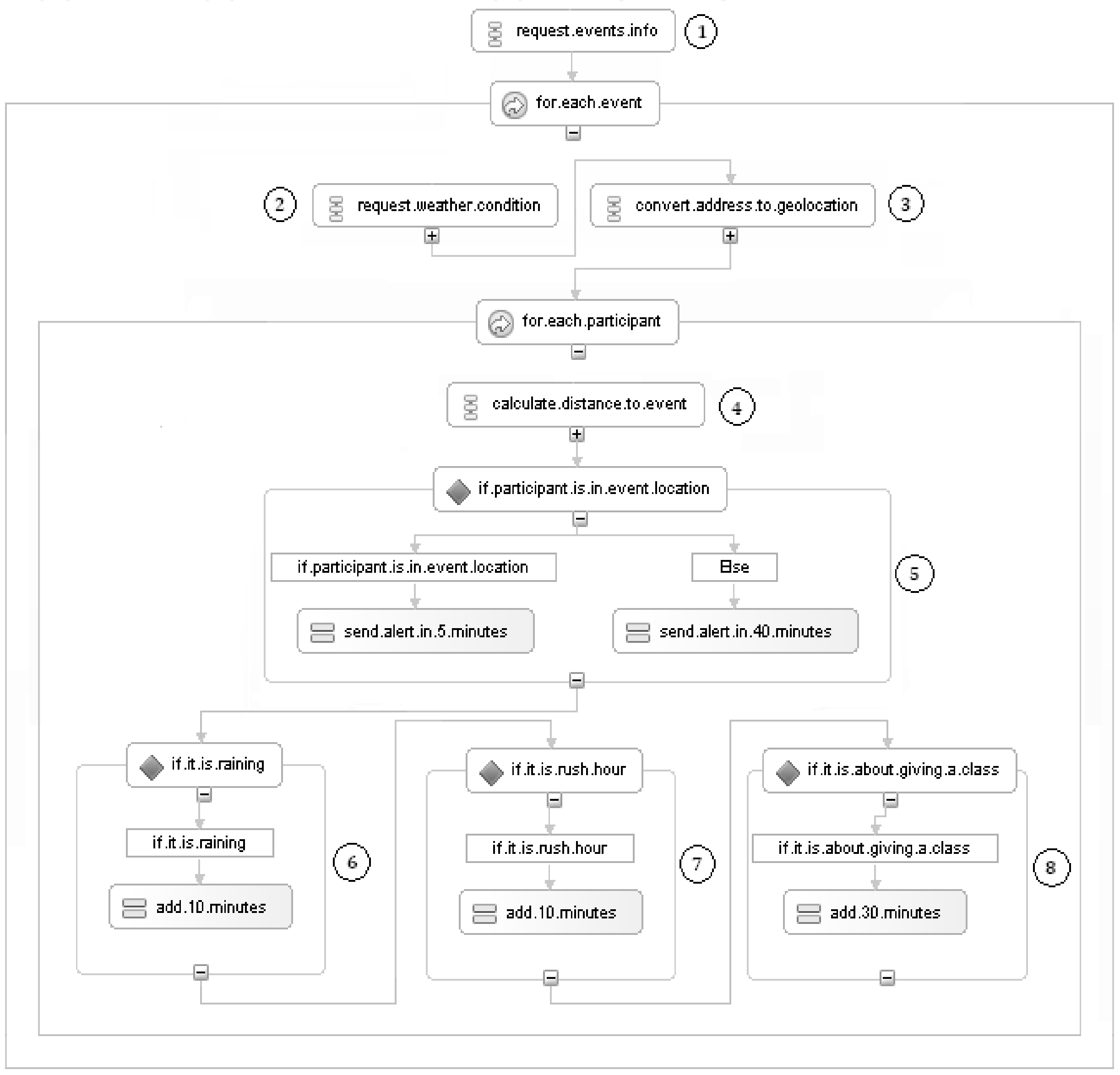


Fig. 9. Context adaptation based on WSBPEL (Carvalho & Silva, 2011).

After the modeling of the context adaptation, the BPEL designer is able to generate a WSDL document that externalizes the new generated composite web service to client applications. Figure 10 shows an example of the result of the adaptation supplied by the meeting alert web service.

[2] The description of the event is searched for the word "class" and similars.

```
To participant identified by 'luispsc@yahoo.com.br' the alert message must be sent
within 50 minutes!
To participant identified by 'paulocaetano.dasilva@gmail.com' the alert message
must be sent within 15 minutes!
```

Fig. 10. Adaptation supplied by the meeting alert web service (Carvalho & Silva, 2011)

4.2 Coupling CCMF and Ontologies

The O-CCMF can, as well, be used to develop the meeting alert application. In this section, its activities are performed to re-implement the software.

