Cover Page for Paper:

A WEB SERVICE-BASED PLATFORM FOR CSCW OVER HETEROGENEOUS END-USER APPLICATIONS

Authors

Georgios-Dimitrios Kapos, Aphrodite Tsalgatidou, Mara Nikolaidou
Department of Informatics and Telecommunications
University of Athens
TYPA Buildings, Panepistimioupolis
Ilisia, Athens, 157 84, Greece
{gdkapos, atsalga, mara}@di.uoa.gr

Presenting Author

Georgios-Dimitrios Kapos

Contact Author

Georgios-Dimitrios Kapos

Keywords

CSCW, web services, interoperability, standards-based collaboration.

A WEB SERVICE-BASED PLATFORM FOR CSCW OVER HETEROGENEOUS END-USER APPLICATIONS

Abstract

In this paper, a flexible and extensible platform for computer supported cooperative work based on web service technology is presented. This platform, called HERMES, enables collaboration among users of heterogeneous applications over the web. Web services provide an open, standard communication infrastructure that eliminates dependencies on proprietary technologies and platforms. Heterogeneous end-user application support is facilitated by the definition of abstract collaboration protocols supporting coordination. Domain specific collaboration protocols are based on domain standards and are specified in terms of the proposed Collaboration Protocol Specification Language. Development of a collaborative business process modeling application using HERMES demonstrates the potentials of the approach.

1 INTRODUCTION

Collaborative environments enable distant users to work jointly through a series of provided facilities, such as resource sharing and on-line text, audio or visual communication. These environments usually consist of end-user applications and the underlining collaboration infrastructure. A broad distinction of end-user applications can be made depending on the coupling between them and the collaboration infrastructure, resulting in *collaboration-aware* and *collaboration-transparent* applications [7]. In the first case, end-user applications are developed according to the requirements and special characteristics of the collaboration infrastructure. Therefore, they provide a rich set of features that facilitate collaboration among users (e.g. advanced resource sharing, instant messaging). In the case of collaboration transparency, preexisting single user applications may be used in collaborative environments. The collaboration infrastructure bypasses the application and exploits lower level concepts, such as file system sharing or input/output manipulation. Therefore end-user applications are completely independent of the underlying collaboration infrastructure. However, the fact that collaboration is transparent to end-user applications has a negative impact on the provided collaboration support, in terms of features and facilities.

Most approaches, either supporting collaboration-aware or collaboration-transparent applications are restrictive in that they do not support collaboration over heterogeneous applications [5], [6], [14]. For instance, users have to use the same type of text editing application (e.g. MS Word) in order to collaboratively create and edit a document. Therefore, collective contribution from applications with distinct features is prohibited. Even in collaboration environments where application heterogeneity is supported [9], [10], it is restricted by factors such as the use of specific component models (e.g. JavaBeans) or programming languages (e.g. Java).

Computer Supported Cooperative Work (CSCW) systems need to be built in an open, modular, scalable, highly extensible and customizable way [6], [8], [11]. Monolithic collaboration infrastructures fail to support evolving collaboration requirements in end-user applications, due to modifications in their complex internal structure. They also mix up application implementation, communication mechanisms, and collaboration rules, resulting in unmanageable systems. This effect is greater in collaboration-aware applications, where the coupling between end-user applications and infrastructure is tighter due to the large set of provided features. Thus, the need for modular, tailorable collaboration infrastructures has been identified and dealt with notions such as collaboration specification languages [2], [4], [8], coordination protocols [3], [11], and other similar concepts. In these approaches each collaboration type is specified by a definition, which determines how each collaboration participant may act while in a collaboration.

We believe that the following requirements need to be supported by any CSCW system:

- Transparent collaboration between users of heterogeneous applications of a specific domain (e.g. different text editing applications).
- Accommodation of different collaboration patterns for different application domains.
- Adjustment of collaboration coordination using rules regarding user rights while in a collaborative session (policy schemes).

