Pervasive Enablement of Business Processes

Dipanjan Chakraborty[1]
University of Maryland Baltimore County
dchakr1@csee.umbc.edu

Hui Lei
IBM T. J. Watson Research Center
hlei@us.ibm.com

Abstract

People are an important part of many business processes. Current workflow-based implementations of business processes constrain users to the desktop environment; require them to periodically check for pending tasks; and do not support direct or synchronous people-to-people interaction. On the other hand, the wide spectrum of people collaboration tools ranging from telephones to instant messaging and to email have no provision for structured collaboration and are separate from business processes. We have designed and implemented a system, dubbed PerCollab, that integrates workflow and collaboration technologies. It allows people to participate in business processes from anywhere using a traditional collaboration mechanism. It proactively engages users in business processes by pushing interaction to a convenient device of the users. PerCollab uses an extended BPEL to formally define business processes with human partners and exploits dynamic user context to solve the personal mobility problem. Our prototype implementation integrates a variety of collaboration tools: instant messaging, email and e-meeting.

1. INTRODUCTION

A business process is "a procedure where documents, information or tasks are passed between participants according to defined sets of rules to achieve or contribute to an overall business goal" [24]. Participants of a business process may be human beings, Web services or other software agents. In particular, human beings form a very important part of many business processes. A great number of large-scale as well as small-scale business processes like product planning, software design, service after sales, travel request approval and candidate evaluation require the engagement of human participants.

Mechanisms concerned with the modeling and execution of business processes are generally referred to as Workflow Management Systems (WFMS) [3,5,13]. A WFMS provides formalisms (e.g. Petri nets [13], task graphs [3]) through which business processes are defined. It also includes a corresponding workflow engine that carries out automatic scheduling and activation of component tasks according to the defined business process. Most existing workflow systems are based on the desktop computing paradigm. They employ a workplace metaphor to present tasks that are to be claimed and performed by human participants. Such tasks differ from tasks that are performed by software agents and are referred to as *staff activities*. A workplace-based user interface has a number of disadvantages: (1) Users are constrained to the desktop computing environment. They don't have access to business processes when they are away from their desktop; (2) A workplace essentially adopts a pull-based approach, where the user is burdened to periodically inspect his workplace to check out pending staff activities; (3) WFMS allow for indirect and asynchronous people-to-people communication only, but not direct and synchronous exchanges among human participants. The latter form of interaction is in fact very common in business environments.

On the other hand, a large array of collaboration and communication tools exist that support direct people-to-people interaction. Tools range from hardware devices like cell phones, pagers and iPAQs to software systems like Short Messaging Service (SMS), Instant Messaging (IM), email and e-meetings. These heterogeneous modalities offer flexibility and extra opportunities in human collaboration. In particular, they allow for synchronous interaction and proactive engagement of people in collaboration by pushing communication messages to them. However, current collaboration tools have their own limitations: (1) They support ad hoc, unstructured collaboration only. There is no built-in mechanism for enforcing any policies or structures on the collaboration process; (2) Collaboration tools are not integrated with business processes. People have to explicitly switch back and forth between business processes and collaboration tools and manually move data between the two.

Our work seeks to address the respective limitations in workflow systems and collaboration tools by effectively integrating the two. Specifically, we want to enable people to engage in business processes anytime and anywhere, using any traditional communication mechanism. Further, we want to add orchestration support to collaboration tools by mediating them with a workflow

[1]This work was performed while the author was visiting IBM Watson Research.

engine. The integration of WFMS and heterogeneous communication tools, however, presents the challenge of personal mobility. Although a person typically has multiple communication tools, he may have access to only a subset of them at a particular time. Depending on the circumstance, he may also have a preference on which of the available tools to use. Thus there is a need to dynamically select an appropriate device or modality to engage the user for a particular interaction.

We have designed and implemented a system, dubbed *PerCollab*, that integrates business processes and ubiquitous collaboration and communication mechanisms. We have extended BPEL [1], a widely accepted business process definition language for Web services, to accommodate human participants. A business process defined in the extended language can be translated to standard BPEL and executed by a BPEL engine. The BPEL engine engages human partners through an Interaction Controller (IC), which serves as a proxy for all interacting human entities. The Interaction Controller selects the most appropriate collaboration modality to reach a user, based on user context information such as location, activity and preferences. To illustrate the features and benefits of our system, we present the following scenario for the business process of travel request approval. The process involves two roles: an employee and a manager. Each role player uses a convenient communication tool to participate in the process.

During the execution of a customer support application ODS, it comes to a decision point that Michael, the technician, should be sent to visit a customer site. In order to comply with the corporate policies, ODS instantiates a travel request approval process via the Web Service interface of PerCollab. The instantiation specifies Michael as the requesting employee and his manager George as the approval manager. Because Michael is currently logged on the Instant Messaging (IM) system, PerCollab starts an IM session with him prompting him to fill out a Travel Request Form. The different fields of the form such as Purpose, Destination and Cost Estimate are presented to him as individual IM messages so that he can answer them one by one.

