First Approximation to DHD Design and Implementation

Alejandro R. Sartorio
CIFASIS (Centro Internacional Franco Argentino de Ciencias de la Información y de Sistemas), CONICET (Consejo Nacional de Investigación Científica y Tecnológica,UNR (Universidad Nacional de Rosario)
Rosario, Argentina,
sartorio@cifasis-conicet.gov.ar

and

Maximiliano Cristiá
CIFASIS (Centro Internacional Franco Argentino de Ciencias de la Información y de Sistemas), UNR (Universidad Nacional de Rosario)
Rosario, Argentina,
cristia@cifasis-conicet.gov.ar

Abstract

This work presents a proposal on design and implementation of web sites e-learning with properties of service reconfiguration in execution time, offering a new adaptation and personalisation mechanism with context-aware characteristics.
We describe an integration model to incorporate "the coordination context-aware contracts" into an e-learning framework. The implementation has been solved using design patterns which allow superimposition of rules through contracts, by means of the interaction between the objects that implement services (i.e messages edition on a Forum) and the objects that require them.

Keywords: Dynamic Hypermedia Device, Contract, E-learning.

1. Introduction

The advance in researcher and development of e-learning platforms offer improvement and innovation tools (such as video conference, portfolios, wikis, workshops, etc.) and their corresponding services. As a result, there is a growth in the amount of possible e-learning sites configurations. These configurations cover different types of requirements that correspond to the different design and development stages and they even require the adaptation of the e-learning site in terms of execution time. From these requirements, e-learning processes are defined (called Pe-lrn) [4] in a similar way as business processes but on another application domain.
Like business processes in a conventional Web Application, Pe-lrn is formed by web transactions [4]. In this context, a transaction (or e-learning transaction) is defined as a sequence of activities that a user executes through an e-learning Application with the intention of performing a task or reaching a goal, where the set of activities, their properties and the rules that control their execution, depend on the Pe-lrn by the Application. An example of didactic strategy is a student's possibility of accessing to certain material (videos, files, etc.) depending on his participation in Forums. These types of requirements are difficult to implement with the current e-learning applications of extended use at global level.
In the framework of the analysed items, we can state that the Sakai project www.sakaiproject.org offers one of the most consolidated proposals on design and development of collaborative e-learning environments for education, providing tools that

can be implemented through common services (base services). For example, the message edition service is used with tools like Forum, Announcement, Blog, Portfolio, etc. Moreover, another outstanding characteristic of Sakai is its versatility for extension and/or configuration. Indeed, Sakai allows to alter certain configurations in execution time, for example, arranging a new function in a Sakai base service. However, these solutions cannot solve those Pe-lrn which involve changes in the behaviour and relationship between a component (client) that requests the execution of a server component (supplier).
These types of changes refer to the system dynamic adaptation capacity [6] to extending, personalising and improving the services without need of compiling and/or restarting them. In this paper, we present a proposal for the incorporation (aggregate) of dynamic adaptation properties to the base services included in the Sakai framework. They have been especially designed to implement Pe-lrn, where new adaptation [13] aspects are required in perspective of Dynamic Hypermedial Devices in the e-learning field with context-aware characteristics [1,12]. We understand by DHD "a social net mediated by ICT (Information and Communication Technologies) in a physical-virtual context where the subjects investigate, teach, learn, talk, disagree, test, produce and carry out transformation processes on objects in a very responsible way. These are regulated, according to the case, by a coordination of contracts integrated into the participant Workshop method". (San Martín et al, 2008).