In order to address these requirements, we have decided to use web services technology as the infrastructure for the development of an open collaboration management platform, which we call HERMES. Web services technology [16] has been receiving great interest in the last few years. The need for intense communication between clients and web services has recently been identified and dealt with [1], [12]. Web services provide a standard communication platform among heterogeneous applications operating on a variety of environments and frameworks. Thus, web services promote interoperability and extensibility. In addition, web service-based systems are open, extensible and reconfigurable. Also, as this technology receives great research attention, new characteristics related to security, reliability or efficiency are continuously considered and added.

HERMES is a web-service based infrastructure for tailorable collaborative environments between distant partners which cooperate, using heterogeneous collaboration aware applications. Tailorability is achieved through a Collaboration Protocol Specification Language (CPSL). CPSL is the means for defining different collaboration types, with distinct features –varying from user entry requirements to action acceptance rules. End-user application heterogeneity is realized by the web service-based underlying infrastructure, which is based on open standards.

Incorporation of preexisting, single-user applications in HERMES framework can be a simple, manageable process. Presentation of the proper adaptation of a single-user business process modeling (BPM) application demonstrates this, as well as other advantages of the framework.

The rest of the paper is organized as follows. In the following section, the functional and architectural characteristics of the HERMES platform are outlined. In section 3, we present the collaboration protocol specification and collaboration coordination in HERMES. Section 4 presents the use of the HERMES platform for collaboration of BPM applications. Finally, in the last section, the main characteristics and advantages of HERMES are summarized, and our considerations for future extensions and research directions are stated.

2 HERMES FUNCTIONALITY AND ARCHITECTURE

In this section we present HERMES, a CSCW framework for collaboration-aware applications based on web services. In order to fulfill the requirements for collaborative frameworks, we followed an action-centric approach for collaboration coordination issues and befitted the utilization of web services technology. This approach resulted in a series of benefits in terms of openness, tailorability and interoperability. The two main entity types of the collaboration framework are the *Collaboration Management System (CMS)* and the *end-user application*. The first is the central collaboration management and coordination infrastructure, while the latter is the application of end-users. Communication of the CMS with end-user applications and between the components of the CMS is carried out according to the web services standards.

Collaboration support in different application domains is ensured by the use of the *Collaboration Protocol Specification Language (CPSL)* that we have developed, which enables the specification of collaboration protocols. A *collaboration protocol* determines the actions that can be performed by participants in a specific type of collaboration and the way these actions are handled by the collaboration management system. For instance, a protocol for collaborative BPM would define domain-specific actions for the creation, editing and removal of business process activities. The actual specification of these actions should be based on widely acceptable standards (not proprietary solutions), in order to enable collaboration between heterogeneous end-user applications.

The main goal of the framework is to enable collaboration over heterogeneous applications and therefore considers:

a) heterogeneous end-user applications of specific domains that:
   i) though different, they support a common format for defining resources and actions of the domain
   ii) have a collaboration-communication module
b) a web-service based collaboration management system that manages and coordinates collaborations

The variety of features that collaboration-aware applications may deliver, such as customization, enhanced interaction, intelligent collaboration support, contrary to collaboration-transparent ones, is the main reason for adopting this feature in our approach.

In order to achieve the aforementioned goal in an attractive and efficient manner, the approach ensures that:

- The addition of the collaboration-communication module in an existing application is a simple, straightforward task that does not discourage developers from extending their applications. However, this does not entirely rely on the framework, but also depends on the extensibility and flexibility of each application itself. Simple, layered architectures facilitate the incorporation of a collaboration-communication module, while complex, monolithic designs complicate it.
- The collaboration management system provides a simple interface to end-user applications, in order to retain genericity. This is achieved by having the

collaboration management system coordinate collaboration actions at a high, declarative level, letting end-user applications define and interpret specific actions. This is achieved through the use of collaboration protocols that define high-level coordination policy over different application domains. Different policies are required for different collaboration types and domains.

- Communication is based on an open, standard technology. This diminishes uncertainty of proprietary solutions, and increases viability and applicability across heterogeneous, restrictive platforms and networks.

2.1 Provided Functionality

As we stated earlier, the framework focuses on actions performed by users rather than collaboratively manipulated resources. Resource manipulation is achieved through appropriate actions, defining the intended resource processing. The actions are delivered to end-user applications by the CMS and are interpreted and performed at each end-user application. Therefore HERMES framework supports distributed resource management and manipulation, retaining the CMS as simple and generic as possible.

