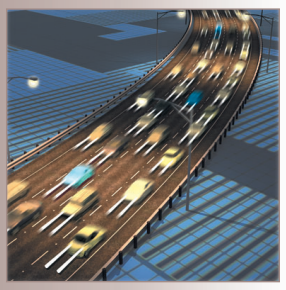
IEEE Internet Computing
www.computer.org/internet

Unifying Human and Software Services in Web-Scale Collaborations

Daniel Schall, Hong-Linh Truong, and Schahram Dustdar

Vol. 12, No. 3
May/June 2008



Editors: M. Brian Blake • *mb7@georgetown.edu*
Michael N. Huhns • *huhns@sc.edu*

Unifying Human and Software Services in Web-Scale Collaborations

As collaborative Web-based platforms evolve into service-oriented architectures, they promote composite and user-enriched services. In such platforms, the collaborations typically involve both humans and software services, thus creating highly dynamic and complex interactions. However, today's collaboration tools don't let humans specify different interaction interfaces (services), which can be reused in various collaborations. Furthermore, humans need more ways to indicate their availability and desire to participate in collaborations. The Human-Provided Services (HPS) framework lets people manage their interactions and seamlessly integrate their capabilities into Web-scale workflows as services. It unifies humans and software services and supports ad hoc and process-centric collaborations.

Daniel Schall, Hong-Linh Truong, and Schahram Dustdar
Vienna University of Technology

Web services have paved the way for a new type of collaborative system. Services let us design collaborative systems in a modular way in a distributed environment, adhering to standard interfaces — using, for example, the Web Services Description Language (WSDL).[1] Users can create collaborative features by (re)using and composing Web services. Services already play an important role in fulfilling organizations' business objectives because process stakeholders can design, implement, and execute business processes using Web services as well as languages such as the Business Process Execution Language (BPEL). Services have started exploiting the Web and are increasingly found in Web-scale collaborations. Web services are tools that users and developers can reuse in various applications by exposing well-defined interfaces and APIs.

The spectrum of collaboration ranges from *process centric* to *ad hoc* collaboration models.[2] Process-centric collaboration defines process models and follows a top-down approach. The business analyst or process architect must fully understand the processes before modeling and then enacting (instantiating) them. Such models' reusability is generally high, because we can apply process models several times. However,

 1089-7801/08/$25.00

flexibility is rather limited; if changes occur (such as exceptions), process architects have to remodel the process. On the other hand, ad hoc collaboration (for example, situations in which people or businesses must act spontaneously and creatively) follows a bottom-up approach. It's more flexible but less reusable, because many aspects depend on the actual players (that is, humans) involved in the process (see Figure 1).

However, Web-scale collaborations demand a flexible yet reusable approach because they might involve numerous people and software services. Here, we introduce Human-Provided Services, which you can use in ad hoc or process-centric collaborations. The HPS framework helps integrate humans into service-oriented infrastructures, thus promoting reusability and flexibility.

Web 2.0's Collaboration Landscape

The Web 2.0 paradigm encourages users to collaborate and share knowledge and information, so the Web is no longer a "read-only" information repository. Consider Figure 2a, in which User A publishes Web content using open service-oriented applications. Other Web users can then consume, aggregate, or filter the content. However, because the depicted scenario relies purely on ad hoc interactions, users can't apply the same procedure in other collaborations (for example, to share content including documents, videos, and photos). In addition, the collaboration isn't structured because there's no interaction link between the users. This makes it difficult — if not impossible — to manage interactions that might span multiple users and services.

Figure 2b shows a process-centric collaboration involving human actors (depicted as *human activity* in the process model). An example of such collaboration might be to model human interactions in BPEL processes as *BPEL4People activity*. However, the applicability of such models in Web-scale collaborations is rather limited because you can't model emerging interactions between humans and services in advance.

Opportunistic service composition is the trade-off in loosely structured collaborations (see Figure 1); you lose high reusability of compositions (processes) but gain flexibility in collaboration. Web-scale collaboration demands the composition of complex systems, which comprise not only interactions between software services but also humans as parts of flexible compositions.

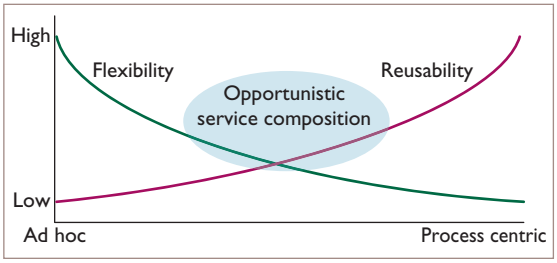


Figure 1. Flexibility vs. reusability in collaboration. Opportunistic service composition represents the trade-off in loosely structured collaborations.