According to the process, PerCollab now needs to contact George for approval. However, George is in a meeting and does not want to be interrupted. PerCollab thus sends him an email message stating that he has to review a Travel Request Form from Michael and also fill out a Travel Approval Form. George finds the message in his mailbox after the meeting. He happily grants the travel request by filling out the Travel Approval Form and including it in a reply email message to PerCollab. In the next step of the process, Michael needs to be notified of the approval. Although he has logged off IM, he has his SMS-enabled cell phone with him. Therefore, PerCollab sends him an SMS message stating that his travel request had been granted. PerCollab exits the travel request approval process and returns to the calling ODS application with the completion status. ODS then continues.

We can observe five salient features from the above scenario: (1) Business processes (e.g., travel request approval) are seamlessly integrated with and accessible from a range of collaboration tools; (2) Staff activities in business processes are proactively pushed to users; (3) Dynamic user context information is exploited to select an appropriate collaboration modality for engaging the users; (4) Coordination policies and structure can be imposed to synchronous, realtime-style people-to-people collaboration; (5) Collaboration processes can be instantiated programmatically and collaboration results fed back to the calling application.

The rest of the paper is organized as follows. Section 2 discusses several considerations that contributed to the design of PerCollab. Section 3 provides a high-level overview of PerCollab and the interoperation of various components. Section 4 presents extensions of the BPEL language that enables us to explicitly model human participants as a part of the business process. Section 5 describes the details of the various system components. Section 6 presents our prototype implementation. Section 7 discusses related work and Section 8 summarizes our paper.

2. DESIGN RATIONALE

A number of considerations have influenced the design of PerCollab. Among them are integration of WFMS and ad hoc collaboration tools, provision of an orchestration formalism, personal mobility and context awareness.

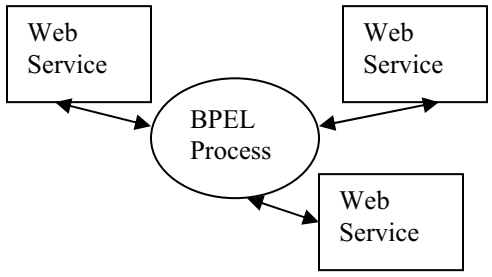


Figure 1: Process-centric BPEL model

As WFMS inherently involve multiple human participants carrying out parts of the business process, it is natural to explore how to synergize people-to-people collaboration tools and WFMS. While both technologies are geared towards supporting collaboration among people in organizations, they differ in their modes of operation and are mostly complementary. Workplace-based