There are two different types of actions provided to collaborating end-user applications: *predefined* and *domain-specific actions*. Predefined actions allow users to perform general, collaboration related actions, such as request to join in or exit a collaboration, register a new collaboration protocol, as well as other administrative tasks. Domain-specific actions (e.g. creation of a business process activity in a BPM) are defined in collaboration protocols and used during actual collaboration. Actions of this type may affect resources.

Participants' actions are dispatched as action-messages to the *Collaboration Management System (CMS)*. CMS is responsible for collaboration configuration, as well as handling real-time collaboration actions. Collaboration configuration includes user information management and collaboration protocol specifications management. Real-time collaboration requirements include application input receipt, interpretation and handling, according to the collaboration protocol, messaging management between users, and response to other user requests. The CMS may coordinate multiple collaboration instances of the same or different collaboration protocols simultaneously.

CMS receives actions from collaboration participants, interprets them according to the collaboration protocol, and takes further actions. The latter may include communication with a series of participants for synchronization and consistency maintenance, as described in section 3. This kind of communication is made in terms of a set of primitive messages that each end-user application should be able to handle.

The sequence diagram of Figure 1 presents a sample series of messages that may be exchanged between collaboration platform entities. With messages 1 and 2, the two participants register with the CMS. Message 3 registers a new collaboration protocol (X) and message 4 initiates a collaboration based on this protocol (participant B). Message 5 confirms the initiation of the collaboration and provides its ID. With message 6, participant A requests the list of active collaboration instances based on protocol X, receives the ID (7) of the collaboration earlier initiated by participant B, and joins the collaboration (8). Messages 9 to 12 involve two actions and their respective consistency maintenance messages. Finally, participant A exits the collaboration (13) and participant B ends it (14).

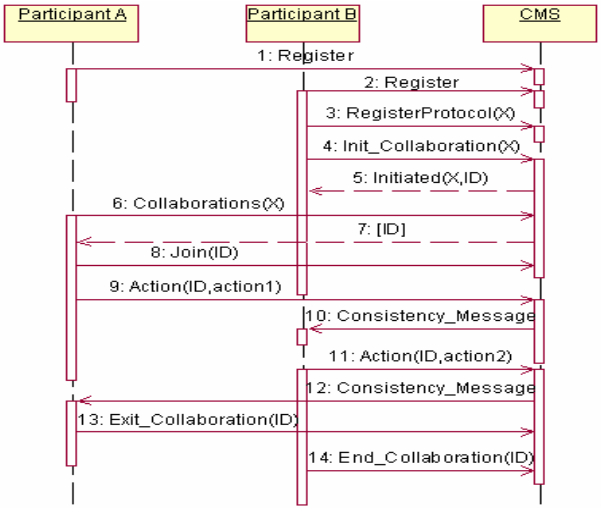

Figure 1. Message exchange sequence diagram.

The basic information flow during collaboration is summarized in Figure 2. Collaboration participants perform actions and receive coordination messages. The Collaboration Management System interprets participants' action requests, according to the Collaboration Protocol, and sends coordination messages, when necessary. The system may handle multiple collaborations of different types concurrently.

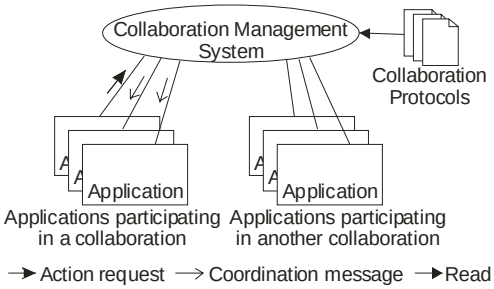

Figure 2. Information flow during collaboration.

2.2 Architectural Design

An architectural view of the CMS (Figure 3) shows that it is a composite web service, consisting of *Collaboration Management Web Service* (CMWS), *Protocol Management Web Service* (PMWS), *User Management Web Service* (UMWS), and *Messaging Management Web Service* (MMWS). As the names of the web services imply, each one is responsible for the management of a different aspect of the system. The CMWS is the main component of the CMS. The other web services manage collaboration protocols, user information and messages.