Motivating Use Cases

Consider the following motivating scenarios detailing the problems arising in collaborations involving software services and humans.

Ad Hoc Contribution Requests

In Figure 2a, User A records a video and posts it on the Web. However, current platforms don't let consumers (User B) actively find available users who can contribute to collaborations by producing the desired content on demand.

In particular, users should be able to find any person who can deliver the desired content using whichever platform (service) has been chosen to host the Web content. This use case depends on the activity to be performed and the involved services but not on the platform being used to share or host the content.

User-Defined Processes

Continuing the previous use case, the collaboration might involve numerous services and people. For example, a (software) service should automatically check the input User A receives (for example, for file format compatibility) and convert it into a suitable format if needed. The requester can then check whether the provided contribution needs to be revised or re-recorded.

We observe the case in which interactions interleave tasks that humans and software services perform. However, current systems don't address reusability aspects of loosely structured processes in such collaboration scenarios, which would let users (requesters) manage interactions involving people as well as software services. This scenario targets opportunistic service composition comprising human and software services.

Interactions with Formalized Processes

It becomes increasingly important to enable interactions between business processes and human actors, if human input is required in

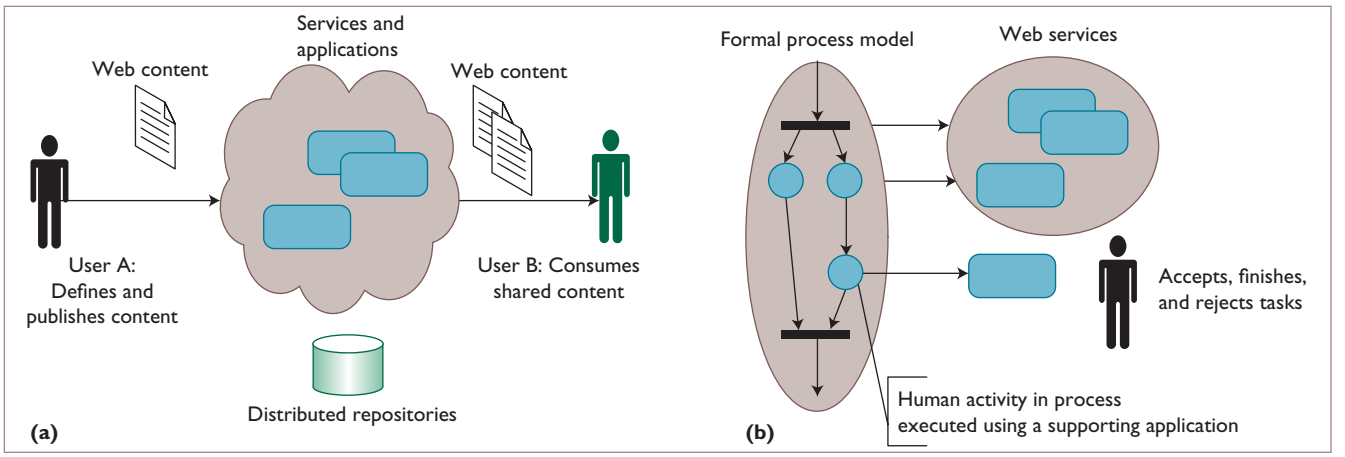


Figure 2. Web-scale collaboration. We can see both (a) ad hoc and (b) process-centric collaboration models.

a process (Figure 2b). However, people are increasingly on the move using different (mobile) devices for collaboration. So, we must consider mobility aspects, such as the location and limited processing power of a user's mobile devices, and we must adjust interactions according to the user's context. In particular, processes must be able to find and select the right person available for performing certain tasks, whereas humans involved in interactions with business processes must be able to define (interaction) rules to deal with requests — for example, to automatically pre-process certain requests.

However, current systems can't cope with human-process interactions that scale to the Web. They can't find humans participating in Web-scale interactions with processes or manage interactions involving multiple people.

HPS in Web-Scale Collaborations

We introduce HPS because current systems don't sufficiently address the challenges and problems presented in the motivating use cases (see the "Related Work in Collaboration Systems" sidebar). Here, we present the applicability of HPS in Web-scale collaborations and introduce a framework to embrace the integration of human capabilities and interactions in service-oriented collaborations and Web-scale workflows.

