Extending the Reach of Business Processes

Dipanjan Chakraborty, University of Maryland, Baltimore County
Hui Lei, IBM T.J. Watson Research Center

A business process is a systematic set of activities by which an enterprise conducts its affairs. For example, a typical service parts business receives parts orders from customers, determines the appropriate distribution center to fill those orders, ships the parts to the customers, orders replacement parts from suppliers when inventory falls below a certain threshold, and periodically optimizes the inventory level.

Computing technology has made managing such activities much easier. Enterprise systems maintain knowledge of ongoing business processes and engage people, Web services, software agents, and other entities to execute tasks as needed. Contemporary business processes include product planning, software design, post-sale services, supply-chain monitoring, travel request approval, and job candidate evaluation.

Conventional business-process management systems rely on a workplace-based staff that accesses enterprise databases using high-end desktop computers. However, these systems are inefficient because they place the burden on users to periodically "pull" tasks. They also do not support direct synchronous communication between users.

Various technologies—including pagers, cell phones, pocket PCs, instant messaging (IM), and the short message service (SMS)—have emerged that people can use to communicate even when they are on the move or far away. Many such devices support synchronous communication as well as proactively "pushing" messages to users. However, these devices have no mechanism to control or structure the information that users are exchanging, and they are not integrated with business processes based on workplaces.

To address these problems, we have designed and implemented PerCollab, a middleware system that facilitates structured collaboration between various communication devices for business processes and pushes tasks to users. Because people typically use a subset of available mobile devices at a given time, one of the system's primary functions is to dynamically determine the most appropriate device based on the user's current context.

PerCollab integrates communication devices with business processes and pushes tasks to users.

SYSTEM OVERVIEW

Figure 1 illustrates Percollab's basic architecture. The Business Process Execution Language serves as the system's underlying process modeling formalism. Because BPEL assumes that all business partners are abstracted as Web services, we have introduced additional constructs to represent human users and define human interaction patterns.

Each business process defined in our xBPEL extension has an external interface defined in the Web Service Definition Language (WSDL) that applications use to initiate the process. At invocation time, the business process accepts configuration parameters such as actual human participants and acceptable communication devices. Because all communication messages exchanged in PerCollab are defined in WSDL, the system components are interoperable with other Web services.

Engine and translator

The driving component in PerCollab is the BPEL engine, which determines the list of required business tasks and the order in which the system performs these tasks based on the process definition. The engine executes human tasks by dispatching them to the corresponding participants via the interaction controller. In addition to human users, Web services can act as partners or task consumers of the business process; the BPEL engine communicates with these Web services directly.

The xBPEL translator converts process definitions in xBPEL to those in standard BPEL. Apart from generating BPEL policies, the translator generates the required WSDL description of the business process that invoking applications use to start the business process.

Interaction controller

The BPEL engine sends all human tasks to the interaction controller, which delegates them to the appropriate communication device for each user and sends the results back to the engine. The interaction controller exports itself as a Web service, thereby facilitating its invocation using standard Web service interfaces.

When the interaction controller receives a task, it obtains context-specific information about the intended human participant and determines the proper device or modality to use. The controller uses an address-book service to obtain the person's device-specific address such as cell phone number or e-mail address and then communicates the activity to the device-specific modality adapter.

Tasks are either *notification-based* for one-way activities or *request-response-based* for two-way activities. For two-way activities, the interaction controller provides the modality adapter with the desired reply's message format.

Communication with the modality adapter can take one of two forms. A *blocking* call waits for the reply from the human participant, while a *non-blocking* call uses events and callback mechanisms to convey the reply to the BPEL engine. The interaction controller uses blocking calls for IM, cell phones, and other connection-oriented modalities while nonblocking calls are for connectionless modalities such as e-mail.