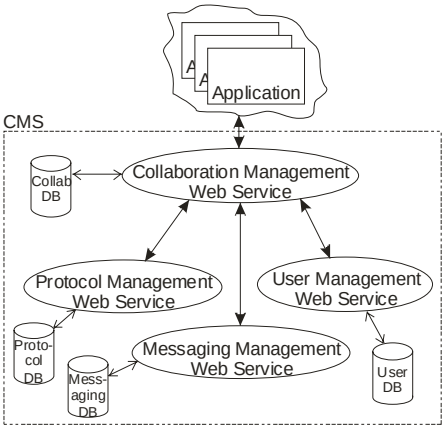

Figure 3. Collaboration Management System (CMS) architecture.

As Figure 3 shows, the CMWS uses the functionality provided by the other three web services, in order to administer the collaboration management information (static part of collaboration) and coordinate all running collaboration instances (dynamic part). It is the heart of the collaboration management system and the interface to the end-user applications. The collaboration protocol interpretation and the coordination of running collaboration instances are also carried out by the CMWS, with the contribution of the other web services, as explained in the following paragraphs.

The PMWS stores collaboration protocols that are being registered to the CMS. It also retrieves or updates registered collaboration protocols specified by the CMWS. Before performing any addition or update to the collaboration protocol database, it performs certain checks, in order to ensure the validity of the new protocol. Similarly, the UMWS keeps a database with information concerning all registered users. Apart from that, user authentication is its most important task.

The MMWS is responsible for delivering text messages exchanged by the collaborating users or sent by the management system to end-users. These messages are high level informative messages for human users and cannot by interpreted by the CMS or end-user applications. This kind of messaging, due to its important and application-independent nature in collaboration, is handled independently of other application-dependent interaction between collaborating users. The MMWS is responsible for the message distribution. Logs of message exchanges are also kept in a database.

End-user applications may be either specially developed for this collaboration environment, or properly adjusted by adding a collaboration-communication module. In either case, the requirements set by the proposed architecture are restricted in the existence of the collaboration-communication module and do not interfere with application internals (representation structures, resource handling). Thus, the integration of existing applications in HERMES framework is facilitated in a flexible manner, in contrast to other approaches. The loose coupling between application internals and collaboration issues reinforces the separation of concerns, which amplifies the clarity of collaboration models and simplifies application implementation.

From a technical point of view, end-user applications need to be able to communicate with the collaboration management system. This is done as defined in the web services standards, via SOAP message transmission over HTTP [13]. Thus, the application does not depend on any proprietary communication mechanism. On the other hand, it is bound to the chosen collaboration protocol. However, this binding can be perceived as a natural, conceptual dependency of the application to its domain. Thus, collaboration protocols are expected to be based on domain standards (e.g. XPDL for business process modeling) and therefore this is an open, non-proprietary binding.

3 COLLABORATION SPECIFICATIONS AND COORDINATION

3.1 Collaboration Protocol Specification

Collaboration protocols are defined with an XML-based language, the *Collaboration Protocol Specification Language (CPSL)*, developed for this purpose. A CPSL document defines a set of specific actions that users may perform during collaboration in a certain context (application domain). Intended interpretation of actions by the collaboration management system is also specified. The interpretation does not deal with action actual semantics, or the resource it may affect. It rather specifies how each action will be carried through during collaboration, so that all users are aware of the same actions. For example, in a BPM context, actions for creating and editing BPM entities should be defined.

The schema of the CPSL is simple and generic. For each collaboration it allows an ID, a textual description and a list of actions. These actions are the only domain specific actions that may be

performed during collaboration. For each action defined in the collaboration protocol the name, a set of parameters and the mode is specified. The name uniquely identifies each action, while the mode deals with action permission, as described later. There may be defined as many parameters as needed for each action, as there is no predefined number of parameters. Every parameter has a name and a type. When an action is requested by a user, those parameters will have to be provided.