The HPS framework lets people supply services based on their skills and expertise. HPSs act as interaction interfaces toward humans, letting users define various HPSs for different collaborative activities indicating their ability (and willingness) to participate in ad hoc as well as process-centric collaborations. The users can manage their interactions, which might span various platforms and services. Human actors benefit from HPSs because they can reuse different services in various collaborations (such as in different workflows), thus fostering the reusability of human capabilities. Moreover, HPSs can increase flexibility in collaborations because they let human actors provide services that can address problems that software services alone can't solve.

The Framework

However, this novel blend of service-oriented architectures requires a new platform to let humans effectively provide services and to efficiently deal with interactions through HPSs. Current service-oriented platforms can't sustain HPSs because

- conventional service registries don't offer suitable lookup interfaces for finding them,
- current platforms can't enhance service-related information by describing the human characteristics of HPSs as needed, and
- current platforms don't address HPS interaction patterns, so they can't introduce new service (HPS) interaction patterns to let human actors efficiently deal with requests.

Figure 3 shows the steps that the HPS framework takes to address these challenges, illustrating the scenario for ad hoc contribution requests.

Step 1. Register the profile and service. Human actors define high-level collaboration activities (for example, *createReport*) using an HPS interface editor that the framework hosts. The HPS framework automatically translates these activities into low-level service interfaces described in WSDL. User profile information includes name, skills, and competency, which the HPS framework uses to enhance the discovery,