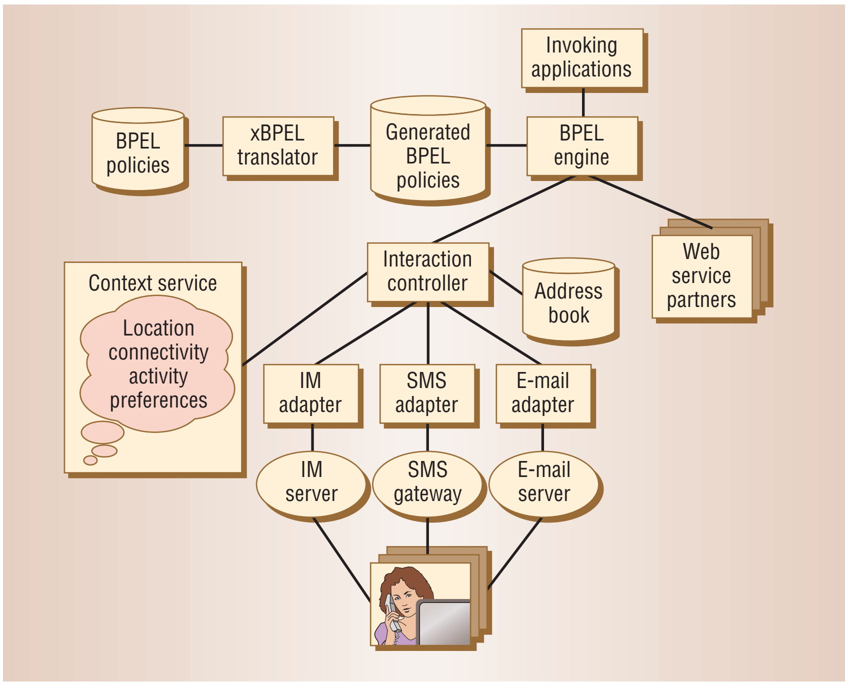


Figure 1. PerCollab system architecture. The Business Process Execution Language (BPEL) serves as the underlying process modeling formalism. The system currently integrates instant messaging (IM), short message service (SMS), and e-mail modalities.

Context service

The context service is responsible for collecting and managing contextual information about the human participants, including the user's preferred communication device in different situations. It uses dynamic contextual data such as IM online status and calendar entries as well as static, user-specified prioritization to determine the appropriate device or modality. The context service easily incorporates new context data and supports both synchronous queries and asynchronous callback functions.

Modality adapters

These plug-in components are responsible for engaging human participants via a specific device and transcoding the messages to an appropriate format. Each type of adapter caters to a specific class of devices—such as cell phones, pagers, IM, and SMS—and implements a uniform interface.

Modality adapters

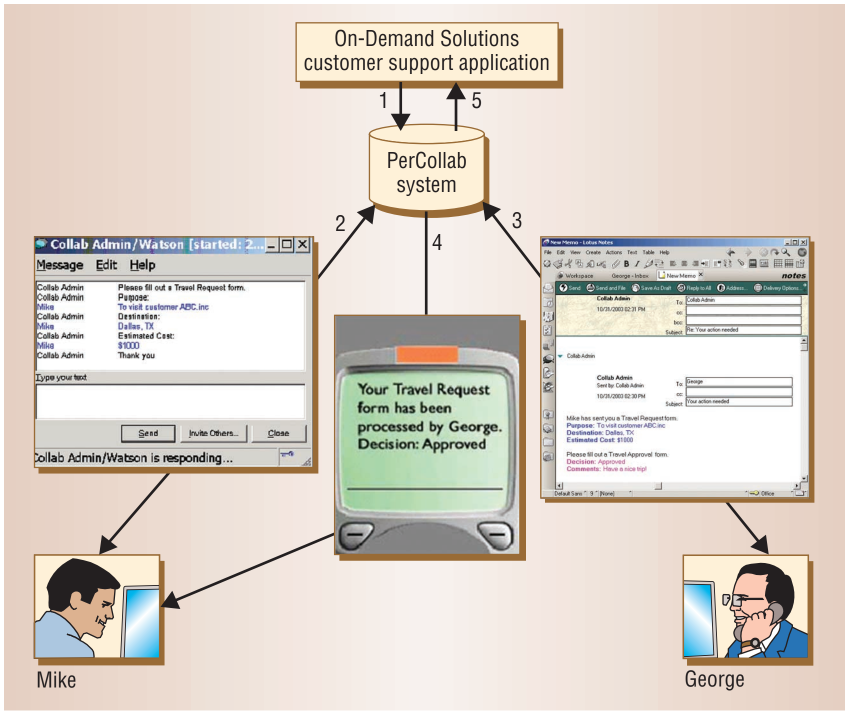
- use the modality-specific server to establish a connection to the user;
- present tasks in the modality-specific format, obtain the reply from the user, and return it in WSDL format to the interaction controller; and
- manage the connection with the device, detect disconnections, and ensure reliable message delivery through retransmissions.