As far as allowance to perform an action is concerned, there are three levels of action permission:

- An action is stated to be *free*, when there is no restriction for any participant to perform it. A free action is considered valid upon reception of the appropriate request by the CMS.
- When an action is stated to require *confirmation by the initiator* (the person who initiated the collaboration), it is considered valid only after the confirmation by the initiator.
- When an action is stated to require *confirmation by the majority* of the collaboration participants, it is considered valid only after more than half of the participants have confirmed it.

In case an (non free) action is not confirmed by the initiator or the majority, it is considered invalid and any actions caused by it to requestor application has to be undone.

Table 1 illustrates a sample collaboration protocol for incomings-expenses handling. It defines two actions: one for the addition of an income and one for the addition of an expense.

Table 1. Sample collaboration protocol.

```
<COLLABORATION_PROTOCOL id="INC-EXP"
        description="A collaboration protocol for
                     incomings-expences handling">
  <ACTIONS>
    <ACTION name="AddIncome">
      <PARAMETERS>
        <PARAM name="date" type="DATE" />
        <PARAM name="amount" type="MONEY" />
        <PARAM name="description" type="TEXT"
      </PARAMETERS>
      <MODE type="FREE" />
    </ACTION>
    <ACTION name="AddExpense">
      <PARAMETERS>
        <PARAM name="date" type="DATE" />
        <PARAM name="amount" type="MONEY" />
        <PARAM name="description" type="TEXT"
      </PARAMETERS>
      <MODE type="INITIATOR_CONFIRM" />
    </ACTION>
  </ACTIONS>
</COLLABORATION_PROTOCOL>
```

In order to instantiate a collaboration, administrative policy issues need to be determined. These are specified in an additional document which defines whether users may join, exit, or even end a collaboration instance and how they would do so. Joining a collaboration may be free, require the confirmation by the initiator, the confirmation by the majority of existing participants, or the provision of a password, which is set by the initiator of the certain collaboration. Similar rules may be applied for exiting or ending a collaboration. Rules for sending text message are also defined here. They specify which users may send unicast, multicast, and broadcast messages. The XML fragment of Table 2 defines a sample collaboration policy where joining a collaboration requires the initiators confirmation, exiting the collaboration is free, and ending a collaboration requires the approval of the majority.

Table 2. Sample collaboration policy.

```
<COLLABORATION_POLICY id="Coll_Policy_1">
  <JOIN_COLLABORATION mode="INITIATOR_CONFIRM" />
  <EXIT_COLLABORATION mode="FREE" />
  <END_COLLABORATION mode="MAJORITY_CONFIRM" />
  <MESSAGING>
    <BROADCAST allowed="initiator" />
    <MULTICAST allowed="all" />
    <UNICAST   allowed="all" />
  </MESSAGING>
</COLLABORATION_POLICY>
```

3.2 Collaboration Coordination

In the proposed approach, collaboration protocols play a key role in all collaborations. A collaboration protocol practically defines the way users may collaborate within an application domain, e.g. how users may use their BPM applications jointly in order to create a business process model. In this case, the protocol could specify which standard representation would be used for the business process definitions, what actions could be performed on them, and how these would be coordinated by the collaboration management system. The BPM applications may use their own representations, as long as they exteriorize their behavior in terms of the protocol and the standard representation that it specifies.

The collaboration specification mechanism has been designed to be simple, in order to make the approach generic and easily applicable in many fields. This also has a positive impact in efficiency. It is implied, though, that greater functionality is expected at the end-user application level, for the interpretation and execution of the collaboration protocol-dependent action.

As far as the collaboration infrastructure (CMS) is concerned, interpretation and coordination of user actions is a four-step process, as it is described in the following:

a) *Action-request message receipt*: User credentials are checked and relevant collaboration is identified. In case of failure an error message is returned.
b) *Request interpretation*: The request is interpreted according to the protocol of the specific collaboration. Validity of action name and parameters is checked. In case of success, a list of

further actions is constructed. Otherwise, an error message is returned.

c) *Execution of further actions*: The collaboration management system executes the actions in the list produced in the previous step. These actions are either requests for confirmation by one or more participants, or propagations of the performed action.
d) *Validation or cancellation of the action*: According to the results of confirmation requests, the action is either considered valid or it is cancelled.