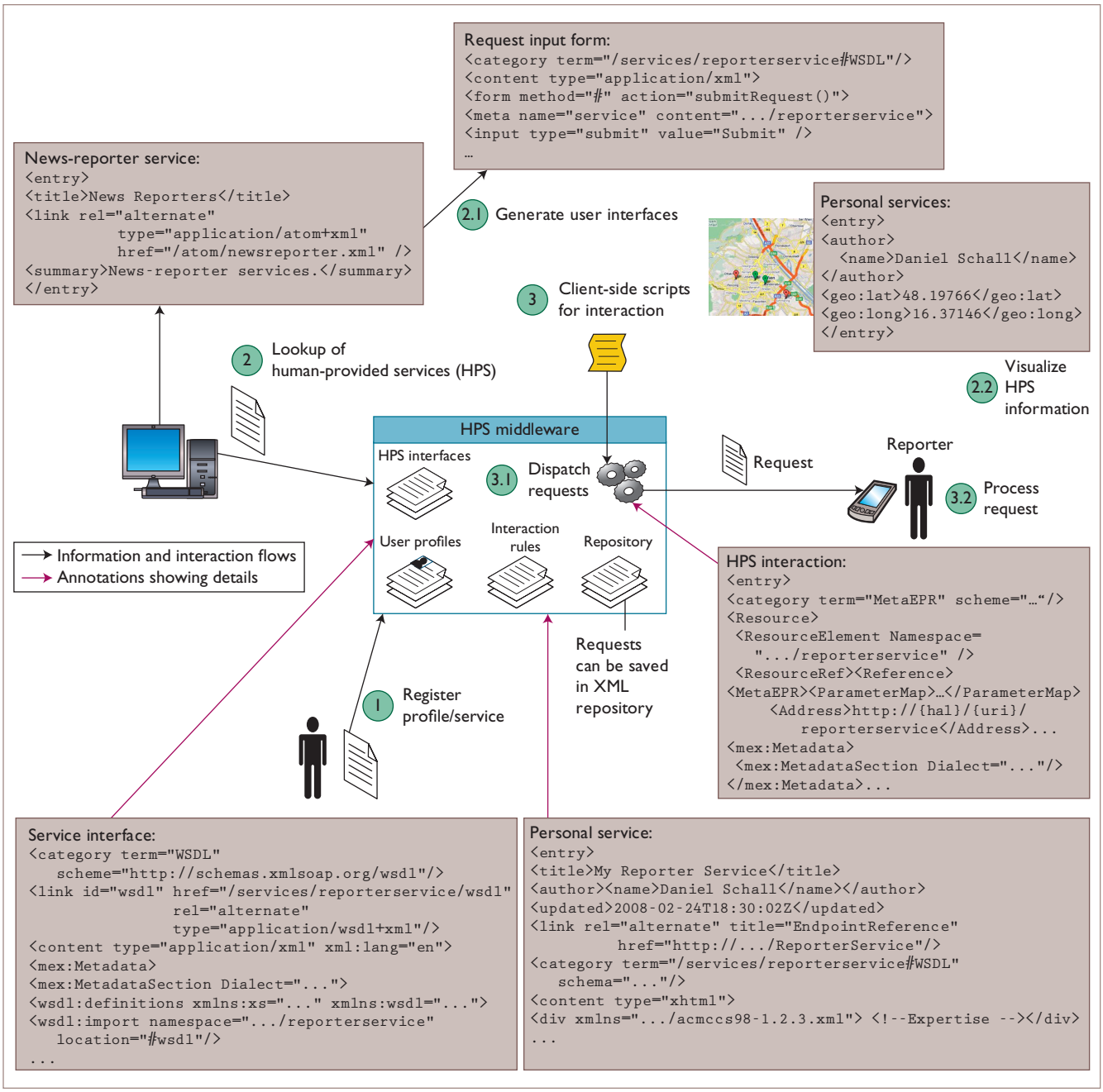


Figure 3. HPSs in Web-scale collaborations. This scenario illustrates ad hoc interactions between requesters and HPSs.

selection, and recommendation process to find the most suitable HPS. The user specifies basic personal profile information or uploads this information as a vCard file. Humans provide a service by registering it as a personal service.

The HPS scenario in Figure 3 shows an example in which humans provide *reporter services* to contribute Web content such as news reports. The middleware hosts a set of XML documents in the service registry that's managing the interface description and personal service information. So, it's easier to achieve cross-organizational collaboration because companies can share information stored in the service registry — the very foundation of Web-scale workflows. Other people who want to provide the same type of service can then reuse the service interfaces.

Figure 3 shows a snippet of the XML description of a personal service. The description contains user-related information, a reference to the service interface description, and information regarding the user's expertise rooted in

taxonomies. This information is embedded in Atom feed entries. The Atom Syndication Format is an XML language describing frequently updated content such as news. Atom feeds contain, for example, author information, links to content, and summaries. The HPS framework uses Atom feeds as a container format for WSDL documents and various content including taxonomies describing users' expertise; additional context information, such as location (<geo> tags); and category information to tag services.

The HPS framework supplies the *personal service hosting environment*, which users can download to their desktop computer or mobile devices using mobile Java technology (JavaME). This environment lets the computer or device deploy software for personal services as *gadgets*. It comprises a micro-OSGi environment (www.osgi.org/osgi_technology/), a set of tools to manage the gadgets (services), a common lightweight SOAP library, and a user-interface rendering engine displaying user interfaces described in XML.

Step 2. Look up a service. HPSs can be discovered through an interface implementing the Atom protocol model or a Web service interface. Figure 3 shows an example in which location and availability information enhance the discovery process given that requesters might want to find reporter services located in some areas of interest. The Atom lookup interface returns a feed containing a ranked list of entries comprising personal HPS information. It ranks the services based on various HPS metrics, such as skill level and user response time.

The lookup returns additional user interface rendering information — for example, XForms, which are automatically generated based on WSDL interfaces (step 2.1 in Figure 3) if human requesters attempt to interact with HPSs. XForms are a forms technology expressed in XML that describe user interfaces in a device-independent way. For example, the lookup returns interface rendering information, which can be embedded in markers of a geographical map (step 2.2).

Step 3. Interact with HPS. Ajax scripts can issue requests asynchronously toward the middleware platform. The middleware implements an HPS Access Layer interface (HAL) to dispatch HPS requests. HAL provides a security module to prevent unauthorized access, policy management to protect the users' privacy, and request filtering to shield HPSs from denial-of-service attacks.

HAL dispatches and routes service requests to the appropriate HPS and device. The HAL interface description is denoted in Figure 3 as HPS interaction using Web Services Resource Catalog (WS-RC) Meta-Endpoint definitions that are parameterized by HPS addressing information, such as user identifiers. (For more on the WS-RC, see www.ibm.com/developerworks/library/specification/ws-rc/index.html.) HPSs aren't always online, given that the personal service hosting environment might be deployed on mobile devices, which rely on wireless network availability and coverage. If the HPS isn't available at the time of interaction, an XML-based repository can store service requests (see HPS Middleware in Figure 3) and process them whenever the HPS is back online (step 3.2). Pending requests can be received via push- and pull-based mechanisms depending on the hosting environment's configuration. At this stage, HAL comprises request processing and routing capabilities and request filtering. Implementing security and policy management features is our current work in progress; we hope to address such implementation in the next steps.

