OpenMuseum: a framework for collaboration between cultural heritage institutions*

Mirtha L. Fernández Venero[1] , Cláudia J. Abrão de Araújo[1] , Diego Mira David[1] , Flávio Soares Corrêa da Silva[1] , Ana Gonçalves Magalhães[2]

[1]Department of Computer Science, University of São Paulo, Brazil 05508-090

[2]Museum of Contemporary Art, University of São Paulo, Brazil 05508-050

{mirtha, claudiaj, diegomd, fcs}@ime.usp.br, amagalhaes@usp.br

Abstract. *This work outlines current research issues concerning the implementation of a platform, called OpenMuseum, for collaboration and knowledge sharing between cultural heritage institutions. Most of the work developed for this domain is oriented towards the use of global models for integration. Contrastingly, OpenMuseum has been designed using a loosely coupled approach in which institutions collaborate through a lightweight interoperability platform, called JamSession, which effectively deals with the heterogeneity of service-oriented systems. In order to manage the complex nature of cultural heritage information, some extensions to JamSession must be implemented. The extended platform will serve for cooperative interactive systems in general.*

1. Introduction

Information is considered a key resource for supporting business processes crossing organizational boundaries. Due to its increasingly complex and changing nature, an accurate and cost effective management of information resources is still a difficult task when building inter-organizational information systems. A suitable approach is the development of system architectures focusing in information sharing environments. The approach for interoperability between organizations ensures the interaction between them and provides several benefits such as, for example, the use of their legacy platform and the interaction between their systems. However, this interoperability requires the definition of common concepts that standardize and guide the interactions between organizations. Through these common concepts, organizations can exchange information with each other and still maintain their independence and particularities in their internal systems. It is also crucial to define the workflows within each organization and integrate these workflows, to ensure collaboration between organizations. Therefore, we must consider the fact that organizations have very different workflows and therefore it is essential to use techniques that allow the interaction between these workflows (cross-organizational workflows). These interactions should be conducted in a flexible and reliable way to ensure that each organization has control over its processes.

This paper discusses some research issues concerning the implementation of a platform, called OpenMuseum, to allow collaboration between cultural heritage institutions - more specifically, between art museums. Currently there is a need for museums

*This work has been partially supported by FAPESP (Foundation for Research Support of the State of São Paulo) under the process 2010/52505-0.

to seek new business models for their survival and an approach that is emerging is exactly the virtual presence of these institutions in the Web, providing services to users and minimizing their operational costs. This virtual presence can bring great benefits to society, not only by providing services related to communication and display of artworks, but also by fulfilling the role of museums in its entirety, including the safeguarding and preservation of tangible and intangible heritage and scientific analysis to understand and establish its significance and its possession. At this point, we believe that interoperability between museums acquires a broader focus, because it is based on the exchange of information between these institutions to enrich their interpretations of their collections, artists, exhibitions, techniques and historical account of their items.

Nevertheless, most of research found in the literature for this domain has focused on enhancing physical navigational facilities (e.g. [Cosley et al. 2009]) or on the 3D virtual reconstruction of museums and archeological sites. Besides, information sharing has been oriented towards the use of global integration models. Therefore, institutions are called for a global semantic agreement for the data stored at their repositories. However, due to the variety, complexity, amount, size, amount of non-standardized resources, a large scale knowledge integration may not be an effective approach in this scenario.

Interoperability among museums relates primarily to sharing distributed information about artwork, managing the exchange of artwork for temporary exhibitions and designing/managing joint exhibitions. In order for all this to occur, it is essential that these institutions (museums) have a clear definition of the concepts they share. In the case of art museums, the shared concepts refer to artworks in their collections and items participating in exhibitions (which may be part of the collections of others). Shared concepts also involve other organizations such as the cultural institutions that perform exhibitions, artwork collectors that can assign their items to specific exhibitions, and sponsors of cultural institutions. Last but not least, we need to consider the users of this specialized sort of information systems, ie visitors to art exhibitions, who need to interact with the items in an exhibition as well as with the institutions that organize the exhibition, this way characterizing interoperability between museum and visitors.

OpenMuseum is based on the use of a lightweight interoperability platform for the coordination of interactions between heterogenous service-based applications. This loosely coupled approach is important since it minimizes the internal changes in the participating institutions, while preserves the independence of these institutions. The use of a lightweight platform is important to ensure that the benefits of knowledge sharing are greater than the (computational) costs ensued by it [Constant et al. 1994].