3.3 Consistency Maintenance

Within HERMES framework, collaboration is not directly associated with central or distributed resources. It is rather action oriented, since users request actions to be performed. Requests are known to the whole collaborating community and are validated by it. Taking into account that all collaborating users (a) are aware of the same action request, (b) know whether the action has been validated or not, and (c) share the same understanding of the collaboration protocol, it is presumable that consistency is retained after each action. Actions performed by participants are shared within the collaboration group and the independently replicated resources are managed across all collaboration participants' sites by end-user applications.

This scheme seems to require increased user input for the estimation and validation of other users' requests, in comparison to approaches where sophisticated resource management is utilized [4] through resource sharing with locks. However, such approaches [2] focus on a quite lower level and therefore do not support advanced features, like collaboration customization and support for heterogeneous end-user applications. Furthermore, kind of intelligence may be incorporated at end-user application level in order to facilitate users, e.g. by decreasing required user input for action validation purposes. Incorporation of intelligence at the client site will be quite straightforward, as it will not interfere with the entire collaboration infrastructure. Also, different type of user support may be provided for different applications, according to their distinct user interfaces and functionality.

4 COLLABORATIVE BPM WITH HERMES

In this section we present how an existing application may become collaboration-enabled within the HERMES framework. We do this by presenting the adaptation of an existing application for business process modeling. First, we briefly describe the application and our initial considerations and expectations from the transformation. Then we provide the collaboration protocol we defined for collaborative BPM. Finally, we focus on implementation aspects of the adaptation, such the necessary architectural modifications.

4.1 Use Case Context

The application in question is a single user BPM application, facilitating business process modeling using multilevel Modified Petri Nets (MPN) as described in [15]. The approach is based on mapping business process entities to Petri Net elements for the creation of multilevel nets. The formal processing that can be applied on the latter assists the accurate estimation of behavioral characteristics of the corresponding business process.

The BPM application provides a graphical user interface for the creation and manipulation of business process models, expressed in terms of Petri Net elements. Although the internal representation of entities is also based on Petri Nets, an XPDL interface has also been developed for interoperability purposes. XPDL (XML Process Definition Language) is the standard format for defining business processes, proposed by the Workflow Management Coalition (WfMC). Thus, the application could share BPM definitions with other applications supporting XPDL. A XPDL document defines business processes mainly by specifying the activities of the business process, the activity transitions (from one activity to another), and the participants that perform the activities.

By incorporating the MPN BPM application in the HERMES framework we expect to enable collaborative BPM so that distant users of MPN and other BPM applications may collaboratively create business process definitions given that:

a) though different, the applications support the collaboration protocol, which is based on the common format for BPM definitions (XPDL)
b) the applications have a collaboration-communication module capable of communicating with the web service-based CMS

4.2 A Collaboration Protocol for BPM

The collaboration protocol we defined for business process modeling using the CPSL is shown in Table 3. The actions defined in this protocol are based on XPDL and all deal with the manipulation of XPDL elements. These are XML tagged elements that describe various aspects of business processes. Due to the large variety of XPDL elements, the protocol does not define actions for specific elements, but it rather abstractly deals with any element, as long as it is defined in the XML schema definition for XPDL. This shows the ability of the framework to adjust the level of detail, according to user requirements. For the purpose of the example we preferred a simple collaboration protocol, instead of a detailed, complicated one.

The first action (AddXPDLElement) is for the addition of an XPDL element in the business process definition. The XPDL element could be a participant, an activity, a transition, or other business process elements. This is a free action (does not need confirmation by other participants) and requires the specification of the position and the element to be added. Editing of existing elements can be performed with the second action (EditXPDLElement). The element to be replaced and the new element must be specified, while the confirmation by the initiator of the collaboration is required. Finally, the third action (RemoveXPDLElement) allows users to remove an element they have specified provided that confirmation by the majority of participants has been granted.