The first activity of O-CCMF is that of finding the appropriate ontologies to enable the modeling of context information. For instance, in relation to the meeting alert application, SOUPA, Standard Ontology for Ubiquitous and Pervasive Applications, (Chen et al. 2004) can be used for this purpose. SOUPA comprises two sub-ontologies, SOUPA Core and SOUPA Extension, which contain, among others, classes suitable for the representation of meetings. In Figure 11, such classes are highlighted (in yellow).

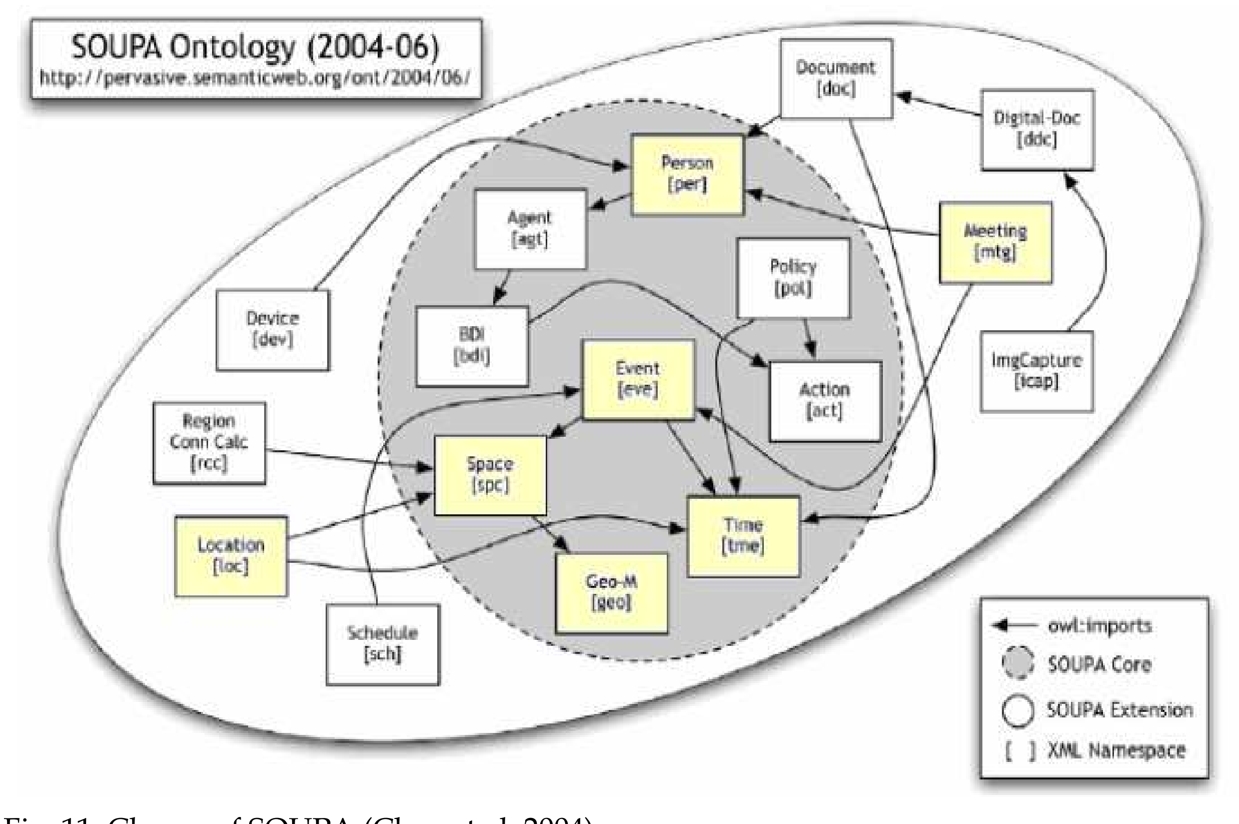


Fig. 11. Classes of SOUPA (Chen et al. 2004).

Considering the set of context information contained in Figure 5, SOUPA must be coupled with other ontology in order to represent weather conditions, since the adaptation requires the evaluation of such information prior to furnishing context adaptation. With that purpose, the ***O****WEATHER*, Weather Ontology, (Gajderowicz, 2008) is bound to SOUPA to provide classes which enable the modeling of weather-related information. Figure 12 shows the three layers of abstraction of the ***O**WEATHER* ontology. The upper layer, Class Level 1, contains a top generic Weather class under which grouping classes (e.g. Wind, Precipitation at Class Level 2) are defined. Class Level 3 contains classes that represent specific natural phenomena (e.g Gusting, Rain).