2. Context-aware Contracts

Our proposal for satisfying the mentioned requirements about dynamic adaptation starts with the construction of a contract model oriented to the implementation of context-aware services. The use of contracts comes from the notion of Programming by Contract by Meyer [5] based on the metaphor that an element from a software system collaborates with other maintaining mutual obligations and benefits. In our application domain we will consider that an object-client and an object-server "agree" by means of a contract (represented with anew object) that the object-server will satisfy the client's order, and at the same time, the client will fulfil the conditions imposed by the supplier. As an example of application of Meyer's idea in our domain of e-learning system, we outline a situation in which a user (client) makes use of a message edition service (server) by means of a contract that will guarantee the following conditions: the user should be able to edit those messages for which he has authorization according to his profile (supplier's obligation and client's benefit); the supplier should have access to the information of the user's profile (client's obligation and supplier's benefit).
From the conceptualisation of contracts according to Meyer, we propose an extension by adding new components to arrange mechanisms which allow the execution of actions depending on the context.
In context-aware applications [3], the context (or context information) is defined as the information that may be used to characterise the situation of an entity no matter the qualities that may define it. In our case, an entity is a user (student, teacher, etc.), a place (classroom, library, consultation room, etc.), a resource (printer, fax machine, etc.), or an object (exam, practical work, etc.) which communicates with another entity by means of a contract. In [1] we propose a specification of the concept of context from Dourish's considerations [9] and adapted to the e-learning domain that we will consider in this paper.
Context is all type of information that can be registered and processed through the e-learning application and that characterises a user or environment, for example: participation on a forum, average marks, abilities, levels of knowledge, connected machines (IP addresses), participation level on forums, amount of connected users, dates and times, courses statistics, etc. In general terms, the coordination of contracts is a connection established among a group of objects, even though in this paper only two objects have been considered: a client and a server. When an object client calls an object server (for example the editing service of the Forum tool), the contract "intercepts" the call and establishes a new relationship taking into account the object client context, the object server context and the relevant information acquired and represented as environment context [9].
Next, we will bring details about some of the components and essential relationships for the integration of this model into the Sakai framework and the methods that arrange contracts coordination.

2.1 Elements of context-aware contracts

A contract following Meyer's ideas contains all the information about the services that clients will use. To incorporate context awareness, our contract should have references about some type of context information for its use. In the diagram showing the relationship among entities 1, we describe the elements that compose the concept of context-aware contract, especially for Pe-lrn.

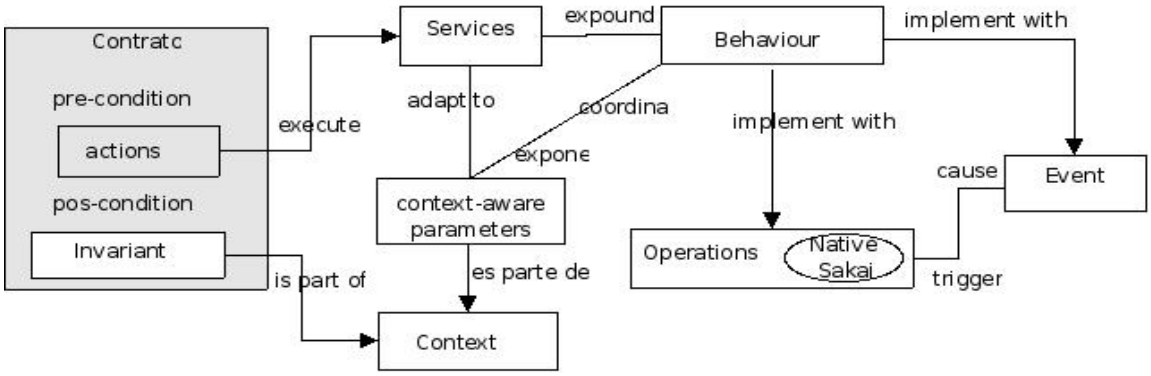

Figure 1: Model of context-aware contract

The figure begins with the representation of a contract according to Meyer's idea, where the main component elements are shown (pre-conditions, actions, post-conditions). The arrows leaving from the grey area point out two types of relationships (action-service and invariable-context) that should be arranged to incorporate a mechanism which can give the contracts context-aware characteristics. On the right side of Figure 1 we can see the entities needed to obtain context-aware contracts. They are explained below:

Services: In this component we represent the necessary elements to identify a service, and a classification of services that can be part of the contracts actions. For example, service name, identifiers, reaching, purpose, etc. [1]. The functional behaviour of each services is expounded through **Behaviour** component.