Modality adapters can be classified according to their mode of operation. *Connection-oriented* modalities such as cell phones and IM maintain a consistent connection per interaction session; the connection closes only after the entire interaction session is over.

Connectionless modalities such as e-mail are state dependent and event driven. The adapter models tasks as events and sends them to the users, who finish the tasks and return the reply event to the adapter.

Space-sharing modalities such as e-meetings require all participating users to share a whiteboard or common workspace. Their design requires using context-appropriate devices to send "invite" messages to users to join the shared space. Once all users have joined, the modality adapters channel tasks to the shared space and send completed tasks back to the interaction controller.

PerCollab uses a deadlock-free, queue-based scheduling algorithm to channel task inflow and outflow. Each



Collab Admin Please fill out a Travel Request form.
Collab Admin Purpose:
Mike To visit customer ABC.inc
Collab Admin Destination:
Mike Dallas, TX
Collab Admin Estimated Cost:
Mike $1000
Collab Admin Thank you

Collab Admin/Watson is responding...

Your Travel Request form has been processed by George. Decision: Approved

Collab Admin
10/31/2003 02:31 PM
To: Collab Admin
cc:
bcc:
Subject: Re: Your action needed

Collab Admin

Collab Admin
Sent by: Collab Admin
10/31/2003 02:30 PM
To: George
cc:
Subject: Your action needed

Mike has sent you a Travel Request form.
Purpose: To visit customer ABC.inc
Destination: Dallas, TX
Estimated Cost: $1000

Please fill out a Travel Approval form.
Decision: Approved
Comments: Have a nice trip!

Figure 2. Travel request approval process. At each step, PerCollab selects a communication device based on dynamic user context and prepares the messages in a device-appropriate way.

task contains user identifiers to indicate who should perform it.

IMPLEMENTATION

Our PerCollab prototype integrates e-mail, Sametime IM, and e-meeting modalities. It uses the BPEL engine from IBM's alphaWorks; the xBPEL translator, interaction controller, context service, and modality adapters are in-house developments. PerCollab supports cross-modality interaction—for example, one participant can use IM while another participant simultaneously uses a telephone.

Figure 2 illustrates PerCollab's functionality through a hypothetical travel request approval involving Mike, a customer service technician, and George, his manager. At each step, the system selects a communication device based on dynamic user context and prepares the messages in a device-appropriate way.

In Step 1, the On-Demand Solutions (ODS) customer support application determines that Mike, who is currently logged on to the company's IM system, should be dispatched to meet with a client.

In Step 2, PerCollab instantiates the approval process by prompting Mike to complete a travel request form. The different fields of the form such as purpose, destination, and cost estimate appear as individual messages so that Mike can fill them out one by one.

The process then calls for George's approval, but he is in a meeting and does not want to be interrupted. Thus, in Step 3, PerCollab sends George an e-mail message requesting him to review Mike's travel request form and to fill out the attached travel approval form. George finds the message in his mailbox after returning from the meeting. He grants Mike's request by completing the travel approval form and including it in an e-mail reply to PerCollab.

Once George has replied, PerCollab notifies Mike of the decision. In Step 4, PerCollab determines that Mike has logged off IM and thus sends a short message to his SMS-enabled cell phone informing him that George has granted his travel request.

Finally, in Step 5, PerCollab exits the travel request approval process and returns to the calling ODS application, which then continues.

By integrating multiple communication devices with workflow systems such as BPEL, PerCollab extends the reach of business processes to almost anywhere. It improves business efficiency by proactively pushing tasks to users, enhances user experiences by selecting the most convenient device based on dynamic context information, and fosters collaboration by imposing coordination policies and structure. ■

***Dipanjan Chakraborty** is a PhD student and research assistant in the Department of Computer Science & Electrical Engineering at the University of Maryland, Baltimore County. Contact him at dchakr1@csee.umbc.edu.*

***Hui Lei** is a research staff member at the IBM T.J. Watson Research Center. Contact him at hlei@us.ibm.com or visit his Web page at www.research.ibm.com/people/h/hlei.*

Editor: Upkar Varshney, Department of CIS, Georgia State University, Atlanta GA 30002-4015; uvarshney@gsu.edu