Table 3. Collaboration Protocol for BPM.

```
<COLLABORATION_PROTOCOL id="BPM-XPDL"
     description="A collaboration protocol for BPM
                  applications, based on XPDL">
  <ACTIONS>
    <ACTION name="AddXPDLElement">
      <PARAMETERS>
        <PARAM name="where" type="XPATH" />
        <PARAM name="what" type="ELEMENT"
               from=
   "http://www.wfmc.org/standards/docs/xpdl.xsd"/>
      </PARAMETERS>
      <MODE type="FREE" />
    </ACTION>
    <ACTION name="EditXPDLElement">
      <PARAMETERS>
        <PARAM name="what" type="XPATH"/>
        <PARAM name="new" type="ELEMENT"
               from=
   "http://www.wfmc.org/standards/docs/xpdl.xsd"/>
      </PARAMETERS>
      <MODE type="INITIATOR_CONFIRM" />
    </ACTION>
    <ACTION name="RemoveXPDLElement">
      <PARAMETERS>
        <PARAM name="what" type="XPATH"
               from=
   "http://www.wfmc.org/standards/docs/xpdl.xsd"/>
      </PARAMETERS>
      <MODE type="MAJORITY_CONFIRM" />
    </ACTION>
  </ACTIONS>
</COLLABORATION_PROTOCOL>
```

4.3 Collaborative BPM with HERMES

In this section we present how our MPN BPM application was adapted in order to fit in the HERMES framework and enable collaborative BPM. The original architectural layout of the application is depicted in Figure 4 (A). The user interface allows users to model business processes in terms of graphically represented Petri Nets. These are processed and stored in the application core with the appropriate data structures. The XPDL interpreter translates XPDL documents to Petri Nets and vice versa. It has been developed to achieve interoperability with other BPM applications.

Figure 4 (B) shows the architecture of the application after it has been modified for use in the HERMES framework. Two major modifications are made:

a) User interface functionality is extended to handle messages and required user input regarding collaboration and consistency.
b) The collaboration module is incorporated and practically becomes a wrapper of the application core.

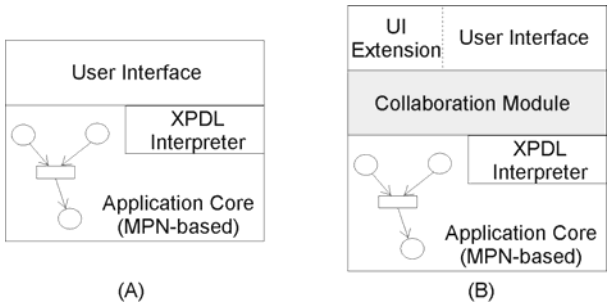

Figure 4. MPN BPM application, single-user (A) and adapted for HERMES (B).

The collaboration module is capable of communicating with the CMS. It receives messages from the user interface and performs combined forwarding to the application core and the CMS. The messages forwarded to the CMS are transformed into collaboration protocol-dependent actions. Since the protocol is based on XPDL, the XPDL interpreter is used for this purpose. Similarly, when it receives messages from the CMS, it interprets them and forwards necessary message to the user interface and the application core. The XPDL interpreter is used here also. Therefore, the extended version of the application fits in the HERMES framework and allows collaborative BPM. The communication is made in terms of the XPDL-based collaboration protocol, allowing collaboration with other BPM applications as well.

We conclude that applications with layered architectures and interfaces to domain standards may easily become collaboration enabled for the HERMES framework, after small scale modifications.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we have presented HERMES, a web service-based generic CSCW infrastructure for collaboration aware applications. The infrastructure may be used in several application domains, as it is capable of dynamically incorporating collaboration protocols that define different collaboration contexts. Collaboration protocols are defined using a Collaboration Protocol Specification Language (CPSL). The collaboration management system is implemented as a set of cooperating web services and thus communication is based on web service operation invocations. The resources are replicated among participant sites in the format used by each application, as collaboration is conceived in an action-centric rather

than a resource-centric manner. The transformation of an existing single-user application to a collaborative application using HERMES was also presented.