Behaviour: A service behaviour is obtained combining operations and events which are represented by the component **Operations** and **Events**.
In the same way the service may be implemented using events, represented by the component **Events**, that can zoom out operations from the **Operations** component. For example, according to the roles (ie, student, instructor, teacher, etc.) assigned to a tool user involved in a Pe-lrn, in a determined surrounding context (for example, a Forum context), and according to the user (for example if the person has moderator permission), the **service** component offers different functions (for example, editing a message) which are arranged by means of concrete actions (for example, saving a message on a table) and/or through the publication or subscription of events.

Context-Aware parameters: Context-aware parameters, are the representation of context information that is part of the entry parameters of the functions and the methods exported by services, establishing in this way a relationship between the **Service component** and the **Context-Aware Parameters** and **Behaviour**.

Context: In our model this type of information may be used in two different ways: firstly, for assignation of values taken by context-aware parameters; secondly, this information can be used to define the invariability represented in the contracts.

3. Integration model: Sakai and contracts

In this section, we present a sketch with the proposal to incorporate Sakai a coordination mechanism for context-aware contracts, by means of a presentation of a design which

describes the communication among modules [7] and their dependences. In figure 2, the modules are represented by UML packets and the relationship between them is represented by means of arrows indicating communication and dependences. At the same time, in the background, we show which the specific classes in each module are and how to implement the integration through them.
The Sakai framework, represented in Figure 2 by the Sakai module, has been designed according to a four-layer architecture [11]: the aggregation layer is completely in charge of the implementation of the interface with the user, in the same way as Web sites implementation. The second layer called presentation is responsible for allowing the Web components to be reused (for example, “widget” which provides calendars, “WSYWIG” editors, etc.). The third one belongs to a tools layer where we find the business logic of Sakai tools that interacts with the user (for example Forum, Wiki, etc.). Finally, the service layer implements Sakai services under a Service Oriented Architecture which will be used by various tools through API (Application Programming Interface).
For our interaction proposal , we will only take into account those services which compose the service layer belonging to Sakai base services. Sakai nucleus services will be ”wrapping” by means of contracts coordination, what is represented in Figure 2 with reference to the **Sakai Module** towards the **Contract Coordination Module**.
In this way, we are able to control the invocation of the Sakai nucleus services by means of the **Contract Coordination** module, establishing a first integration relationship. The same relationship is implemented at classes level as an aggregation relationship between the Sakai Service class (belonging to the Sakai module) and the Connector class belonging to the coordination framework that will be discussed in Section 4 through the use of design patterns.