In OpenMuseum, heritage objects may change location dynamically; thus location-dependent services and protocols, capable of tracking and locating objects, are required. Based on this fact, the first OpenMuseum prototype was implemented using the JamSession platform. JamSession was proposed in [da Silva 2012] as a language for specifying and executing interaction protocols in distributed environments. Its architecture assumes that the environment is composed by a set of locations, which are related by pathways to form a directed graph. The locations are inhabited by passive entities which may be seen as tokens, agents or resources. Protocols are the active components which rule the use of the passive entities (hereafter considered as resources) by the clients. The resources are manage by predicates and can be moved between locations. Protocols combine movements and predicates using logical connectives, and can be nested and recursively defined. Protocols and predicates are attached to locations where they are executed.

JamSession provides a conceptual tool for coordination of services and sharing computational resources. Its current implementation allows the encapsulation of services into predicates in a very straightforward manner. But, only few data can be attached to resources, i.e. identifier, type information and location. This limited set of information has shown to be be enough in those scenarios where the resources are interpreted as control flow tokens, e.g in choreographies for inter-organizational workflows [da Silva et al. 2013] or the coordination of location-based multimedia applications. However, more complex scenarios require effective knowledge-based representations to structure and manage the information resources. In this paper we describe the main directions in which JamSession platform will be extended to attain OpenMuseum's goals. The extensions are not domain specific. Therefore, we expect that the resulting platform will be useful for building cooperative applications between heterogenous systems.

The paper is organized as follows. In Section 2 we present our motivation, the OpenMuseum project, along with the related work. Section 3 briefly introduces the JamSession language. Section 4 describes how the first prototype for OpenMuseum was modeled using JamSession. Section 5 outlines the extensions currently in development for OpenMuseum. Some concluding remarks are pointed out in Section 6.

2. OpenMuseum's scenario

OpenMuseum is the result of an ongoing project between the Museum of Contemporary Art[1] and the Institute of Mathematics and Statistics at the University of São Paulo. The aim of OpenMuseum project is twofold. Firstly, the development of a tool to provide physical museums with a virtual presence that can be reached worldwide. This tool will provide two basic functionalities: (1) the storage of information and knowledge about cultural heritage objects; and (2) a virtual space of the museum as a 2D or 3D representation of its physical space with navigational facilities. The virtual space should be available to a large spectrum of users including visitor, teachers, scientists, collectors, etc. Thus, it should provide an automatic adaptation of the format and content displayed, depending on user profiles. The data may be used by a variety of applications, like websites, interactive applications and mobile computing devices, etc; therefore different views of the same information are required. A third planned functionality is the possibility of the virtual reconstruction of past exhibitions and the dynamic creation of new ones. The virtual exhibitions may include objects belonging to different institutions.

Several applications have been developed in this domain using 3D virtual reality. For example, the Google Art Project[2], provides an online art experience where users can take a virtual tour of many museums and even build their own collections to share. The British Museum[3] provides a Website where users can search over five thousands highlight objects and can navigate through different collections. The Vatican Museums also provides some virtual tours including the visit to the Sistine Chapel[4].

[1] http://www.mac.usp.br/mac/

[2] http://www.googleartproject.com/

[3] http://www.britishmuseum.org/explore/online_tours.aspx

[4] http://www.vatican.va/various/cappelle/sistina_vr/index.htm

In these works, the goal is to provide a virtual presence of art collections that can be reached and experienced by users world wide. Another approach comes from the area of multi-agent systems, where agents are used to understand and simulate the behavior of ancient societies or recently to provide a group experience of the virtual visits [Costantini et al. 2008, Bogdanovych et al. 2009, Yee-King et al. 2013]. The technology employed is based on Virtual Electronic Institutions and 3D visualization tools. Here, past or present scenarios are constructed, characterized by norms, agents that play roles and an environment where complex interactions take place. Nevertheless, these works are restricted to a single organization, population or culture.

In contrast to the above initiatives, the second goal of OpenMuseum is the creation of a platform for knowledge sharing and collaboration between institutions aiming to safeguard tangible and intangible world cultural heritage. Here, sharing is not associated with maximizing the access of the museum's repositories by means interactive virtual tools. Our approach is to use sharing for collaboration, i.e. participant institutions work together to achieve common tasks. Therefore, not only digital information but also services and processes should be shared. Several challenges should be overcome to attain this goal. The cultural heritage domain is highly heterogenous combining historical, archeological, cultural, educational, scientific, and commercial institutions. Each institution system may have a vast amount of digital information with no uniform representation and classification. The information may comprise text, high resolution images and videos and augmented reality 3D scenarios. The same data may be stored in different institutions under different formats. A number of other sensitive aspects concerning technology (hardware, software and network platforms), legacy systems (norms, laws, digital rights, privacy, fair use), etc, should also be considered.