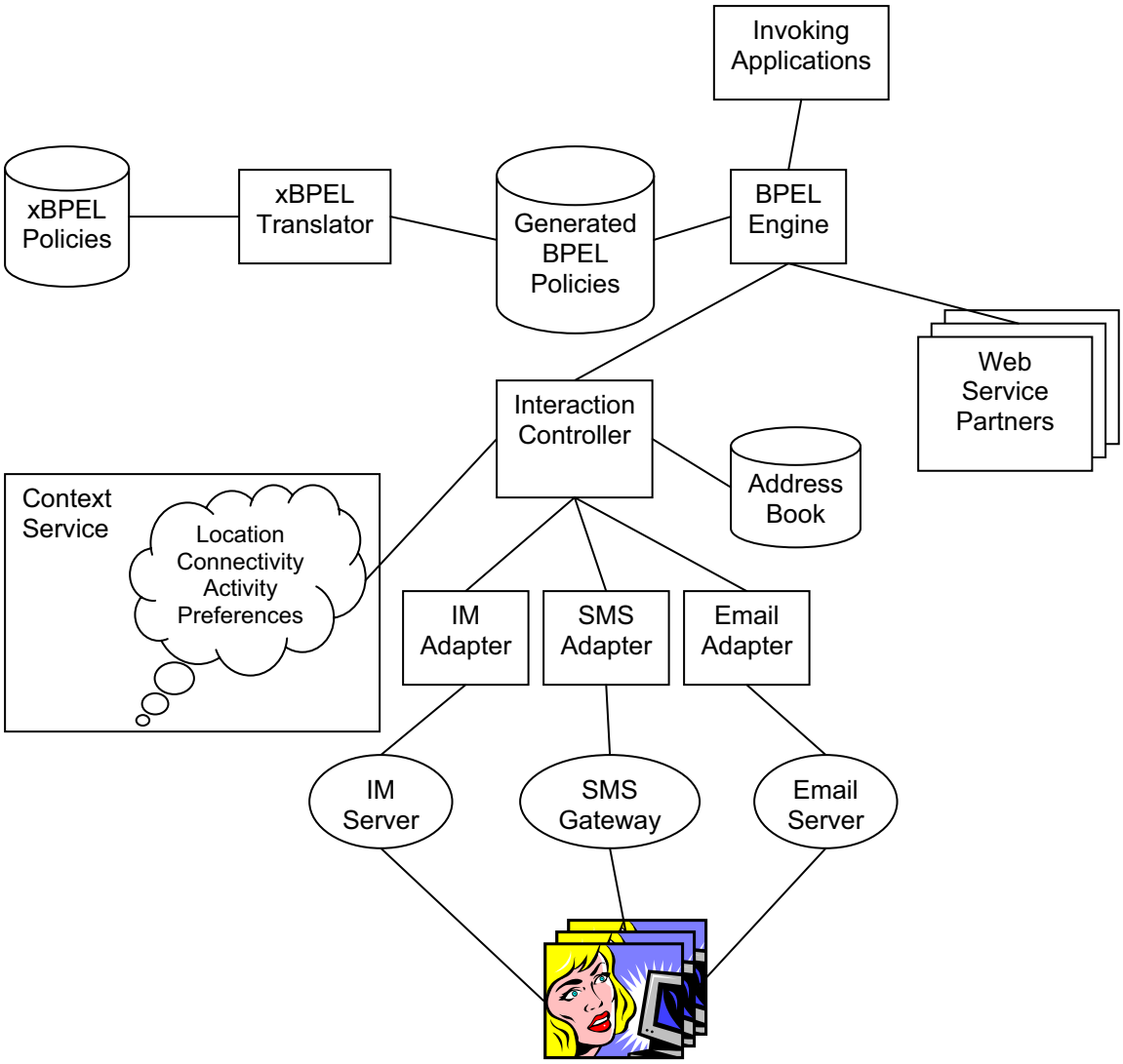


Figure 2: System Architecture

collaboration advocated by WFMS is well orchestrated and structured whereas ad hoc collaboration tools support unstructured collaboration. There is no control on the information being exchanged using ad hoc collaboration tools. Workplace-based collaboration is pull-based and only supports asynchronous collaboration

Ad hoc collaboration tools on the other hand support both synchronous and asynchronous collaboration and are primarily push-based. By integrating with ad hoc collaboration tools, WFMS allow users to participate in business processes anytime and anywhere, in a manner sensitive to their context. Further, a push-based interaction paradigm reduces demand for user attention and can potentially increase user productivity. Ad hoc collaboration tools on the other hand would also benefit from explicit process support in WFMS and provide more value-added features to their users.

Like traditional WFMS, PerCollab requires a process-orchestration formalism. For maximal acceptance and interoperability, we chose Business Process Execution Language (BPEL) [1] as the underlying framework for defining business processes. BPEL is a language for orchestrating Web services, which provide a standard framework for Enterprise Application Integration (EAI). BPEL specifies the potential execution order of operations from a collection of Web services, the data shared between these Web services, which partners are involved and how they are involved in the business process, joint exception handling for collections of Web services, and other issues involving how multiple services participate. BPEL adopts a process-centric view to define a business process, as illustrated in Figure 1. It considers the process as a coordinator of the whole business process; all Web services coordinate with each other

through the process. On the other hand, BPEL assumes all participants (or partners) are Web services and does not provide explicit support for human participants. We have augmented BPEL with additional constructs to accommodate human participants in a business process. More specifically, we provide constructs to explicit model interactions between processes and persons and between persons.

Since PerCollab integrates multiple collaboration tools into business processes, personal mobility becomes an issue. As people move from place to place, their connectivity and accessibility to various collaboration tools may change. Depending on the circumstance, some types of tools may also be more preferable than others. Our system design was presented with the challenge of selecting the most appropriate device on which to engage a user in a business process. Such device binding should be granular enough as the choice of an appropriate device may change in the course of a business process. This also entails the need to support inter-modal collaboration so that people using different collaboration modalities are able to participate in the same business process.

We advocate exploiting a person's context information to proactively select the appropriate collaboration modality [25]. The best means of engaging a particular person at a particular time depends on a lot of factors: the person's location, activity, connectivity and personal preferences. Such attributes are often referred to as user context. By automatically obtaining information on dynamic user context, the system can make intelligent selection decisions on a per-interaction basis. Proactive use of context reduces burden placed on users and potentially increases user productivity and thereby the efficiency of business processes.

3. SYSTEM OVERVIEW

Our PerCollab system enables people to participate in a business process via a convenient communication/collaboration mechanism. Figure 2 shows the overall architecture of PerCollab. It includes the xBPEL Translator, the BPEL Engine, the Interaction Controller (IC), the Context Service, and an extensible set of Modality Adapters.

We use the xBPEL language, an extension of BPEL, to formally specify business processes with human participants. The xBPEL Translator is responsible for translating process definitions (policies) in xBPEL to those in standard BPEL, which are then executed on the BPEL Engine [16]. Each business process has a Web service interface, defined in the Web Service Description Language (WSDL) that is used by invoking applications to instantiate the process. An invoking application can be a standalone business application, another business process, or some user interface mechanism that accepts user commands. The BPEL business process during instantiation accepts configuration parameters such as actual bindings of human participants to user IDs, and a list of modalities acceptable to the invoking application. The list of acceptable modalities is considered in conjunction with individual users' personal preferences in determining the appropriate collaboration modality.

The BPEL engine dispatches tasks meant for Web service partners and human partners. Tasks meant for Web service partners are routed to the corresponding services by the engine directly. Tasks meant for human partners are routed to the IC. The IC acts as a proxy to represent all human participants. It leverages the user context information supplied by the Context Service to select the access mechanism that is most convenient for a particular human participant. The IC also employs an extensible set of Modality Adapters that suffice as access points for specific communication devices and collaboration modalities. The primary job of a Modality Adapter is to interpret tasks being sent by the IC and present it in a modality-specific format. Adapters use device-specific platforms, servers or gateways to communicate with the specified human partner instance.