Ad Hoc Collaboration Example

The example in Figure 3 shows an ad hoc interaction without any means for control or coordination. We can create tasks to control interactions and to share status information with the requester. Specifically, we need tasks for interactions between HPSs and processes to determine whether the HPS will process the requests. Task states include *inprogress*, *rejected*, or *finished*. Additionally, actions can be triggered automatically based on task-state changes such as sending notifications upon state changes.

Interactions with HPS might be long-running conversations comprising a multitude of messages, possibly in different formats (such as SOAP/XML, instant messaging, or email), notifications, tasks, people, and documents. The HPS middleware implements an XML-based file system, which provides access to the XML repository and querying and filtering capabilities through XQuery. To manage complex interactions, users can specify *Interaction Rules* (see Figure 3) to create loosely structured (user-defined) processes, which users can then apply and reuse in various interactions (such as interactions through services).

Related Work in Collaboration Systems

The first column in Table 1 shows features, or capabilities, which collaboration systems must support to address the needs in large-scale collaborations and workflows involving human and software services.

Human computation aims to leverage human capital in computational processes.[1,2] For example, human actors perform certain tasks in a program. Related to human computation, shown as *human-reviewed data* in Table 1, are systems such as Yahoo! Answers[3] (http://answers.yahoo.com) and Amazon Mechanical Turk (www.mturk.com), letting people claim and process tasks, which human or software requesters can issue.

BPEL4People and the WS-HumanTask specification[4] provide a design for enabling human and process interactions in Business Process Execution Language processes. Expert finder systems[5] aim to define ontologies that describe the skills and expertise of people to help others find the right person (expert) on the Web.

The main differences between the systems shown in Table 1 and Human-Provided Services are that the latter are user-defined interaction interfaces that people supply and compose for various collaborations. Compared to concepts in the BPEL-4People specification, people using the HPS framework decide which service to provide — for example, for a specific collaboration context — and manage their interactions using HPS interfaces. These interfaces let human and process requesters interact with HPSs, whereas BPEL4People specifies how the process architect can involve people in designed processes but doesn't specify how and which services people can offer. Thus, the BPEL4People specification doesn't let people specify their contributions as services in Web-scale collaborations; however, BPEL4People-based processes can interact with HPSs. HPS follows the Web 2.0 paradigm, in which services are user-driven contributions rather than tasks tailored to specific processes.

References

1. L. Ahn, "Games with a Purpose," *Computer*, vol. 39, no. 6, 2006, pp. 92–94.
2. C. Gentry, Z. Ramzan, and S. Stubblebine, "Secure Distributed Human Computation," *Proc. 6th ACM Conf. Electronic Commerce* (EC 05), ACM Press, 2005, pp. 155–164.
3. Q. Su et al., "Internet-Scale Collection of Human-Reviewed Data," *Proc. 16th Int'l Conf. World Wide Web* (WWW 07), 2007, pp. 231–240; http://www2007.org/papers/paper461.pdf.
4. M. Amend et al., "Web Services Human Task (WS-HumanTask), Version 1.0.," Jun. 2007; http://download.boulder.ibm.com/ibmdl/pub/software/dw/specs/ws-bpel4people/WS-HumanTask_v1.pdf.
5. I. Becerra-Fernandez, "Searching for Experts on the Web: A Review of Contemporary Expertise Locator Systems," *ACM Trans. Internet Technology*, vol. 6, no. 4, 2006, pp. 333–355.

Table 1. Related collaborations systems and the features they support.

Feature	Human computation	Human-reviewed data	Human/process interactions	Expert finder systems
Human requesters		X		X
Process (software) requesters	X	X	X	
Interaction-based collaboration		No explicit collaboration link	X	
Applicability in Web-scale collaborations	X	X	Enterprise level collaboration	Ontologies describing skills
User-defined interactions			Defined by process designer	
Context-dependent discovery of humans				
Ranking and recommendation of user/service		X		X
Compositions humans and services				

Process-Centric Collaboration Example

Figure 4 employs HPS as part of a formalized process comprising interleaved human and service interactions. It shows a workflow that integrates HPS and software services to respond to emergency situations by gathering information and input from various human and software services. First, the system receives video footage from a *monitoring service* – a surveillance system that has cameras deployed to monitor certain areas of interest. *A detection service* processes the image data, detects incidents, and generates events accordingly. The *policy service* receives a stream of events and classifies the nature of certain events (for example, classifying events as suspicious activities). Events that