An example of knowledge-based collaboration platform in the cultural heritage domain is WissKI project[5] [Hohmann 2009]. The platform uses ontologies to harmonize and provide an integrated access to the digital information repositories. The goal of the project is to support communication, documentation and exchange of information between scientists and researchers from several memory institutions (libraries, archives, and museums). However, more complex cooperative tasks cannot be performed. Another initiative, the eCultura platform [Cornejo et al. 2012], is based on the extension of the Java/J2EE collaborative portal Liferay with a suite of services and applications for the cultural domain. It also allows the use of ontologies as knowledge repositories; but firstly they should be integrated into the platform by means of ontology mappings. Besides, no support is provided for coordination of interorganizational activities. To successfully face the aforementioned domain challenges, a collaboration platform should combine knowledge-based representations with an effective mechanism for process interoperability.

3. JamSession Overview

JamSession is a middleware designed to simplify the coordination of distributed and heterogenous resources and services. The services are encapsulated within predicates which are responsible for managing the shared resources. The resources are hosted at specific locations (which represent physical or logical sites) and can be moved between them.

[5] `http://wiss-ki.eu/`

Therefore, the coordination mechanism of JamSession uses a directed graph in which nodes represent locations and the arcs characterize admissible migrations or movements for the resources. A configuration of the graph or environment is a function indicating the location of each resource. The composition of the services is implemented by means of interaction protocols which are also linked to locations. Several protocols can be run in parallel while sharing the same environment of resources.

A protocol definition in JamSession has the syntax $[l]\ K(\widetilde{x}, \widetilde{y}) ::= F$ where F is a formula in disjunctive normal form. The literals in F may be the constants $\perp$ and $\top$, move orders and protocol or predicate calls. A move order **move**(r, l_1, l_2) is executable if the current location for the resource r is l_1 and (l_1, l_2) is an arc of the graph of locations. The syntax of a predicate call is $[r, l]\ \rho(\widetilde{x}, \widetilde{y})$, i.e. each predicate name ρ operates using a resource hosted at some location. The (possibly empty) tuples $\widetilde{x}$ and $\widetilde{y}$ represent the input and output parameters. The special constants $\perp$ and $\top$ represent the failure or the success of an entity call. The conjunction denotes a sequential evaluation of the atoms while the disjunction represents an alternative branch of computation if the previous one fails. JamSession was firstly presented using a graphical language which simplifies the design of the protocols [da Silva 2012]. However, in the present article we use the plain text notation.

In the current JamSession's implementation, a cloud-based server stores and manages the information about clients, locations and resources. Whenever a JamSession client is subscribed to the server, it exports to the server a collection of predicate interfaces and protocol definitions which are locally stored at the client. Although JamSession's language does not include explicit constructions for asynchronous communication, the exchange of messages is the base of its coordination mechanism. The communication between clients and the server for executing the interaction protocols and predicates is performed by means of web services. As depicted in Figure 1, a (move, predicate or protocol) call represents a request-response operation. Whenever a client or an application needs to trigger a predicate/protocol, it sends a request to the server and waits till the response message. If the request is valid, the appropriate client is activated and the predicate/protocol definition is effectively triggered and executed.

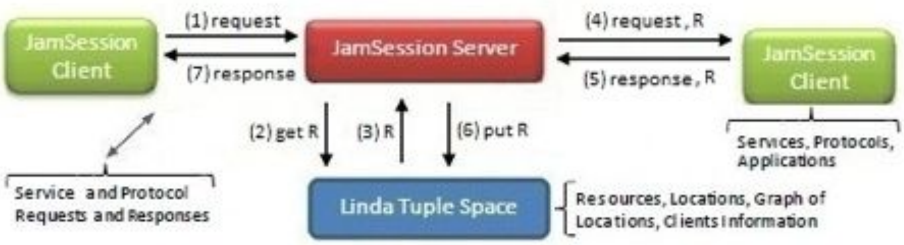

Figure 1. Architecture and message exchange in JamSession.

In addition, predicate calls and move orders must obtain exclusive access to the corresponding resource before they can be executed. This way a resource can be accessed by just one predicate at any given time, when it is hosted at the location required by the predicate. As long as the predicate is being executed, the resource is locked and it is impossible for other processes to use it. The lock is released when the process finishes the execution. Other predicates and move orders willing to use the same resource are suspended until the resource is released and reaches the expected location.