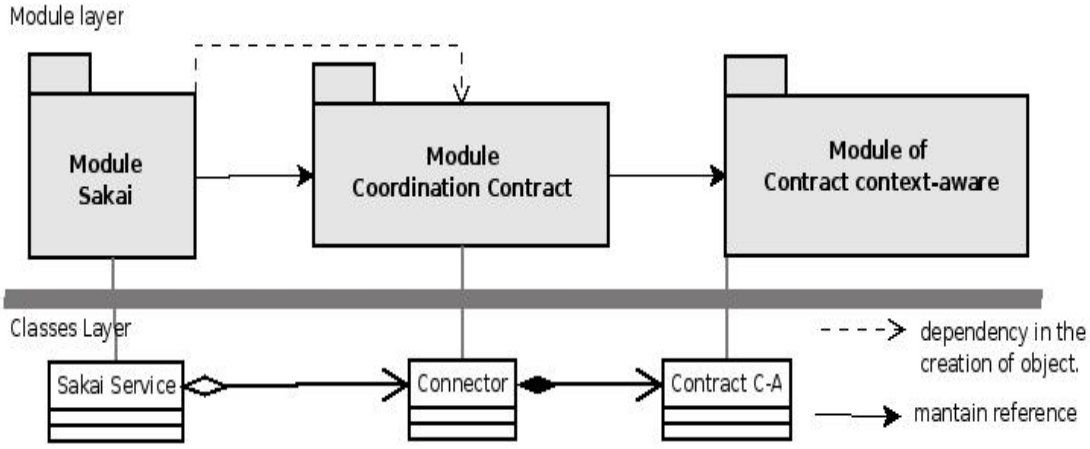

Figure 2: Design of interaction of Sakai with contract

The last integration relationship is produced between the **Contract Coordination** module and the **Context-Aware Contract** module which has been explained in Section 2. In this case, Figure 2 shows how from the coordination framework, a composed relationship between the Connector class and the C-A Contract class that implements the Contract element represented in grey in Figure 1.
In this way, instancing different contracts in a dynamic way, Sakai will present flexibility in execution time (currently lacked and required by DHD).

4. Implementation of contracts in Sakai

In the above sections the contract has been mentioned as an active agent in charge of coordinating its constituent components which form it. In this section, we describe how this mechanism has been implemented by means of a tool which allows the modification of Java type implementations of Sakai services to incorporate mechanisms of context-aware contracts coordination, according to the design shown in Figure 2. However, we believe that first it is convenient to summarise briefly other attempts to provide the flexibility required by DHD.

4.1 Sakai flexibility levels

Different standards and frameworks of software development based on components and service injection have been proposed to improve the flexibility and adaptation[1] of e-learning Web systems. The most important are: CORBA, JavaBeans, Hibernate, Spring, JSF, RSF, etc. However, none of these standards provides an abstract mechanism suitable to carry out the representation of the relationship between users and services as a first class object [5]. This idea is related to computing solutions where the potentiality of a language is reinforced by implementing a service belonging to the services layer instead of the presentation layer. The first antecedents of this type of solutions were used for communication systems applied in the datagram layers for a change in protocols by means of rules [8].

In this sense, we propose a solution using design patterns which can implement the characteristics required by TCCs-c over the base service layer of Sakai framework.

From the point of view of implementation, this proposal changes Java2EE-Sakai[2] philosophy where the incorporation of new versatile services for configuration and implementation are achieved through class extensions, reflection techniques, etc. The following UML class diagram shows the design proposed to incorporate Sakai coordination of context-aware contracts.