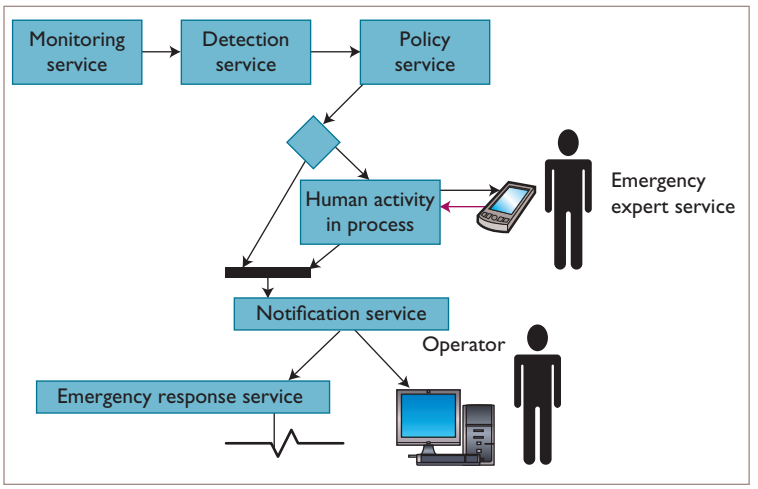


Figure 4. The workflow for an emergency scenario using HPSs. In this case, the process involves both humans and software services.

the policy service can't classify require human intervention. Classifying an emergency event constitutes an additional activity in the process, which the *emergency expert service* (that is, an HPS) performs. The process requires a human to evaluate the situation. The process accomplishes this by dynamically discovering a nearby HPS (user) who can review the situation and provide desired input for the process.

Although not explicitly shown in Figure 4, it's possible to consult multiple HPSs. The process continues and invokes a *notification service* to inform local authorities (that is, the *human operator*) about the incident. The authorities invoke the *emergency response service*, which automatically deploys an emergency response team to the emergency area. The HPS framework's contributions in this scenario are that software services and processes can discover HPSs using information available in a service registry, including users' profiles, service-specific information, and context information. In addition, humans might use mobile devices to interact with processes, which is increasingly important in today's collaborations.

The most promising direction for further HPS development is the automatic generation of service interfaces based on user skill and profile information. At this stage, the users design services, specifying collaborative activities, which the HPS framework translates into XML interface descriptions.

Furthermore, we're working on methods and algorithms for analyzing interactions to better understand complex behaviors in a mixed system of human and software services. Based on a set of HPS-related metrics, including social aspects and reputation, we rank the services to help requesters find and interact with the most suitable HPS. Because HPSs are user-driven services, we consider unexpected behavior by modeling the quality of an HPS and rewarding models, taking into account performance and reliability aspects in processing tasks and requests. HPS ranking is essential in large-scale collaborations and workflows because HPSs can be dynamically discovered, and because potentially a large number of users might provide a particular service, thus the recommendations must guide the service selection.

Additionally, we're working on improving tools for executing user-defined processes and for integrating a BPEL4People engine[3] to use HPS in BPEL processes. For more information, see www.vitalab.tuwien.ac.at/autocompwiki/index.php/Human-provided_Services.

Acknowledgments

This work has been partially supported by inContext (FP6-034718).

References

1. M.P. Papazoglou et al., "Service-Oriented Computing: State of the Art and Research Challenges" *Computer*, Nov. 2007, pp 64–71.
2. S. Dustdar, "Caramba – A Process-Aware Collaboration System Supporting Ad Hoc and Collaborative Processes in Virtual Teams," *Distributed and Parallel Databases*, vol. 15, no. 1, 2004, pp. 45–66.
3. T. Holmes, M. Vasko, and S. Dustdar, "VieBOP: Extending BPEL Engines with BPEL4People," *Proc. 16th Euromicro Int'l Conf. Parallel, Distributed and Network-Based Processing* (PDP 08), IEEE CS Press, 2008, pp. 547–555.

Daniel Schall is a research assistant and PhD student in the Distributed Systems Group at the Vienna University of Technology's Institute of Information Systems. Contact him at schall@infosys.tuwien.ac.at.

Hong-Linh Truong is a research scientist in the Distributed Systems Group at the Vienna University of Technology's Institute of Information Systems. Contact him at truong@infosys.tuwien.ac.at.

Schahram Dustdar is a full professor of computer science and head of the Distributed Systems Group at the Vienna University of Technology's Institute of Information Systems. Contact him at dustdar@infosys.tuwien.ac.at.