4. BUSINESS PROCESS DEFINITION

In this section, we present our extensions to BPEL to explicitly accommodate human participants in business processes.* The process-centric model of BPEL uses the BPEL business process (Figure 1) as a mediator for communication with its business partners. In our extension, called *xBPEL*, we distinguish two kinds of business partners: (1) Web service partners: partners that are Web services or agents having a Web service interface; (2) human partners: partners that represent human participants, which don't have a predefined Web service interface. The BPEL Engine governs the business process and communicates with its partners to fulfill business tasks. Staff activities, or tasks meant for human partners, fall into one of two categories: (1) *one-way activities*: notification-type activities for alerting human partners; (2) *two-way activities*: request-response type activities where human participants have to provide a reply back to the business process. We have introduced the following three additional types of constructs in xBPEL:

- *Human Partner*: used to define a human entity as a participant in a business process
- *Process-to-People*: used to model communication between the human partners and the BPEL process

* Readers not familiar with BPEL and its operation can skip this section without any loss in continuity.

- *People-to-People*: used to model direct communication between the human participants. This set of constructs supports a simpler description of direct people-to-people interaction that could also be modeled using process-to-people constructs.

BPEL uses the construct <*partner*> to declare a Web service that participates in the business process. Following this convention, we introduce the construct <*humanPartner*> to declare a human participant. A *humanPartner* essentially defines a role, which represents distinct functions played by a group of people in the business process. Examples of roles are manager, developer, interviewer etc. A *humanPartner* can be bound to one or more instances of actual users or human participants at the instantiation time of the business process. These are referred to as *humanPartner instances* or *humanPartner bindings*. We also allow for specification of constraints on the admission of humanPartner instances to a certain role. An example definition of the construct defining a role of "approver" is as follows:

```
<humanPartner name="manager" role="approver" >
    <admissionConstraints>
        <disjointWith role="travelRequester"/>
        <minCardinality=1/>
        <maxCardinality=1/>
    </admissionConstraints>
</humanPartner>
```

We also introduce one process-to-people construct and two people-to-people constructs. The process-to-people construct is <*interact*>. It models the interaction between a human partner and the BPEL process. Since the BPEL model is process-centric, this construct alone can be used to indirectly model any people-to-people collaboration. We show an example of the *interact* construct below:

```
<interact name="submitTravelRequest"
    humanPartner="travelRequester"
    promptData="$Please fill out the Travel Request Form"
    replyData="travelRequestForm" />
```

The *humanPartner* attribute of the <*interact*> construct denotes the human partner that the process interacts with. The attribute *promptData* defines the message that the process sends to the humanPartner. It may contain a string literal or a typed WSDL message. We differentiate a string literal from a WSDL message by prefixing the former by a '$'. The attribute *replyData* defines the desired response from the human partner. In a one-way activity, the *replyData* is set to *NULL*.

The two people-to-people constructs are <*notify*> and <*converse*>. The former defines a one-way communication from a *humanPartner* sender to another *humanPartner* receiver. The latter defines a two-way request-response between the two *humanPartners.* We provide examples of the two constructs below:

Notify Construct:-

```
<notify name="sendRequestlForm"
    sender="travelRequester"
    receiver="manager"
    promptData="$Please fill out the Travel Request Form"
    senderData="travelRequestForm" />
```

Converse Construct:-

```
<converse name="travelRequest"
    sender="travelRequester"
    receiver="manager"
    promptData="$Please fill out the Travel Request Form"
    senderData="travelRequestForm"
    receiverData="approvalForm" />
```

Figure 3 shows snippets of the xBPEL policy for the travel request approval business process that was alluded to in Section 1. The role players in the example are the travel requester and the manager. Lines 4 –6 describe the human participants using our extension to BPEL. Lines 15 – 43 describe the structure of the collaboration. We observe that using standard BPEL constructs, we can manipulate the data that is being exchanged in the collaboration. Lines 23 – 29 automatically approve the travel request if the 'estimatedCost' of travel is less than $800. The process contacts the manager only if it cannot approve the request automatically. However, if the business process dictates that the requests are sent to the manager, then we could specify the policy using the <*converse*> construct directly. We observe that BPEL also enables exporting the collaboration result to the calling application.

4.1 XBPEL TRANSLATION

We use the xBPEL Translator to convert xBPEL policies to standard BPEL policies for execution on the BPEL engine. The xBPEL Translator primarily performs the following functions:

- Transform all people-to-people constructs into process-to-people constructs. This is done since BPEL follows a process-centric model and all communication between partners actually are mediated by the business process.
- Transform all process-to-people constructs to <*invoke*> constructs in standard BPEL, using the Interaction Controller Web service as a proxy for all human partners.
- Automatically generate the WSDL definition for the resultant BPEL policy. The WSDL definition can then be used by other applications to instantiate the business process.

```
1.<?xml version="1.0" encoding="UTF-8"?>
2. <process>
            ……
3.       <partners>
4.            <humanPartner name="travelRequester" role="traveller">
5.                  <admissionConstraints> <disjointWith role=”approver”/> <minCardinality value="1" /> </admissionConstraints>
6.            </humanPartner>
            ……..
7.      </partners>
8.     <variables>
9.            <variable name="travelRequestForm">
10.                 <message name="travelRequestForm">
11.                      <part name="purpose" type="xsd:string"/> …….. <part name="estimatedCost" type="xsd:string"/>
12.                 </message>
13.           </variable>
            ……..
14.       </variables>
15.       <sequence name="travelSequence">
16.           <receive name="processInstantiation"
17.                 partner="processInstantiator" portType="tns:processInstantiationPort" operation="processInstantiate"
18.                 variable="processInstantiationVar" createInstance="yes">
19.           </receive>
20.           <interact humanPartner="travelRequester" promptData="$Please fill out the Travel Request Form"
21.                 replyData="travelRequestForm" />
22.           <switch>
23.                 <case condition=”bpws:getVariableData(‘travelRequestForm’,’estimatedCost’) < 800”>
24.                     <sequence>
25.                       <assign>
26.                          <copy> <from expression=”approved” /> <to variable=”approvalForm” part=”decision” /> </copy>
27.                       </assign>
            ……..
28.                     </sequence>
29.                 </case>
30.                 <otherwise>
31.                     <sequence>
                          <!-- Manager fills out an approval form -->
32.                       <interact  humanPartner="manager" promptData="travelRequestForm" replyData="approvalForm" />
33.                     </sequence>
34.                 </otherwise>
35.           </switch>
            <!-- Send the approval/rejection back to the travel requester -->
36.           <interact humanPartner="travelRequester"  promptData="approvalForm" replyData="NULL" />
            <!-- Reply is fed back to the business process -->
37.           <assign >
38.              <copy> <from variable="approvalForm" part="decision"/> <to variable="collabOutputVar" part="msgType"/> </copy>
39.           </assign>
40.           <reply name="structProcessReply"  partner="processInstantiator" portType="tns:processInstantiationPort"
41.                       operation="processInstantiate" variable="processOutputVar">
42.           </reply>
43.       </sequence>
44.  </process>
```