4.2 Design patterns for coordination of context-aware contracts

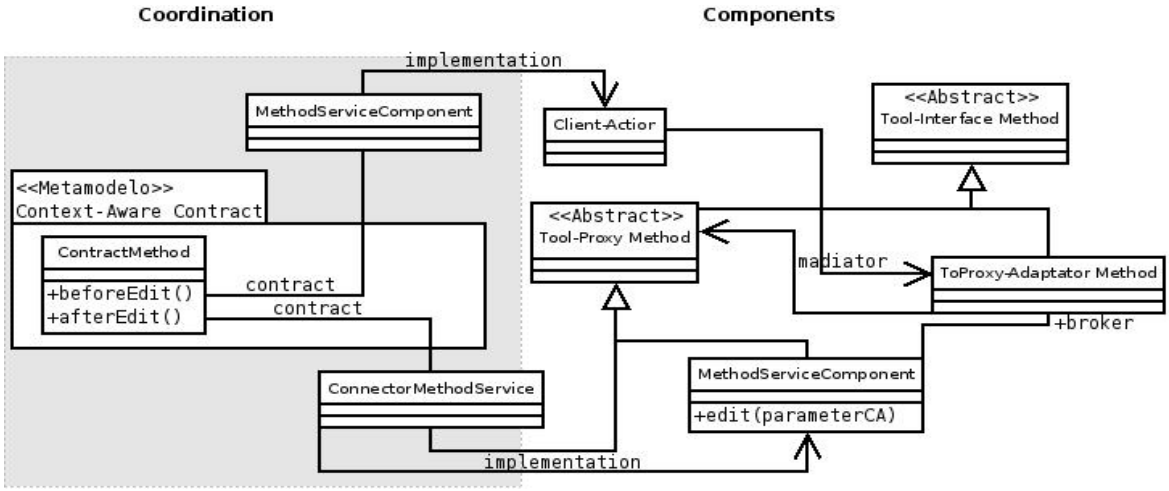

Figure 3: Design pattern to implement coordination contract in Sakai

Tool-Interface Method: This is an abstract class which defines a common interface of services from a Tool provided by **Tool-Proxy Method** and **ToProxy-Adaptator Method**.

ToProxy-Adaptator Method: This is a concrete class which implements an intermediary in connection with the Proxy-Method class to take charge of the received requests. In execution time, this entity can indicate **MethodServiceComponent** that there are not contracts involved or it can arrange a connection between **ConnectorMethod Service** to connect the real class **MethodServiceComponent** with the contract **ContractMethod** in which you can find the coordination rules.

Tool-Proxy Method: This is an abstract class which defines the interface that **MethodToolToProxy-Adaptator** and **ConnectorMethodService** have in common. The interface comes from **MethodTool-Interface** in order to guarantee that it offers the same interface of **MethodToolToProxy-Adaptator** with which a real client should interact (for example, an object which invokes the Forum edition service).

1 Adaptation in the sense proposed by hypermedial systems which allow the personalisation (adaptation) according to individual users (Henze, 2000)

2 Sakai keeps the development philosophy of Java2, service-oriented and which may be reliable, inter operational and extendable.

EditServiceComponent: This is the concrete class where we find the Forum edition logic and the concrete implementation of Sakai original services.

ConnectorMethodService: It connects the contract with the real objects (in this case **MethodService-Component**) taking part in the contract. In this way, is not required the creation or instantiation of new objects to coordinate the same real object by adding or removing other contracts. This is only possible with a new association to a new contract and with the instantiation of a connection with a **ConnectorMethodService** existing instance. This means there is only one instance of this class associated to a **MethodServiceComponent** instance.

ContractMethod: This is the class whose instance generates a coordination object that, when notified, can make decisions when a request is invoked through a real object. In case there is no contract coordinating a real object, the design on Figure 3 may be simplified; and only the classes and relationships that do not pass the grey rectangle area will be necessary. The class that implements the contract, called **ContractMethod**, is represented inside the figure of an UML packet called **Context-Aware Contracts** indicating that it belongs to an underlying model described in Section 2.
According to this explanation, we think it will be useful to reinforce our argument about advantages of this proposal comparing it with the current technological solutions based on components and frameworks for service injection and representation. At the same time, it can be observed that the way to its implementation is ward due to the required generation (automatically) of code portions and the creation of new Java file from the original ones on the Sakai platform that should be compiled and put into operation. It is necessary to highlight that Sakai compilation and starting up is performed only once. Any kind of change and configuration will be introduced through context-aware contracts.

5. Study case for contract implementation

In this section, we will present a case to illustrate the different stages that should be fulfilled to obtain the inclusion of contracts in Sakai under DHD perspectives, returning to the idea of including a contract in the edition service from Sakai Forum tool.
Carrying on with the study case from earlier stages, we show a Java code portion corresponding to the original Sakai source code where we propose a method called *editMessage* corresponding to the implementation of the edition service from the Forum tool.
The following fragment of Java source code implements an inheritance class *BaseDiscussionService* where the edition service interfaces corresponding to Sakai framework nucleus are located.

```
import org.sakaiproject.discussion.api.DiscussionMessage;
public class DiscussionService extends BaseDiscussionService{
public MessageEdit editMessage(MessageChannel channel, String id){
return (MessageEdit) super.editResource(channel, id);}
```

Then, through the SwC tool, XML files are created (similar to CED tool) to specify the contract rules which will involve services implied in the *editMessage* method (Stage1). After this, we execute the automatic code generation to create two different types of Java files (Stage2). The first one (file1) implements the functionalities which allow the connection with Proxy (for example *MethodTool-Proxy* in our study case); the second one (file2) implements the connector *ConnectorEditService* represented in Figure 3. Then, we show code fragments which allow a better illustration of the acquired characteristics from the modifications suffered by the source code from Sakai application by means of the tool.