To illustrate the functioning of JamSession we use a small example describing a basic shopping service. The graph of locations, shown in Figure 2, has three locations. The $customer$ location is inhabited by users who may or may not be interested in buying goods. In the first case, a customer should move to the $buyer$ location and after the purchase is completed, to $done$. If the customer cannot afford the desired item because of the price, it is moved back to the $customer$ location. Two additional locations are used for the messages which are exchanged between the buyer and a seller. These locations are related in a bidirectional way. The messages are represented by the tokens $askMsg$, $buyMsg$, $priceMsg$ and $soldMsg$

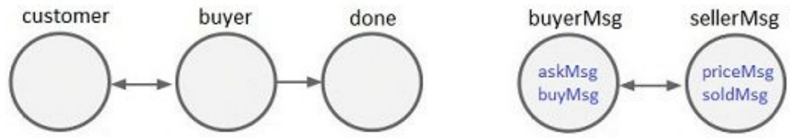

Figure 2. Graph of locations for a basic shopping service.

The protocols corresponding to the buyer and seller roles are shown below. For brevity, we have omitted the parameters of the predicates. We have also used c, b, d, bM and sM to abbreviate the location names. The $pBuyer$ protocol has a user B as input parameter which needs an item X; hence, B should be hosted at c and then moved to b. The $updateAsk$ predicate stores the required data values for the $askMsg$ token and the message is sent to the sM location. After that, the $getPrice$ predicate waits until $priceMsg$ reaches the bM location, carrying the price of X as data value. When this occurs, the message $priceMsg$ is sent back to the sM, as an acknowledgment and to allow other sellers to use it. After checking that the customer can afford X at price P, the $buyMsg$ token is updated and sent to sM. The $getConf$ predicate waits till the $soldMsg$ message reaches bm. Once the message is received, it is sent back and the purchase is completed by moving B to d. If the price cannot be paid, then the purchase is cancelled, the $buyMsg$ token is updated and sent to sM and the customer returns to c. The behavior of the $pSeller$ protocol is analogous.

$[b]\ pBuyer(B, X) ::=$
$\quad \textbf{move}(B, c, b) \wedge\ [askMsg, bM]\ upAsk()\ \wedge\ \textbf{move}(askMsg, bM, sM) \wedge$
$\quad [priceMsg, bM]\ getPrice()\ \wedge\ \textbf{move}(priceMsg, bM, sM) \wedge$
$\quad [B, b]\ afford()\ \wedge\ [buyMsg, bM]\ acc()\ \wedge\ \textbf{move}(buyMsg, bM, sM) \wedge$
$\quad [soldMsg, bM]\ getConf()\ \wedge\ \textbf{move}(soldMsg, bM, sM)\ \wedge\ \textbf{move}(B, b, d)\ \ \vee$
$\quad [buyMsg, bM]\ cancel()\ \wedge\ \textbf{move}(buyMsg, bM, sM)\ \wedge\ \textbf{move}(B, b, c).$

$[sh]\ pSeller() ::=$
$\quad [askMsg, sM]\ getAsk()\ \wedge\ \textbf{move}(askMsg, sM, bM) \wedge$
$\quad [priceMsg, sM]\ instock()\ \wedge\ \textbf{move}(priceMsg, sM, bM) \wedge$
$\quad [buyMsg, sM]\ okBuy()\ \wedge\ \textbf{move}(buyMsg, sM, bM) \wedge$
$\quad [soldMsg, sM]\ conf()\ \wedge\ \textbf{move}(soldMsg, sM, bM)\ \wedge\ [soldMsg, sM]\ done()\ \ \vee$
$\quad [buyMsg, sM]\ cancelSale()\ \wedge\ \textbf{move}(buyMsg, sM, bM).$

JamSession was implemented using Microsoft .NET and the functional programming language *F#*. Predicates are specified either as *F#* or *C#* classes. The service implementation is embodied in the *Execute* method of the class associated to the predicate. The resource and location are input parameters of *Execute*; this way all the cases for which the predicate is defined can be handled in just one method. Each resource has an identifier and also a string value or tag which can be read or written inside the predicate. This string value and the input and output variables are also parameters of *Execute*. Each predicate class is compiled independently to an executable library (*.dll*). An implementation of the *TupleSpaces* concept[6] is used for managing the resource information stored at the server. Therefore, resource movements are always executed at the server.