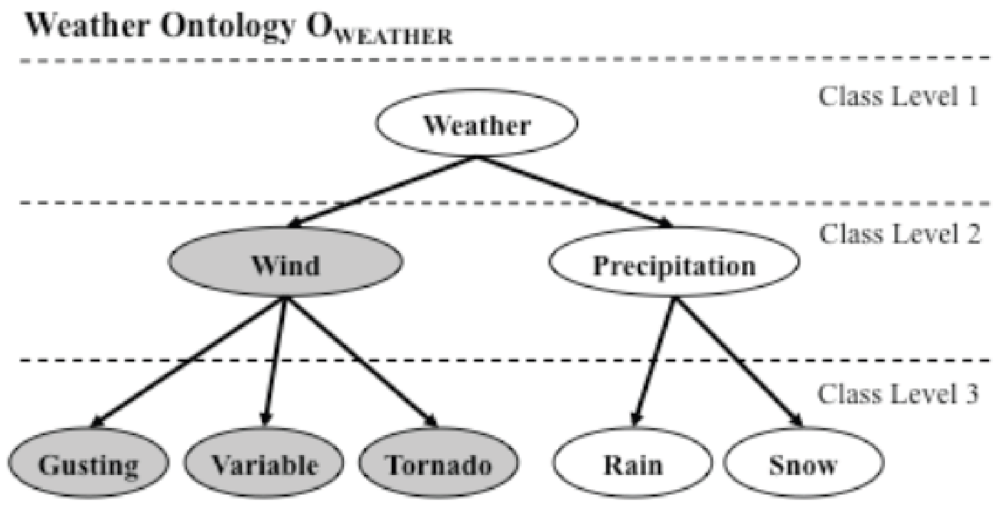


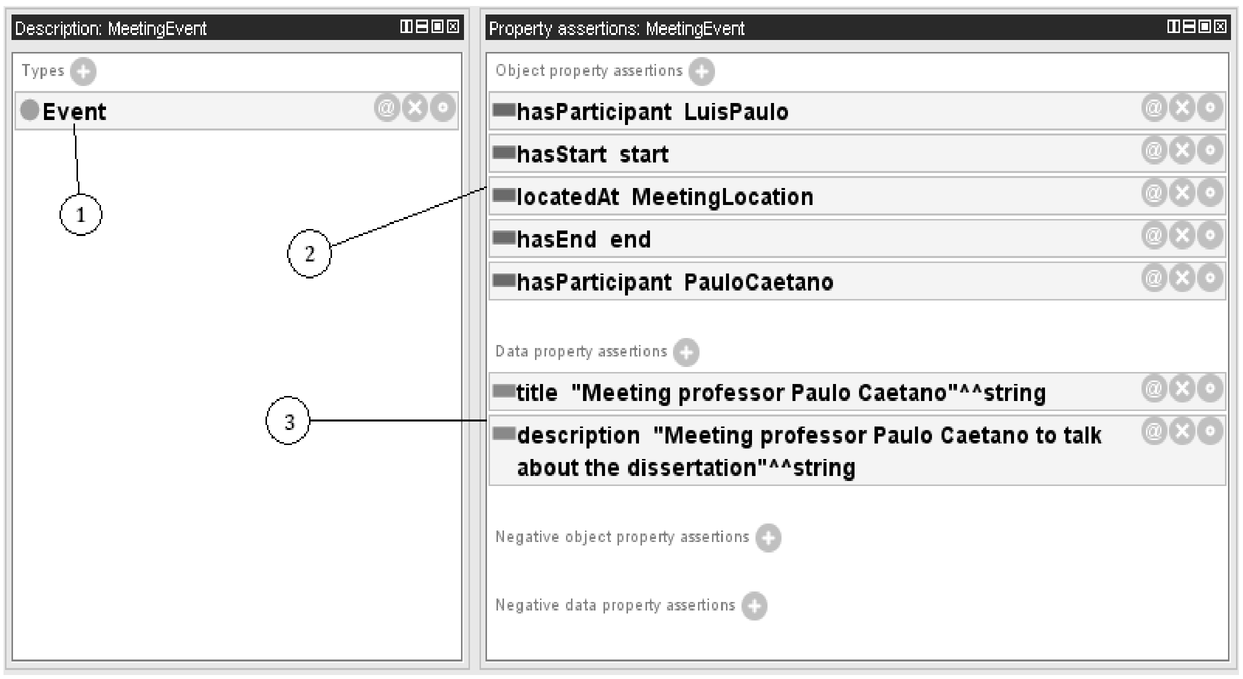
Fig. 12. Classes of ***O**WEATHER* (Gajderowicz, 2008).

The union of SOUPA and ***O**WEATHER* produces a new ontology, MAO (Meeting Alert Ontology), which enables the modeling of the context structures for the meeting alert application. Table 4 relates the classes contained in MAO to the context information defined using CCMD (in Figure 5).