Figure 3: Travel request approval policy with *interact* constructs

The IC is a Web service acting as a gateway to human partners. Hence its input and output parameters should be declared as WSDL messages, in accordance with the WSDL schema [10]. A WSDL message consists of several parts representing fields of the message. Each message part is associated with an XML schema data type. The xBPEL variables (e.g., the values of the *promptData* and *replyData* attributes in an *<interact>* construct) are also WSDL messages, representing messages communicated to and from human partners. xBPEL variables may have different message parts (and hence a different message type) depending on the xBPEL policy (e.g. lines 10 – 12 of Figure 3). However, the input and output messages of the IC must have a predefined, generic type. Thus when transforming an *<interact>* statement to an *invoke* operation on the IC, we need to convert between disparate xBPEL variables and generic IC messages.

We address this type conversion problem through message serialization. Specifically, the input message of

the IC contains two parts called 'prompt' and 'reply' that are of type xsd:string. When processing an <*interact*> in xBPEL, we use XML serialization to convert the *promptData* and *replyData* variables to strings and assign them to the 'prompt' and 'reply' parts of the IC input message respectively. Note that on input, the *replyData* variable does not contain any values in its parts. It is serialized and passed to the IC so that the latter knows the fields that make up the message and can prompt the human partner accordingly.

Similarly, the output message of the IC contains one part called 'reply'. Upon returning from the invocation of the IC, 'reply' contains the serialized form of the *replyData* with the message parts filled in. We employ a message de-serialization web service to retrieve parts of the serialized message and assign them to corresponding parts of the *replyData* variable. In summary, each <*interact*> statement is converted into BPEL code that performs the following:

- Serialize the xBPEL *promptData* and *replyData* variables in the <*interact*> construct and assign them to corresponding parts of the IC input message;
- Invoke IC operation for engaging human partner instances;
- De-serialize the message returned from the IC and assign it to the xBPEL *replyData* variable.

People-to-people constructs (<*notify*>, <*converse*>) are first converted to suitable <*interact*> constructs, which are then further processed as described above to generate standard BPEL statements. We provide an example translation of the <*converse*> statement into appropriate <*interact*> statements.

The <*converse*> statement:

```
<converse name="travelRequestApproval"
      sender="travelRequester"
      receiver="manager"
      promptData="$Please fill out the Travel Request Form"
      senderData="travelRequestForm"
      receiverData="approvalForm" />
```

Generated <*interact*> statements:

```
<interact name="submitTravelRequest"
      humanPartner="travelRequester"
      promptData="$Please fill out the Travel Request Form"
      replyData="travelRequestForm" />

<interact name="getApproval"
      humanPartner="manager"
      promptData="travelRequestForm"
      replyData="approvalForm" />

<interact name="notifyDecision"
      humanPartner="travelRequester"
      promptData="approvalForm"
      replyData="NULL" />
```

5. SYSTEM COMPONENTS

An important component in PerCollab is the BPEL Engine. The BPEL Engine executes the business process and engages human partners and Web services through various forms of exchanges. The Interaction Controller forms an intermediary to support the pervasive engagement of human participants in the business process. Other components of PerCollab are the extensible set of Modality Adapters, the xBPEL Translator and the Context Service. All communication messages are defined as WSDL messages. We discuss various system components in the following subsections.

5.1 INTERACTION CONTROLLER

The IC receives specification of individual staff activities from the BPEL Engine and forwards responses from human partners back to the engine. A staff activity specification contains information about the human partner instance intended to carry out the activity and the relevant messages. The IC exports itself as a Web service. Hence its invocation is no different from any regular Web service and does not necessitate any changes to the BPEL Engine. Upon receiving a staff activity specification, the IC obtains context information of the partner instance from the Context Service and determines an appropriate collaboration modality for the partner instance. It uses an Address Book to look up the modality-specific address (e.g., telephone number, email address, IM identifier) based on the user name. It then establishes communication with the corresponding Modality Adapter and supplies it with all the information regarding the staff activity. Communication is either notification-based (for one-way activities) or request-response based (for two-way activities). For request-response based communication, the IC also provides the Modality Adapter with the message format representing the reply desired.