4. OpenMuseum first prototype

OpenMuseum is being implemented as a web application using *ASP.NET MVC 3*[7]. The site has integrated a JamSession client (*hub*) which contain the common functionality shared by all the museums and the services available to external applications. In addition, each museum has also a JamSession client which provides the implementation of protocols and predicates for using the local resources. The graph of locations stored at the JamSession server has the structure as shown in Figure 3 (left upper corner). Each museum was associated to a main location connected to a central node in the graph. The central node is attached to the hub client and it is the only way through which resources traverse between museums. The museums may have also a subgraph of private locations possibly linked to the representative one. Note that the museum subgraphs are disjoint and disconnected except by the hub node. Any protocol or predicate intended for sharing is attached either to the hub or to a museum main location.

Figure 3 also shows a subgraph, linked to the central location, which is used for creating a shared virtual exhibition. A protocol attached to the $start$ location performs the user authentication and calls a protocol for creating a new exhibition or editing an existing one. Both processes involve the selection of paintings from different museums through a web-based interface. Here, the searching predicates associated to each museum are used for retrieving the objects matching a search pattern. At any time, it is possible to obtain a visualization of the exhibition. The views can be 2D or 3D, although the later possibility is still under development. Before finishing the exhibition, a protocol is triggered to enforce the digital rights of the selected items. It uses a subgraph of auxiliary locations, here represented by the box labelled as $Permissions$. Besides, the workflows or protocols corresponding to the digital permissions in each museum are activated. When the process terminates with success the exhibition is made public and shared, by moving it to the hub location.

In JamSession, services are indeed distributed and effectively managed. However, the same does not applied to resources. They all are stored at a Linda-like tuple space at the server and the total number cannot be dynamically modified. The structure of the graph cannot be modified either. Thus, adding a new location (or museum) implies recreating the whole graph. Furthermore, as we mentioned above, the data attached to each resource is restricted to a string value. Therefore, the information resources in OpenMuseum (heritage objects and exhibitions), their structure and relationships cannot be

[6] `http://sqlspaces.collide.info/`

[7] `http://www.asp.net/mvc/mvc3`

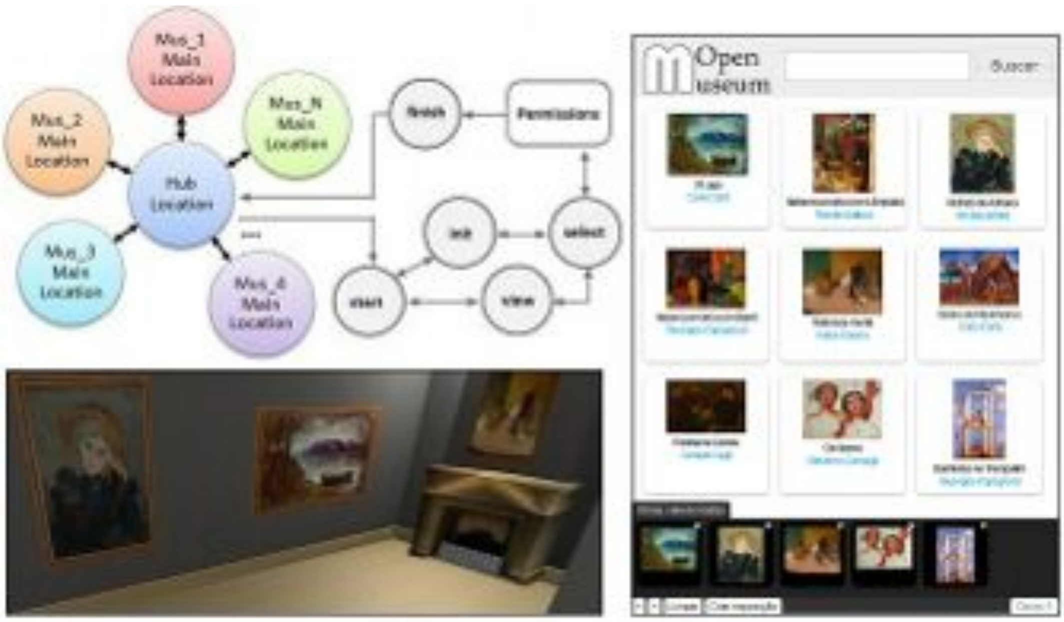

Figure 3. Subgraph of locations and screenshots for creating a virtual exhibition in OpenMuseum.

modelled using JamSession resources. In order to fully implement OpenMuseum using JamSession, a number of extensions must be performed in the platform to overcome these limitations.

5. OpenMuseum's next steps

In broad terms, the design of an interorganizational system for sharing and managing information resources should deal with three important issues: the *organizational structure*, the *information model* and the *behavioral model*. The organizational structure includes the participant institutions and their roles, the resources to be shared and their physical locations; resource flow between the participants, additional physical and logical locations, etc. It is the foundation for an effective design of the system architecture. The information model describes the knowledge to be share for each information resource. The knowledge is the most valuable information accumulated for the resources and it provides the logical view of the resources in the holistic system. It may be described by means of concepts, categories, descriptors, entities, attributes, relations, constraints and rules for reasoning.