Source ontology	Ontology classes	CCMD classes
SOUPA	Meeting + Event (MeetingEvent)	Meeting
	Location	Location
	Time	Temporal
	Geo-M	Geolocation
	Person	Participant
OWEATHER	Weather	Weather

Table 4. Ontology classes to represent context information.

The resulting ontology can be populated by information related to meetings. In Figure 13, number 1 points to a "MeetingEvent" individual (instance of the "MeetingEvent" class of MAO) being exhibited by Protégé (Protégé, 2011). Protégé is a free, open-source platform that provides a suite of tools to construct domain models and knowledge-based applications with ontologies. Number 2 identified the class-to-class properties of the event, i.e. its relation with other complex classes of the ontology. The "hasParticipant" element represents an association between an event and its participants, i.e. the individuals "LuisPaulo" and "PauloCaetano" that are instances of the class "Person". The "hasStart" and "hasEnd" elements are related, respectively, to the starting and ending date and time of the event (they are instances of the "Time" class). The "locatedAt" element is intended to represent the spatial location of the event, i.e. the place where the meeting is intended to happen. The ontology also supplies information about the meeting's title and description (elements pointed by number 3).


Description: MeetingEvent

Types

Event

Property assertions: MeetingEvent

Object property assertions

hasParticipant LuisPaulo
hasStart start
locatedAt MeetingLocation
hasEnd end
hasParticipant PauloCaetano

Data property assertions

title "Meeting professor Paulo Caetano"^^string
description "Meeting professor Paulo Caetano to talk about the dissertation"^^string

Negative object property assertions

Negative data property assertions

Fig. 13. The "Event" individual of the meeting alert ontology.

Although, it is possible to combine different ontologies in the production of a new one, e.g. the combination of SOUPA and *OWEATHER* produces MAO, it is necessary to adapt the ontology to better express the context information. For instance, Event and Meeting were two dissociated concepts of SOUPA, but, considering the meeting alert application, they had to be combined in a single class, MeetingEvent, so that it would supply a unique way to represent meeting-related events. This adaptation exemplifies the execution of activity 2 of the O-CCMF (as illustrated in Figure 4).

The meeting event information of Figure 13 was retrieved from a web service based on the Google Agenda API and stored in MAO using the OWL API (OWLAPI, 2011). The composite web service (the one illustrated in Figure 9 of Section 4.1) was altered with the intention of using the OWL API and ontologies (SOUPA and *OWEATHER* grouped in MAO) as a replacement for the XML serializable classes (i.e. the JAVA XML Beans classes originated from XSD documents). In this case, the ontologies promote the sharing of a common vocabulary that replaces the serialization of context information via XML documents. As a concrete example of this form of interoperability, Figure 14 illustrates how individuals retrieved from Google Agenda are added to MAO with the intention to parameterize the execution of other web services, e.g. the one originated from the Google Geocoding API. This is done by the "getEventEntries" method that selects event entries found in user's agenda and inserts them in the meeting alert ontology. Once the Geocoding converts the address of the meeting to geographic coordinates, they are compared to the current location of the participants to determine how far/near they are from meeting's location. The location is also used by Yahoo Weather Forecast to evaluate the rain likelihood, i.e. the service analyzes the weather-related information to determine if it is going to rain in the area where the meeting is intended to happen.