We have experimented with two communication paradigms between the IC and the Modality Adapters. In a synchronous communication paradigm, the IC opens a communication session with a Modality Adapter and blocks until the communication has been completed and the reply received. This paradigm entails a multi-threaded structure of the IC. In an asynchronous communication paradigm, the IC communicates staff activity information to a Modality Adapter via events. The Modality Adapter later on establishes a callback to the IC returning the response from the partner instance.

5.2 CONTEXT SERVICE

The Context Service, described in detail in a separate publication [17], allows context-aware applications to obtain user context information without having to worry about the details of context derivation and context management. It supports both synchronous query and asynchronous callback context functions, and allows for easy incorporation of new types of context data into the Context Service. The Context Service provides both dynamic user context information and static user preferences. Dynamic context information currently available from the Context Service includes IM online status, activities and contact means derived from calendar entries, desktop activities, as well as user location reported from a variety of sources such as cellular providers, wireless LANs, GPS devices and RIM blackberry devices. The static user preferences include those used to determine the appropriate collaboration modality for a mobile user. Such preferences are represented as rules. Each rule specifies the modalities that may be used under a particular condition. The rule condition is in terms of the user's dynamic context variables such as location and activity and static attributes such as the identity of the corresponding party. Each rule is optionally associated with a priority value to help resolving conflicts between rules.

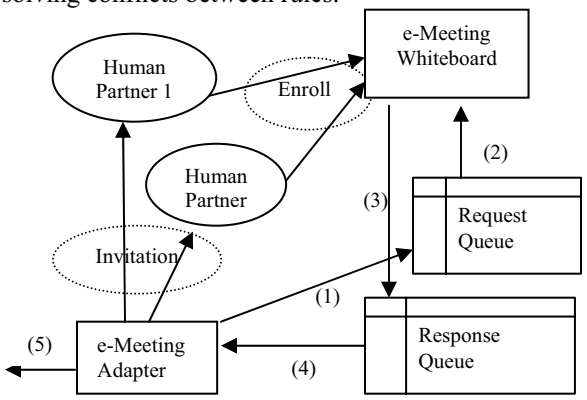


Figure 4: Operation of the e-Meeting modality adapter

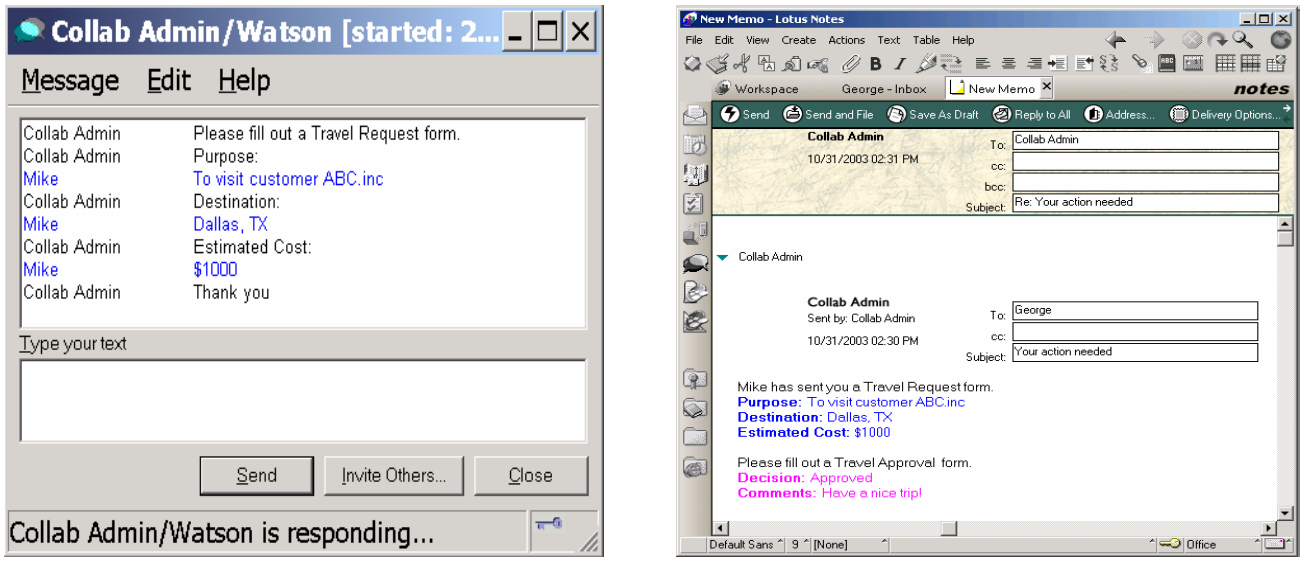
5.3 MODALITY ADAPTERS

Modality Adapters allow disparate collaboration mechanisms to be plugged into our system in an extensible manner. They expose a uniform interface to the Interaction Controller and encapsulate the details of invoking individual collaboration modalities. A Modality Adapter performs three kinds of functions. (1) It interacts with a particular modality server, initiating and terminating modality-specific connections to human partner instances as necessary; (2) It pushes staff activities to partner instances and funnels communication between the IC and partner instances. It further masks disconnections and retransmissions during the communication; (3) It interprets WSDL messages from the IC and presents them to partner instances in a modality-appropriate manner. It also constructs WSDL messages based on modality-specific input from partner instances.