The behavioral model describes the main processes or activities performed within each participants, what resources they need, how these processes are interrelated and how they should be reflected in the holistic system. Process-activity models and its dependencies are usually described by means workflows which may be private, public or inter-organizational. Procedures, services, policies, protocols and events intended for internal collaboration, resources sharing or interaction with external applications are also specified in this model. Inter-organizational systems are highly complex and it is very important to ensure the correct design and proper functioning of the holistic system from early stages of development. To this end, the specification should be formally grounded allowing the use of techniques for verification. Therefore, a important stage on the development should be the verification or the automated generation of the main protocols.

To cope with the above issues, OpenMuseum architecture evolved as depicted in Figure 4. As before, the server will orchestrate the interactions in the system. The whole graph of locations is stored at the server using a tuple-space model. However, each JamSession client manages its own subgraph of locations and resources. The most relevant information about the resources will be stored at a knowledge repository attached to the client. The common knowledge (resources, protocols, predicates) to all participant institutions may be either distributed between them or preferably located at a specific client connected to the server. The locations of this client will be shared and may connect the private subgraphs. The main application and the external ones will interact with the system through this client.

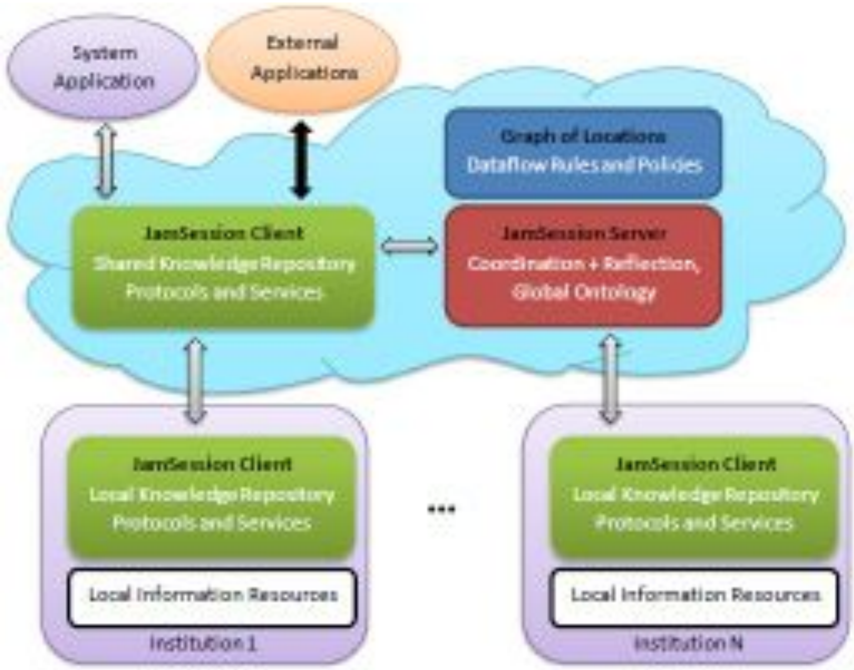

Figure 4. OpenMuseum's architecture.

We have chosen the use of ontology-based knowledge repositories. Since the middle of 1990s, ontologies have been increasingly used in information systems for knowledge and business process integration, interoperability, sharing and management [Farquhar et al. 1995, Noy et al. 2000, Jurisica et al. 2004, da Silva et al. 2010]. They provide a suitable description for concepts, properties and instances of heterogeneous resources. We will focus on ontologies for the semantic web, in particular those built using the W3C's Web Ontology Language (OWL). In particular, the sub-language OWL-DL offers several advantages such as readability, expressiveness, decidability and tools for editing and reasoning (e.g. Protégé, SWOOP, Pellet, Racer, Fact++, etc.).

JamSession is being adapted to use a global shared ontology stored at the server. This ontology should be constructed as result of a collaborative effort of domain specialists. It will provide the minimal consensual vocabulary and knowledge accepted by all participant institutions. Each institution may create its own ontology (as extension of the global ontology) and each JamSession resource must be associated to an individual. For OpenMuseum, the global ontology will be based on the formal ontology recommended by the literature: the CIDOC Conceptual Reference Model (CRM)[8]. This extensible ontology is the international standard (ISO 21127:2006) for integration and interchange of heterogeneous cultural heritage information. Besides, an OWL-DL implementation of CRM has already been developed in the context of the WissKI project[9] [Görz et al. 2008].