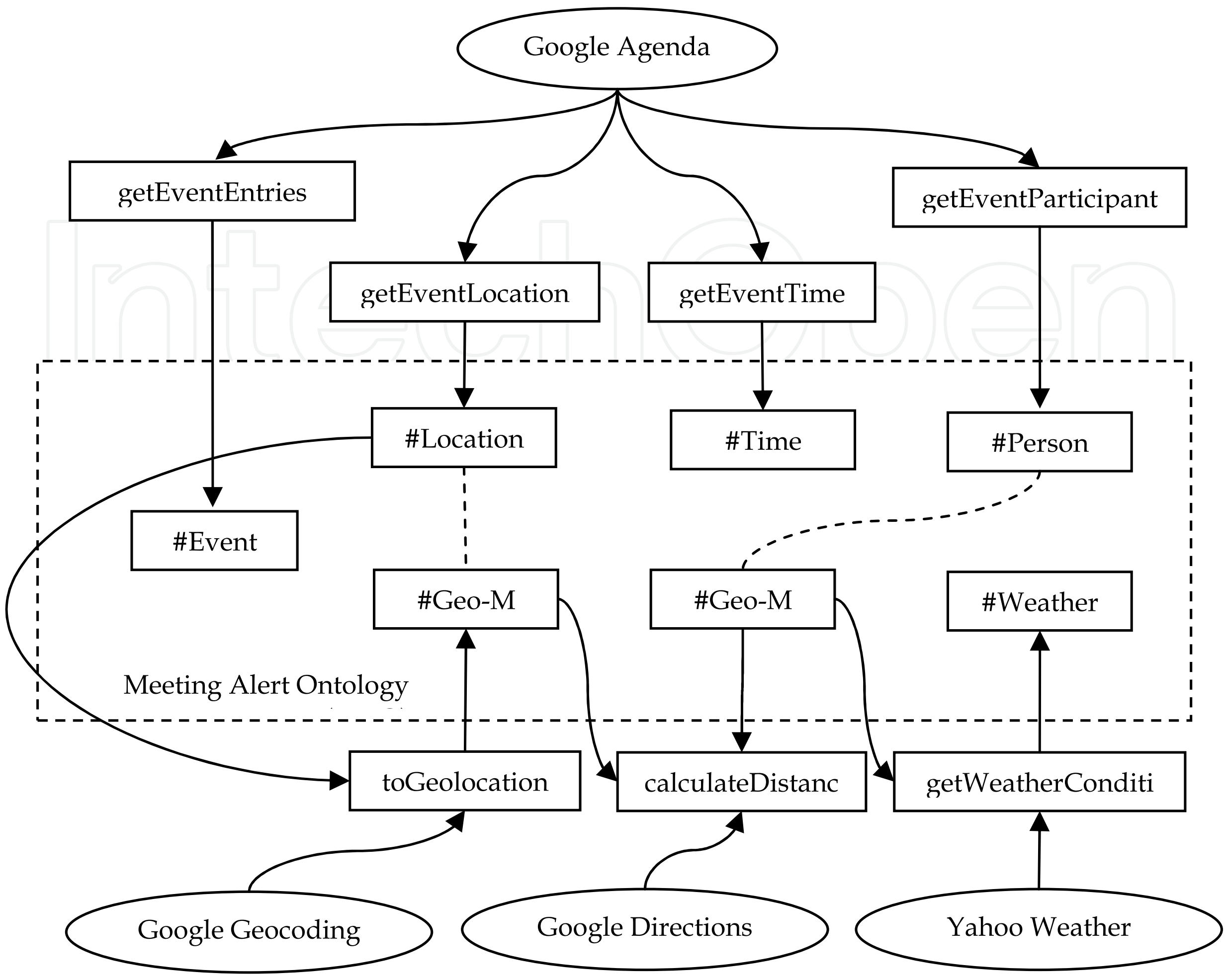


Fig. 14. Using the MAO ontology to interoperate web services.

Considering the scenario depicted in Figure 14, the context adaptation is less influenced by the overhead of modeling serializations via XML. In the making of the meeting alert web application, for instance, it was possible to produce web services which took no parameters as input and, likewise, returned no output, because the context information was selected and saved directly to the ontology (no serialization of context information needed). As a consequence, the modeling of the context adaptation can rather rely on choreographed web services in substitution to orchestrations. Figure 15 illustrates the difference between orchestration and choreography (Gábor et al. 2004). The orchestration (left side of Figure 15) requires that the execution of web services is controlled by one agent which describes how services interact with each other. The WSBPEL, for instance, is an orchestrator of web services. In a choreography (right side of Figure 15), each web service involved in the process describes the part they play in the interaction which is performed in a collaborative manner.

The choreography can be exemplified by the interaction between the Geocoding and Directions web services. The Directions web service is able to extract the geographic coordinates inserted into the ontology by the Geocoding web service. This is performed in a i.e. choreographed manner, since the direct manipulation of the meeting alert ontology by

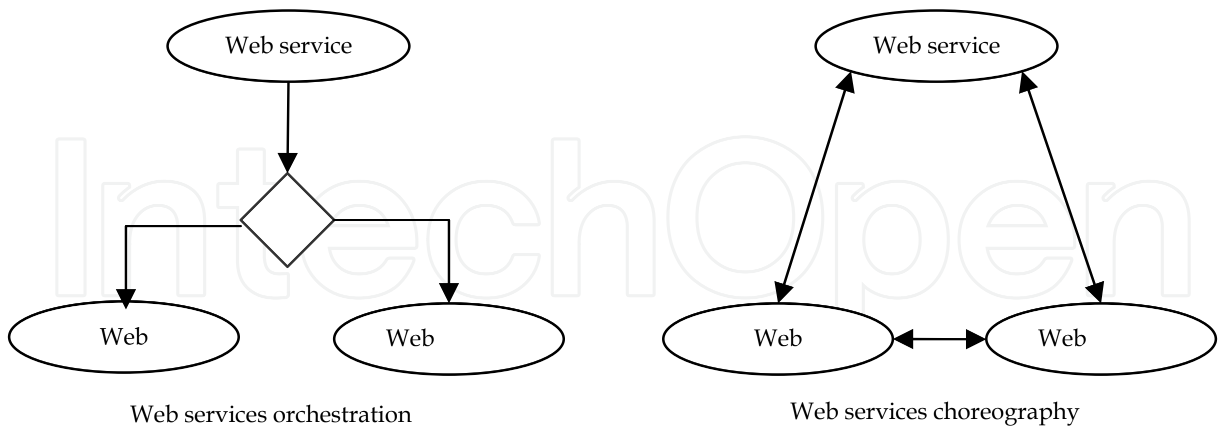


Fig. 15. Choreography and orchestration of web services (Peltz, 2003).