We distinguish three types of collaboration modalities: connection-oriented, connectionless, and space-sharing. Connection-oriented modalities support two-way, synchronous collaboration. Examples of such modalities are instant messaging and cell phones. Connectionless modalities support one-way, asynchronous collaboration. Examples include email and SMS. Space-sharing modalities, such as e-meetings, bring all the partner instances of a business process into one shared discussion space (e.g., an electronic whiteboard). It then uses this shared discussion space to execute staff activities. While the first two types of modalities engage human partners on a per-interaction basis, the nature of space-sharing modalities dictate that they are most appropriate as access media for the entire duration of the business process.

We have implemented Modality Adapters with different designs based on the type of the modality. Adapters for connection-oriented modalities employ a dispatcher and a collection of worker threads. Each worker thread maintains one connection session. A connection is established only if the corresponding partner instance is online or available on the modality server. Adapters for connectionless modalities are based on a state-machine model with state transitions triggered by communication messages from the partner instance. No connection setup and termination are needed in this case as the partner instance does not have to be connected for the communication to take place.

Adapters for space-sharing modalities require another design. We illustrate this using the example of an e-meeting. To bring all human participants to the e-meeting, we first use a context-sensitive modality (e.g., instant messaging, email) to send an 'invite' message to them, giving the address of the e-meeting. Subsequently, all invitees enroll themselves in the e-meeting. To schedule and execute all staff activities that make up a business process, a hidden e-meeting attendee is used to control what goes onto the electronic whiteboard. The e-meeting Modality Adapter and the whiteboard controller coordinate with each other using a deadlock-free protocol with two shared queues: a request queue and a response queue. As shown in Figure 4, the e-meeting Modality Adapter receives specification for the next staff activity from the IC and places it in the request queue (Step 1). The whiteboard controller is awaken by new items in the request queue and engages the correct human partner (Step 2). The whiteboard controller performs all necessary



(A) (B)

Figure 5: Travel request approval through instant messaging

message conversion, collects response from the partner instance and places it in the response queue (Step 3). The e-meeting Modality Adapter retrieves this result, sends it to the IC and awaits further activities (step 5). Since the e-meeting adapter may be involved in multiple meeting sessions simultaneously, data in the request and response queues are properly tagged with session identifiers.

6. Demonstration of Concept

To validate our design, we have implemented a prototype of PerCollab. Our implementation is in Java and runs on WebSphere Application Server (WAS) version 5.0. It uses a standard BPEL engine from IBM's AlphaWorks [16], the Context Service developed in an earlier project [17], and the newly developed Interaction Controller and the xBPEL Translator. It currently supports the collaboration modalities of email, instant messaging and e-meeting. We use Sametime 3.0 Client Toolkit [19] and Lotus Notes 6.0 [20] to communicate with the corresponding modality servers. We have tested our system using a standalone driver application. The application instantiates various business processes by calling their Web service interfaces, which were generated automatically by the xBPEL Translator.

We use the example of the travel request approval business process to demonstrate the workings of our implementation. The xBPEL policy for this process was given in Figure 3. The role players in the demonstration are George, the manager, and Michael, the travel requester. The BPEL process itself is represented by ID *Collab Administrator*. We demonstrate the operations with modalities of instant messaging and email.

The screen shots in Figure 5 show the collaboration being carried out through instant messaging and email. The business process on being instantiated contacts Michael instructing him to fill up a Travel Request Form (Figure 5A). George then receives a notification of Michael's request and is instructed to fill up the Approval Form (Figure 5B). Finally, Michael receives a notification regarding acceptance or rejection of his request.

7. Related Work

Related work ranges from workflow systems, orchestration formalisms for business processes, peer-to-peer collaboration platforms, to unified communication frameworks.

Existing workflow systems [3,4,5,6,7] engage human participants through workplace-type user interfaces. Tools like Websphere Process Choreographer [3] and Dragonfly [4] focus on the integration of a wide variety of services and components into the workflow. Human participants are required to poll their desktop-based workplaces to claim and accomplish their staff activities. In comparison, our PerCollab system pushes staff activities to human participants via an appropriate communication mechanism. It allows people to participate in business processes in a more ubiquitous, flexible and user-friendly manner. Various formalisms have been developed for modeling business processes [3,13,14]. Because the Web

services framework has shown great promise as a standard platform for enterprise application integration, there have been a lot of recent interests in defining languages for orchestrating Web services. The Web Services Flow Language (WSFL) [10] and XLANG [11] are two earlier efforts from IBM and Microsoft respectively. BPEL combines the two and is emerging as an industry standard. We augmented BPEL with support for human participants.

The last few years have seen a proliferation of collaboration technologies, including software systems like email, instant messaging, e-meetings, and discussion threads, as well as devices like cell phones and pagers. These tools support either synchronous or asynchronous peer-to-peer collaboration, but they don't enforce any coordination policies or structures. PerCollab adds process support to these tools by using a BPEL engine to orchestrate the exchanges between people.

A number of projects have addressed the issue of personal mobility to support unified communication. These include the Mobile People Architecture [21], Universal Inbox [22], and our own Mercury system [25]. These projects provide an extensible framework for enabling communication across heterogeneous end-points and route communication to a convenient callee device based on user preferences. Still, the communication supported is ad hoc and unstructured. The key aspect that sets this work apart is the additional orchestration support we have integrated.