This approach offers several advantages that are summarized here. First, the use of collaboration protocols increases the genericity and extensibility of the HERMES platform, allowing its adoption in multiple application domains. Second, collaboration between different applications of the same domain is empowered by the action-based handling of collaboration using standards. Different representation structures and architectural styles do not prohibit collaboration. Third, the use of web services in the collaboration management system induces the benefits of this technology (open and standards-based communication, loose coupling, composability, interoperability) in our approach. Finally, the simple architecture of the HERMES infrastructure and the loose binding between components simplifies the procedure of incorporating the infrastructure in existing applications.

Although HERMES provides a series of advantages and promising features there are still several issues to be studied and exploited. Performance and scalability evaluation are currently explored. Secure collaboration is another feature planned to support. There are cases where constant, real-time collaboration between participants is not required and a batch synchronization procedure once in a time period would suffice. The exploitation of such characteristics will reduce unnecessary load. Finally, as research on web services evolves, several aspects of HERMES system will avail from the new, value-added web service features, such as quality of service.

ACKNOWLDGMENTS

This work has been partially funded by the Special Account for Research (ELKE) of the University of Athens, under contract number 70/4/5829.

6 REFERENCES

[1] Chandrasekaran, S., Miller, J. A., Silver, G., Arpinar, I. B., and Sheth, A. P. Composition, Performance Analysis and Simulation of Web Services Electronic Markets. *The International Journal of Electronic Commerce and Business Media*, 2003.

[2] Cortes, M., and Mishra, P. DCWPL: A Programming Language for Describing Collaborative Work. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, November 1996, pp. 21-29

[3] Edwards, K. Policies and Roles in Collaborative Applications. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, November 1996, pp. 11-20

[4] Furuta, and R., Stotts, D. Interpreted Collaboration Protocols and their use in Groupware Prototyping. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, October 1994, pp. 121-131

[5] Hill, R., Brinck, T., Rohall, S., Patterson, J., and Wilner, W. The Rendezvous Architecture and Language for Constructing Multiuser Applications. In *ACM Transactions on Computer-Human Interaction, Vol. 1, No. 2*, June 1994, pp. 81-125

[6] Kaplan, S., Tolone, W., Bogia, D., and Bignoli, C. Flexible, Active Support for Collaborative Work with ConversationBuilder. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, December 1992, pp. 378-385

[7] Li, D., and Li, R. Transparent Sharing and Interoperation of Heterogeneous Single-User Applications. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, November 2002, pp. 246-255

[8] Li D., and Muntz R. COCA: Collaborative Objects Coordination Architecture. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, November 1998, pp. 179-188

[9] Marsic, I. An Architecture for Heterogeneous Groupware Applications. In *Proceedings of the 23rd International Conference on Software Engineering*, May 2001, pp. 475-484

[10] Marsic, I. DISCIPLE: A Famework for Multimodal Collaboration in Heterogeneous Environments. In *ACM Computing Surveys*, June 1999, Article No. 4

[11] Roseman, M., and Greenberg, S. Building Flexible Groupware through Open Protocols. In *Proceedings of the Conference on Organizational Computing Systems*, December 1993, pp. 279-288

[12] Sheth, A., Cardoso, J., Miller, J. A., Kochut, K. J., Kang, M. QoS for Service-Oriented Middleware. In *Proceedings of the 6th World Multiconference on Systemics, Cybernetics and Informatics (SCI'02), Vol. 8*, July 2002, pp. 528-534

[13] SOAP, http://www.w3.org/TR/SOAP

[14] Trevor, J., Rodden, and T., Smith, G. Out of this world: an extensible session architecture for heterogeneous electronic landscapes. In *Proceedings of the ACM Conference on Computer Supported Cooperative Work*, November 1998, pp. 119-128

[15] Tsalgatidou, A., Louridas, P., Fesakis, and G., Schizas, T. Multilevel Petri Nets for Modeling and Simulating Organizational Dynamic Behavior. In *Simulation & Gaming Journal, Special issue on the dynamic modeling of Information Systems*, December 1996, pp. 484-506

[16] Tsalgatidou, and A., Pilioura, T. An Overview of Standards and Related Technology in Web Services. In *International Journal of Distributed and Parallel Databases, Special Issue on E-Services, 12(2), Kluwer*, September 2002, pp. 135-162