the web services promotes an internalized sharing of context information, without the intervention of controlling agents, such as an orchestration mechanism. As an opposite example, the WSBPEL diagram (e.g. the one illustrated in Figure 9, Section 4.1) is responsible for transferring the context information from one service to other, i.e. using WSBPEL, software designers must model an appropriate sequence of actions so that the context information is interchanged among services. In this case, the WSBPEL workflow acts like a controlling agent.

4.3 Comparing CCMF and O-CCMF

Provided the two scenarios of development of context-aware web applications, the ontological approach, O-CCMF, leads on a reduced framework in comparison with the former version, CCMF, because ontologies have the characteristic of being artifacts suited for both the storage and sharing of information among computational agents. The meeting alert ontology, for instance, represents structures of context information and, too, it encloses the instances of its own ontological classes (individuals). On the contrary, considering that CCMF utilizes CCMD to only model the context information with no regard as to afford the mechanisms to allow the direct storage and instantiation of its classes as concrete objects, it is required from CCMF the transformation of CCMD into XSD/XML documents to support the sharing of context information. Consequently, as computational agents are able to add and select individuals directly from ontologies, the context adaptation can be automated by a workflow that groups and interoperates web services in a collaborative manner, i.e. the adaptation is served to client software by choreographed web services as a replacement to orchestrated ones, in a way that the orchestration eases the modeling of the adaptation by not requiring the provision of serializations.

Table 5 relates the requirements identified in Section 2 (Table 1) to CCMF and O-CCMF, indicating how they were fulfilled by each framework.

Requirement	CCMF	O-CCMF
It must support the development of context-aware software as a two-phase task: specification of context information (structure) and adaptation	CCMD is coupled to the framework to represent context structures. WSBPEL diagram models the adaptation	CCMD is replaced by combined ontologies to represent context information. WSBPEL is maintained to create the adaptation
It must categorize the information into context dimensions	CCMD contains stereotypes that represent the context focus and dimensions	No specific data structures represent the focus and dimensions. It could be, though, added to ontologies (e.g. MAO) during the execution of the customization activity (activity 2 of the framework)
It has to identify the context focus of a task		
It must support the transfer of context information and artifacts between development phases	The transference is performed by XML documents and XML serialization API's	The usage of ontology is twofold: it represents the context and it enables the interoperability of web services by instantiated individuals
It must promote the diversity of context information in a domain-independent manner	CCMD does not constrain the domain	Although ontologies are known for representing specific knowledge domains, they can be combined to support broader concepts
It has to support the reuse of distributed computing systems such as services	Achieved by the usage of orchestrated web services	Achieved by the usage of choreographed web services

Table 5. Evaluating the frameworks against the requirements.

CCMF and O-CCMF are capable of assisting developers in the creation of context-aware web applications. Adopting one or the other as development framework is a matter of deciding what it is the intended context information source: diagrams such as CCMD or ontologies. Another criterion would be the intended method of creating composite web services to serve context adaptation to client software: orchestrated (CCMF) or choreographed (O-CCMF) web services.

5. Conclusion and future work

Related works, as those described in Section 2, are likely to subject the modeling and development of context-aware applications to the utilization of ontologies or diagrams-oriented solutions (e.g. stereotyped UML class diagram). By introducing CCMF, a model-driven framework, and its ontology-oriented variation, O-CCMF, this work points to a heterogeneous scenario where context information is collected from different sources and adaptations are served by varied web services. Such diversification demands adaptive approaches from development solutions, as, for instance, the ability to deal with different modeling technologies (e.g. ontologies, diagrams) in a decoupled manner. Instead of favoring one specific form of development of context-aware web applications, CCMF and O-CCMF promote the reuse of a mixed set of artifacts, tools, API's, information and functionalities sources.

In regard to CCMF, its coupling with CCMD intends to grant to developers an immersive environment in which concepts of computational context (context focus and dimensions) orients the design of context information and adaptation. An advantage that is not provided by O-CCMF. However, contrary to ontologies, CCMD (and CCMF, consequently) does not rely on a single artifact to represent classes and instances of context information. Thus, CCMF has to be coupled with extra mechanisms (e.g serializable JAVA classes) to workaround such limitation which is naturally overcame by O-CCMF.