The link between JamSession and OWL-ontologies is being implemented using the Protégé-OWL API, an open-source Java library [Knublauch et al. 2004]. This API allows to load and query OWL data even using external reasoners. Protégé provides several tools for manipulating the ontologies, even in a collaborative way. The OWL data models may be stored on top of relational databases such as Oracle, MySQL, Microsoft SQL Server, and Microsoft Access. Nevertheless, other local knowledge repositories may be integrated with JamSession clients. For instance, for OpenMuseum we plan to run experiments with two large scale repositories: SOR-Scalable Ontology Repository (previously Minerva) [Lu et al. 2007] and DLDB-OWL (HAWK) [Pan and Heflin 2003].

The JamSession underlying message passing allows a smooth integration with ontology mappings. Note that each time a resource should be moved, a special plug-in predicate or service may be activate by the server in order to perform the mapping corresponding to the destination location. The ontology mappings should be located on clients. Nevertheless, global mappings may be stored at the server level to be applied when there is no available mapping at the involved clients. Internal movements, i.e. those performed between locations of the same client, require no mapping. Specific actions for exchanging data in the movements should be defined on the global ontology.

JamSession has already proved its efficacy for modelling and coordinating inter-organizational workflows [da Silva 2012]. Using this platform, workflows and activities may be modelled by means of knowledge-based interaction protocols and predicates. Furthermore, workflow definitions may be kept local to each workflow management system and just the interaction protocols for the composition patterns are made public. In addition, a reflection mechanism is required to fully implement OpenMuseum behavioral model. According to [Maes 1987], a middleware is reflective if it includes internal structures causally linked to its own behavior. These structures should permit the inspection, reasoning and adaptation of the system self-representation.

In the case of JamSession, one of these internal structures corresponds to the graph of locations. Therefore, the initial stage towards a reflective extension is to provide suitable forms of exposing the current state of the graph. The incremental modification of the topology of the graph and the addition of new resources should also be permitted. Keeping track of past resource movements will be also allowed. This is a relevant feature for OpenMuseum, for preserving the record of heritage objects. It will allow to reason about properties, i.e. temporal ones, which cannot be represented in OWL-DL.

6. Conclusions

In this work we presented an ongoing project, called OpenMuseum, for collaboration and sharing between cultural heritage institutions. Although most of the work developed for

[8] http://www.cidoc-crm.org/
[9] http://www.erlangen-crm.org/

this domain is oriented towards the use of global models for the integration of information resources, OpenMuseum is been designed using a loosely coupled approach where institutions collaborate through a user-friendly coordination middleware called JamSession. JamSession provides a suitable framework for composing services and mediating cross-organizational workflow interactions. However, it imposes restrictions for managing and sharing complex information resources.

A practical solution for fully implementing OpenMuseum turned out to be the extension of JamSession focusing in its integration with OWL-DL ontologies and the addition of some reflective facilities. The ontology-based model provides a flexible, powerful and distributed representation of the semi-structured data in the domain. The reflective extensions allows modifying and reasoning about the system environment. Some theoretical work has been done towards formal verification proofs in JamSession. In [da Silva et al. 2013] it was shown that for certain protocols, a translation into Colored Petri Nets (CPN) is possible; thus, any CPN-tool can be used. This result is being generalized using more powerful nets. The technique will provide a framework within which the design of the protocols can be verified (e.g. via model checking).

We believe that OpenMuseum will result in a reliable tool for interoperability, sharing and collaboration in the cultural heritage domain. We also believe that, due to the use of CIDOC-CRM, it may easily become part of other initiatives in the domain. In particular, we aim its integration with a domain communication platform (e.g. WissKI) since it will improve the collaboration model. It is well known that an effective realization of collaboration can be achieved through the interaction of communication, co-operation and coordination tools. This 3C model has been previously applied with success to the development of groupware systems [Gerosa et al. 2006, Pimentel et al. 2008]. The use of CIDOC-CRM entails a wide range of applicability for OpenMuseum due to the variety of the cultural heritage information (e.g. it covers geographical, natural history and even history of technology). Besides, OpenMuseum is being designed in a domain-neutral way, i.e. the global ontology may be changed in order to deal with other domains. This approach may be used for conceptual modelling and implementation of other information systems intended for collaboration between heterogenous interactive applications.