8. CONCLUSIONS

We have presented the design and implementation of the PerCollab system that effectively integrates workflow technology and ad hoc collaboration tools. PerCollab enables human participants to engage in business processes anytime and anywhere, using a wide range of collaboration mechanisms. It proactively pushes staff activities to human partners through a modality that is sensitive to the user context. It adds process support to peer-to-peer collaboration tools, making them interoperable with other services and applications. We leveraged BPEL as the underlying formalism for defining business processes and introduced a small set of constructs into BPEL to support human partners. Our system employs the Interaction Controller as a proxy for human participants. It currently integrates the modalities of email, instant messaging and e-meeting.

REFERENCES

[1] Business Process Execution Language for Web Services (BPEL), 2002. *http://www106.ibm.com/developerworks/ Webservices /library/ws-bpel/*

[2] N. R. Jennings and M. Wooldridge. Agent-oriented software engineering. *Handbook of Agent Technology (ed. J. Bradshaw). AAAI/MIT Press, 2000.*

[3] P. Grefen, K. Aberer, Y. Hoffner, and H. Ludwig. CrossFlow: Cross-organizational Workflow Management in Dynamic Virtual Enterprises. *International Journal of Computer Systems, Science, and Engineering, 15(5):277--290, 2001.*

[4] IBM Websphere Process Choreographer. *http://www7b.software.ibm.com/wsdd/zones/was/wpc.html*

[5] Dragon Fly Workflow Engine. *http://www.dragonflysoftware.com.au*

[6] M. Merz, B. Liberman, and W. Lamersdorf. Using Mobile Agents to Support Interorganizational Workflow-Management. *International Journal on Applied Artificial Intelligence, 11(6):551–572, 1997.*

[7] M. Merz, B. Liberman, and W. Lamersdorf. Crossing Organisational Boundaries with Mobile Agents in Electronic Service Markets. *Integrated Computer-Aided Engineering, 6(2):91–104, 1999*

[8] A. Reuter and F. Schwenkreis. Contracts - a low-level mechanism for building general purpose workflow management-systems. Data Engineering Bulletin, 18(1):4–10, 1995.

[9] Microsoft NetMeeting Platform. *http://www.microsoft.com/windows/NetMeeting*

[10] Erik Christensen, Francisco Curbera, Greg Meredith, and Sanjiva Weerawarana. Web Services Definition Language 1.1. *Technical report, W3C, 2001.* available at *http://www.w3c.org/TR/wsdl.*

[11] F. Leymann. Web Services Flow Language (WSFL). *white paper. 2001. http://www- 3.ibm.com/software/solutions/ Webservices/pdf/WSFL.pdf*

[12] S. Thatte: XLANG: Web Services for Business Process Design, Microsoft Corporation, 2001. *http://www.gotdotnet.com/team/xml_wsspecs/xlang-c/default.htm*.

[13] W.M.P. van der Aalst. Petri-net-based Workflow Management Software. *Proceedings of the NSF Workshop on Workflow and Process Automation in Information Systems: State of the Art and Future Directions, Athens, Georgia, May 1996.*

[14] A. Tripathi, T. Ahmed, and R. Kumar. Specification of Secure Distributed Collaboration Systems. *In Proceedings of International Symposium on Autonomous Distributed Systems (ISADS 2003), April 2003.*

[15] J. Clark and S. DeRose. XML path language (XPath) version 1.0. *W3C Working Draft, July 1999. http://www.w3.org/TR/WD-xpath-19990709.*

[16] BPEL Engine. *http://alphaworks.ibm.com/tech/bpws4j*

[17] H Lei, D. Sow, J. Davis II, G. Banaduth and M. Ebling. The Design and Application of a Context Service. *ACM Mobile Computing and Communications Review (MC2R), 6(4), October 2002.*

[18] Eclipse Project by IBM. *http://www.eclipse.org*

[19] Real-time collaboration with Lotus Sametime. *Lotus Development Corporation White Paper* (2001). *http://www.lotus.com/sametime*

[20] N. Gandhi, S. Parekh, J. Hellerstein, and D. Tilbury.Feedback control of a lotus notes server: Modeling and control design. *American Control Conference, 2001.*

[21] M. Roussopoulos, P. Maniatis, E. Swierk, K. Lai, G. Appenzeller and M. Baker. Personal-level Routing in Mobile People Architecture. *Proceedings of the USENIX Symposium on Internet Technologies and System. October 1999.*

[22] B. Raman, R. Katz and A. Joseph. Universal Inbox: Providing Extensible Personal Mobility and Service Mobility in an Integrated Communication Network. *Proceedings of the Third IEEE Workshop on Mobile Computing Systems and Applications. Monterey. CA, December 2000.*

[23] H. Wang et. Al. ICEBERG: An Internet-core Network Architecture for Integrated Communications. *IEEE Personal Communications. 2000.*

[24] D. Hollinsworth. The Workflow Reference Model. Technical Report TC00-1003, Workflow Management Coalition, *http://www.aiai.ed.ac.uk/WfMC/.*

[25] H. Lei and A. Ranganathan. Context-Aware Unified Communication. *Proceedings of the 2004 IEEE International Conference on Mobile Data Management, Berkeley, CA, January 2004.*