Both frameworks rely on web services to automate context adaptation and to serve it to remote clients across the internet. One advantage that arises from this approach is that of making possible the addition of further context information and adaptation steps to enhance context-awareness mechanisms. For instance, improvements in the Google Agenda API can be automatically propagated to client software without needing to distribute modification patches. Conversely, faulty web services might lessen the quality of served adaptations. For example, in case a specific functionality of the Google Agenda API either becomes deprecated or fails to retrieve some important context information, client software may not succeed in supplying context-awareness to end users.

Currently, the following tasks must be carried out in order to supplement this work:

1. The ontology-driven framework must be evaluated against further study cases to analyze weather the WSBPEL diagram can be decoupled from the framework in order to favor other composition mechanisms, e.g. the WSCI, Web Service Composition Interface (Gábor et al. 2004), which is representative of the choreography approach;
2. Reasoning mechanisms based on the semantics supplied by ontologies must be surveyed in order to enhance the adaptation. For instance, the conditional test that evaluate if the participant is the same location of the event could be inferred from the ontology by the web service that schedules the alerts;
3. The ontological approach promotes the binding of existent ontologies in the definition of domain-specific context information. Therefore, the searching for fitting available ontologies must be adequately supported by the framework. It must be surveyed how this activity can be better assisted by the coupling O-CCMF with available third-part tools and processes.

6. References

Abowd, G. D. & Mynatt. E. D. (2000). Charting past, present, and future research in ubiquitous computing. *ACM Trans. Comput.-Hum. Interact. 7*, pp. 29-58

Alboaie, L.; Buraga, S. C. & Alboaie, S. (2003). tuBiG: a layered infrastructure to provide support for Grid functionalities, *Proceedings of the second international conference on Parallel and distributed computing*, Washington, DC, USA, pp. 9-14

Antoniou, G.; Bikakis, A.; Karamolegou, A. & Papachristodoulou, N. (2007). A context-aware meeting alert using semantic web and rule technology. *International Journal of Metadata, Semantics and Ontologies 2007*, Vol. 2, No.3, pp. 147 – 156

Bastida, L.; Nieto, F. J. & Tola, R. (2008). Context-aware service composition: a methodology and a case study, *Proceedings of the 2nd international workshop on Systems development in SOA environments*, New York, USA, pp. 19-24

Bonino, L. O.; Wijnen, R. P. & Vink, P. (2007). A service-oriented middleware for context-aware applications. In *Proceedings of the fifth international workshop on middleware for pervasive and ad-hoc computing*. New York, NY, USA

Bontas, E. P.; Mochol, M. & Tolksdorf, R. (2005). Cases on Ontology reuse. *Proceedings of the 5th International Conference on Knowledge Management*

BPELEclipse (2010). BPEL Eclipse Designer. Available from http://www.eclipse.org/bpel

Brézillon, P. (1999). Context in problem solving: a survey. *Knowl. Eng. Rev. 14*, 1, pp. 47-80.

Brézillon, P. & Bazire, M. (2005). Understanding Context Before Using It. *5th International and Interdisciplinary Conference, CONTEXT-05*, v. LNAI 3554, Springer Verlag, Paris, France, pp. 29-40

Brickley, D. & Miller, L (2003). FOAF vocabulary specification. RDFWeb Namespace Document, RDFWeb, xmlns.com.

Bulcão, R. (2006). Um processo de software e um modelo ontológico para apoio ao desenvolvimento de aplicações sensíveis a contexto. Serviço de pós-graduação do ICMC-USP, São Carlos, São Paulo, Brasil

Carvalho, L. P. & Silva, P. C. (2011). CCMD – Computational Context Modeling Diagram – And WSBPEL Integration. *IADIS Applied Computing International Conference*. Rio de Janeiro, Brasil, November

Chen, H.; Perich, F.; Finin, T. & Joshi, A. (2004). SOUPA: Standard Ontology for Ubiquitous and Pervasive Applications. *1st Annual Int'l Conf. on Mobile and Ubiquitous Systems: Networking and Services*, August

Dey, A. K. (2001). Understanding and Using Context. *Personal and Ubiquitous Computing, Special issue on Situated Interaction and Ubiquitous Computing* 5, 1, pp. 4-7

Dey, A. K. & Abowd, G. D. (1999). Towards a better understanding of context and context-awareness, *Proceedings of the first international symposium on handheld and ubiquitous computing*, H. Gellersen, Ed. Lecture Notes in computer science. Springer-Verlag, London, England, pp. 304-307

Dey, A. K.; Abowd, G. D. & Salber, D. (2001). A conceptual framework and a toolkit for supporting the rapid prototyping of context-aware applications, *Human-computer interaction*, vol. 16