Fragment 1 file 1. The packets corresponding to the contracts coordination framework (Figure 3) and the context-aware framework (Figure 1).

```
import cde.runtime.*; import obab.ca.*; // Framework context
public abstract class DiscussionService extends
BaseMessageService implements
DiscussionService,ContextObserver,EntityTransferrer,ForoInterface
```

Fragment 2 file 1. Methods aggregated by the tool to identify the classes that will be intercepted by the contract. *SetProxy* is in charge of the instantiation of Proxy components.

```
public static Class GetClassId() {return _classId;}
public CrdIProxy GetProxy() { return _proxy; }
public void SetProxy(Object p){if(p instanceof CrdIProxy && p instanceof
DiscussionInterface) _proxy = (CrdIProxy)p;}
public void SetProxy(CrdIProxy p) { _proxy = p;}
AccountInterface GetProxy_Account(){if ( _proxy == null ) return null;
return (DiscussionInterface) _proxy.GetProxy(_classId);}
```

Fragment 3 file 1. Aggregated method that implements the call of the client object to Proxy.

```
public long _getNumber(){new ComponentOperationEvent(this , "getNumber")_
.fireEvent(); return number;}
public messageEdit editMessage(MessageChannel channel,String id){
new ComponentOperationEvent(this,"Edit").fireEvent();
return (MessageEdit) super.editResource(channel, id);}
```

Fragment 1 file 2. Code fragment where the framework components are imported and the abstract classes from connectors and Proxy are inherited. The real object class *MethodService-Component,* according to Figure 3, is represented through the subject attribute.

```
import cde.runtime.*; import obab.ca.*;
public abstract class IDiscussionPartner extends
CrdContractPartner implements CrdIProxy, DiscussiontInterface {
protected Discussion subject;
```

Fragment 2 file 2. Definition of abstract methods to connect (*ConnectorMethodService,* represented in Figure 3) the contract with the service.

```
public void SetProxy(Object p) {subject.SetProxy(p);}
protected Object GetSubject_Object() {return subject;}
public void ResetProxy() { subject.SetProxy(null);}
```

Fragment 3 file 2. Methods which allow the access of methods which define services (for example, methods class *MethodService-Component*).

```
protected Discussion GetSubjectDiscussion(){return (Discussion) subject;}
protected IDiscussionPartner GetNextPartner_Discussion(){
return (IDiscussionPartner)GetNextPartner(Discussion.GetClassId());}
```

Fragment 4 file 2. Implementation (by fault), of methods defined in service interfaces. Through the *GetSubjectDiscussion*() methods, detailed above, we accede to methods created by the tool in the first file.

```
public void messageEdit (double amount,Customer c)_
throws DiscussionException { IDiscussionPartner
next = GetNextPartner_Discussion()
if (next != null) next.editMessage(amount,c);
else GetSubjectDiscussion()._editMessage(amount,c);}
```

Fragment 5 file 2. Implementation of conditionals of contracts rules which are located in XML files to be configured.

```
public CrdPartnerRules messageEdit_rules(string texto,Student c) throws
DiscussionException, CrdExFailure {return new CrdPartnerRules (this);}
```

The generation of an automatic code through the tool implies having certain knowledge of: Java language, aspects of implementation of the Sakai base framework and the frameworks which form it; and at the same time, it is important to have experience in compilation through Sakai Maven[3] and the modified-aggregated classes for contract use. This aspect indicates a possible limitation in applying the idea which has been proposed in this paper.