References

Bogdanovych, A., Papaleo, L., Ancona, M., Mascardi, V., Quercini, G., Simoff, S., Cohen, A., and Traverso, A. (2009). Integrating Agents and Virtual Institutions for Sharing Cultural Heritage on the Web. In *Proc. Workshop on Intelligent Cultural Heritage*.

Constant, D., Kiesler, S., and Sproull, L. (1994). What's mine is ours, or is it? a study of attitudes about information sharing. *Information Systems Research*, 5(4):400–421.

Cornejo, C., Ruiz-Rube, I., and Dodero, J. (2012). Semantic management of digital contents for the cultural domain. In *Recent Trends in Information Reuse and Integration*, pages 211–226.

Cosley, D., Baxter, J., Lee, S., Alson, B., Nomura, S., Adams, P., Sarabu, C., and Gay, G. (2009). A tag in the hand: supporting semantic, social, and spatial navigation in museums. In *Proc. SIGCHI Conference on Human Factors in Computing Systems*, pages 1953–1962. ACM.

Costantini, S., Mostarda, L., Tocchio, A., and Tsintza, P. (2008). Dalica: Agent-based ambient intelligence for cultural-heritage scenarios. *IEEE Intelligent Systems*, 23(2):34–41.

da Silva, F. S. C. (2012). Knowledge-based interaction protocols for intelligent interactive environments. Knowledge and Information Systems, 30:1–24.

da Silva, F. S. C., Venero, M. L. F., David, D. M., Saleemb, M., and Chung, P. W. (2013). Interaction protocols for cross-organisational workflows. Knowledge-Based Systems, 37:121–136.

da Silva, T. A. M., Santoro, F. M., and Baião, F. A. (2010). Suporte computacional à integração entre processos de negócio e ontologias de domínio - acessando o Protégé através do ARIS. In *Simpósio Brasileiro de Sistemas de Informação*.

Farquhar, A., Fikes, R., Pratt, W., and Rice, J. (1995). Collaborative ontology construction for information integration. Technical report, Stanford University.

Gerosa, M. A., Pimentel, M., Fuks, H., and Lucena, C. J. P. (2006). Development of groupware based on the 3C collaboration model and component technology. In *Proc. 12th Int. Conf. on Groupware: Design, Implementation and Use - CRIWG*, pages 302–309.

Görz, G., Schiemann, B., and Oischinger, M. (2008). An implementation of the CIDOC conceptual reference model (4.2.4) in OWL-DL. In *Proc. CIDOC - The Digital Curation of Cultural Heritage*.

Hohmann, G. (2009). An ontology-based communication system for cultural heritage: Approach and progress of the WissKI project. In *SCCH09 - Scientific Computing & Cultural Heritage*.

Jurisica, I., Mylopoulos, J., and Yu, E. (2004). Ontologies for knowledge management: An information systems perspective. *Knowledge and Information Systems*, 6(4):380–401.

Knublauch, H., Fergerson, R. W., Noy, N. F., and Musen, M. A. (2004). The Protégé OWL plugin: An open development environment for semantic web applications. In *The Semantic Web - ISWC*, volume 3298 of *LNCS*, pages 229–243. Springer.

Lu, J., Ma, L., Zhang, L., Brunner, J., Wang, C., Pan, Y., and Yu, Y. (2007). SOR: a practical system for ontology storage, reasoning and search. In *Proc. 33rd Int. Conf. on Very Large Data Bases*, VLDB '07, pages 1402–1405.

Maes, P. (1987). Concepts and experiments in computational reflection. *ACM SIGPLAN Notices, Proc. Object-oriented programming systems, languages and applications*, 22(12):147–155.

Noy, N. F., Fergerson, R. W., and Musen, M. A. (2000). The knowledge model of Protégé-2000: Combining interoperability and flexibility. In *Pro. 12th European Workshop on Knowledge Acquisition, Modeling and Management*, pages 17–32. Springer-Verlag.

Pan, Z. and Heflin, J. (2003). DLDB: Extending relational databases to support semantic web queries. In *Workshop on Practical and Scalable Semantic Systems, ISWC*, pages 109–113.

Pimentel, M., Fuks, H., and Lucena, C. J. P. (2008). Um processo de desenvolvimento de sistemas colaborativos baseado no modelo 3C: RUP-3C-Groupware. In *Simpósio Brasileiro de Sistemas de Informação*.

Yee-King, M., Confalonieri, R., Jonge, D., Hazelden, K., Sierra, C., M. Inverno, L. A., and Osman, N. (2013). Multiuser museum interactives for shared cultural experiences: an agent based approach. In *Proc. 12th Int. Joint Conference on Autonomous Agents and Multiagent Systems-AAMAS'13*. In Press.