EMF (2009). Available from http://www.eclipse.org/modeling/emf

EuGENia (2011). Available from http://www.eclipse.org/gmt/epsilon/doc/eugenia/

Gábor, V.; Andersson, B. & Wohed, P. (2004). An Ontology based Analysis of BPEL4WS and WSCI. *Proceedings of the 3rd Nordic Conference on Web Services (NCWS 2004)*, ISBN 91-7636-431-3. Växjö, Sweden

GAgenda (2011). Google Agenda API. Available from http://code.google.com/intl/pt-BR/apis/calendar/

Gajderowicz, B. (2011). Using decision trees for inductively driven semantic integration and ontology matching. Thesis, Ryerson University, Program of Computer Science. Available from http://www.scs.ryerson.ca/~bgajdero/msc_thesis/document/b-gajderowicz-msc-thesis.pdf

GDirections (2011). Google Directions API. Available from http://code.google.com/intl/pt-BR/apis/maps/documentation/directions/

GGCoding (2011). Google GeoCoding API. Available from http://code.google.com/intl/pt-BR/apis/maps/documentation/geocoding/

GMF (2010). Graphical Modeling Framework (Graphical Modeling Project - GMP). Available from http://www.eclipse.org/modeling/gmp

George, A. A. & Ward, P. A. (2008). An architecture for providing context in WS-BPEL processes, *Proceedings of the 2008 conference of the center for advanced studies on collaborative research: meeting of minds*, New York, NY, USA, Article 22

Grassi, V. & Sindico, A. (2009). Model driven development of context aware software systems, *Proceedings of International Workshop on Context-Oriented Programming*, New York, USA, pp. 1-5

Hoyos, J. R.; Molina, J. G. & Botia, J. A. (2010). MLContext: A Context-Modeling Language for Context-Aware Systems, *Proceedings of Electronic Communications of the European Association of Software Science and Technology*

JBDrools (2009). JBoss Drools. Available from http://www.jboss.org/drools/drools-flow

Noy, N. F. (2004). Semantic integration: A survey of ontology-based approaches. *SIGMOD Record*, 33(4), Dec. 2004

OWLAPI (2011). The OWL API. Available from http://owlapi.sourceforge.net/index.html

Patrício, R. F. (2010). CEManTIKA CASE: uma ferramenta de apoio ao desenvolvimento de Sistemas Sensíveis ao Contexto, UFPE, Recife, Pernambuco, Brasil

Peltz, C. (2003). Web Service Orchestration and Choreography. Available from http://wpage.unina.it/rcanonic/didattica/at/documenti/wsOrchestration.pdf

Protégé (2011). Protégé Ontology Editor. Available from http://protege.stanford.edu/

Reinisch, C.; Granzer, W.; Fraus, F. & Kastner, W. (2008). Integration of Heterogeneous Building Automation Systems using Ontologies. *Proc. of the 34th Annual Conference of the IEEE Industrial Electronics Society (IECON '08)*, Nov. 2008, pp. 2736–2741

Schmidt, D. C. (2006). Guest Editor's Introduction: Model-Driven Engineering. IEE Computer 39(2), pp. 25-31

Sheng, Q. Z. & Benatallah, B. (2005). ContextUML: A UML-based modeling language for model-driven development of context-aware web services, *Proceedings of the International Conference on Mobile Business*, Washington, DC, USA, pp. 206-212

Simons, C. 2007. CMP: A UML context modeling profile for mobile distributed systems, *Proceedings of the 40th Annual Hawaii International Conference on System Sciences*, Hawaii, USA

Topcu, F. (2011). Context modeling and reasoning. Available from http://www.snet.tu-berlin.de/fileadmin/fg220/courses/SS11/snet-project/context-modeling-and-reasoning_topcu.pdf

Vieira, V. 2008. CEManTIKA: A domain-independent framework for designing context-sensitive systems, Centro de Informática - UFPE, Recife, Pernambuco, Brasil

XMLBeans (2009). Apache XML Beans 2.5. Available from http://xmlbeans.apache.org/

XSD (2001). W3C XML Schema 1.1. Available from http://www.w3.org/XML/Schema

Yamato, Y.; Nakano, Y. & Sunaga, H. (2010). *Context-aware service composition and change-over using BPEL engine and semantic web*. Intech Open Access Publisher. Rijeka, Croatia

YWForecast (2011). Yahoo Weather. Available from http://developer.yahoo.com/weather/

Wang, X. H.; Gu, T.; Zhang, D. Q. & Pung, H. K. (2004). Ontology based context modeling and reasoning using OWL, *Proceedings of the 2004 communication networks and distributed systems modeling and simulation conference*, San Diego, CA, USA

WSBPEL (2007). OASIS Web Service Business Process Execution Language (WSBPEL) 2.0. Available from http://www.oasis-open.org/committees/wsbpel/