6. Conclusion

In this paper we have took into account in e-learning platforms about the adaptation flexibility degree as regards [10] and [6], that could be inferred by domain expert users (for example, teachers) to develop effective didactic strategies in order to reach planned pedagogical and investigation goals, and that the adaptability cannot be totally solved by intelligent adaptation systems (for example, agents, adaptation hypermedia, expert systems, etc.).

Based on the conceptual framework of Dynamic Hypermedial Devices for education and research and also on the current limitations, we have proposed an implementation of a contract coordination theory in a specific framework about collaborative Web e-learning systems (corresponding to Sakai project) offering a conceptual and technological evolving model.

The final objective of this developing proposal is to give effective technological solutions to cover the necessity of encouraging education and investigation processes with a responsible interaction among the subjects taking part in the DHD social technical net, implementing the solution in services that strongly depend on rules with volatile structures and whose conditionals are influenced by the context (i.e “Context identification” developed in chapter 5 [1] and keeping Dourish’s context outlines [9]).

References

[1] San Martín, P., Sartorio, A., Guarnieri, G. and Rodriguez, G. *Hacia un dispositivo hipermedial dinámico. Educación e Investigación para el campo audiovisual interactivo*. Universidad Nacional de Quilmes (UNQ). ISBN: 978-987-558-134-0. (2008)

[2] Sartorio, A. *Un modelo comprensivo para el diseño de procesos en una Aplicación E-Learning*. CACIC 2007. ISBN 978-950-656-109-3. (2007)

[3] Dey, A.K., Salber, D. and Abowd, G. *A Conceptual Framework and a Toolkit for Supporting the Rapid Prototyping of Context-Aware Applications*. Human-Computer Interaction (HCI) Journal, Vol. 16 (2-4), pp. 97-166. (2001)

[4] Brambilla, M., Ceri, S., Fraternali, P. and Manolescu I. *Process modeling in web applications*. ACM (TOSEM). (2006)

[5] Meyer, B. *Applying Design by Contract.* IEEE Computer, 40-51. (1992)

[6] Dowling J and Cahill, V. *Dynamic software evolution and the k-component model*. In: Proc. of the Workshop on Software Evolution, OOPSLA. (2001)

[7] Buschmann, F., Meunier, R., Rohnert, H., Sommerlad, and P., Stal, M. *Pattern- Oriented Software Architecture*. John Wiley. (1996)

[8] Koutsoukos, G., Gouveia, Andrade and Fiadeiro, L. *Managing evolution in Telecommunications Systems.* IFIP Working Conference on Distributed Applications and Interoperable Systems, Kluwer. (2001)

[9] Dourish, P. *What we talk about when we talk about context*. Personal and Ubiquitous Computing, vol. 8, No 1, Roma, 2004, pp. 19-30. (2004)

3 Proyecto maven: http://maven.apache.org/ref/2.0.4/maven-project/

[10] Houben, G. *Adaptation Control in Adaptive Hypermedia Systems*. in Adaptive Hypermedia Conference. AH2000. Trento, Italia. Agosto. LNCS, vol. 1892. Springer-Verlag, pp. 250-259. (2000)

[11] Severance, C. *Sakai Project Report: The Evolution of the Sakai Architecture*, in: ”http: //elearning.surf.nl/docs/sakainl/200607archhistoryv01.doc“ 25-05-2008

[12] Project “Software Engineering Techniques Applied to Dynamic Hypermedia Divices”. Dir.: Mg. Maximiliano Cristiá. (CIFASIS: CONICET, UNR, UPCAM) Res. C.S. UNR, Nº 945/2008

[13] Project “Obra Abierta: Dynamic Hypermedia Devices to teach and research” Dir.: PhD. Patricia San Martín. (CIFASIS: CONICET, UNR, UPCAM) Res. C.S. UNR, N° 945/2008