16th Euromicro Conference on Parallel, Distributed and Network-Based Processing

VieBOP: Extending BPEL Engines with BPEL4People

Ta'id Holmes, Martin Vasko, Schahram Dustdar
Distributed Systems Group, Institute of Information Systems
Vienna University of Technology, Vienna, Austria
{*tholmes*, *m.vasko*, *dustdar*}*@infosys.tuwien.ac.at*

Abstract

The need for integration of human interaction scenarios into BPEL processes lead to the formalisation of tasks and human roles. The specifications BPEL4People and WS-HumanTask introduce, among other definitions, a dedicated people activity that uses a task performed by a human as well as human roles that describe the relationship of people, processes and tasks. This work presents the architecture of Vienna BPEL for People (VieBOP), a new BPEL4People system that can be coupled with an arbitrary BPEL engine. We will evaluate the standards for BPEL4People and WS-HumanTask against goals as derived from the BPEL4People white paper and compare it to our work.

1. Introduction

Living in a competitive world, businesses are naturally interested in information technology supporting them for *competitive advantage* [16]. As *cooperation* becomes increasingly important for companies [10], new challenges arise for the support of business to business scenarios by information technology [19]. While enterprises already profit from the use of traditional *workflow* management systems (WfMS) [11], the business process execution language (BPEL) [20] permits formal specification of processes and enables companies to *collaborate* with each other by interacting business processes [36]. BPEL can be used for automated processes between businesses using respective services. However, the obvious scenario of a business process that depends on a person to fulfil a certain *human task* as a kind of process activity, is not covered by BPEL [3].

Web services have become widely accepted as the de–facto standard for distributed business applications [4]. They bring maximum *interoperability*, use an open and flexible architecture, and the implementation and complexity of a Web service can be hidden towards a service requestor. Layered on top of these services, BPEL, the de–facto standard for *orchestration*, formally describes processes [8]. While external activities correspond to Web services, human interactions and process activities that are related to human aspects cannot be specified using plain BPEL.

To give a simple example for human interaction we refer to an *approval* of a request at a certain stage of the process. How could such a simple interaction be realised with BPEL? As a matter of fact, the approval activity has to be designed as an *external* activity to invoke. This Web service then *has to be implemented* individually and besides receiving the Web service call, it has to notify a user and offer an interface for giving the approval before resulting a result to the process.

Thus, while BPEL glues together the logic of a process, individual solutions have to be realised when people are integrated into business processes. One may argue that it is justified to accept this circumstance as a customised and optimised user interface for end users makes sense. What to do, however, if this approval must only be given by someone from a certain *group of people*? In this case, one would be quick to suggest, a database containing data on human resource information has to be consulted. It becomes complicated from a developers point of view if a task cannot be performed as easily as an approval but has to be *undertaken* by an authorised person. While on the one hand we are interested not to give the same task to different people, we want users with administrative roles to *observe* tasks and *interact* with processes nonetheless.

Actually, these few considerations already bring us close to BPEL4People as BPEL4People focuses on *integrating* people into processes. A company may be structured in a static and hierarchical way and it may very well have business processes that fit into a system of defined roles. But as we can see this places lots of *restrictions* on a far too limited system. Each process may be disparate from others and the association of human roles may differ completely. Most interestingly: people assignment does not have to take place within the same enterprise. This is to say employers of several companies may be delegated to work within var-

0-7695-3089-3/08 $25.00 © 2008 IEEE
DOI 10.1109/PDP.2008.77

ious *inter–corporate* groups or virtual teams [9], occupying different roles respectively.

As a consequence, IBM and SAP proposed BPEL4People in [3] as an extension to BPEL, defining human participant integration pointcuts.

The lack of capabilities for describing human participants in contemporary business process standards is evident in different areas of process management. On the *modelling* level the business process modeling notation (BPMN) [22] – by the wide industry acceptance the de–facto business process modelling standard – provides no direct support for the integration of human actors in business process models. On the *executional* level BPEL assures to become the workflow standard by strong industry support and its wide acceptance in academic community, despite some drawbacks: BPEL–based workflows are not able to describe interaction scenarios with human business partners. Motivated by this shortcoming two specifications have been adopted in June 2007 that address this workflow domain and cover the integration of human actors: WS-BPEL Extension for People [2] and Web Services Human Task [1].

The focus of this work lies on the *analysis* of BPEL4People and introduces an *architectural concept* as realised in the implementation of VieBOP. Supposing a BPEL4People *layer* to be defined *on top of* BPEL, we would like to use existing BPEL engines for process deployment and execution. The motivation of our approach for an implementation thus was to design a BPEL4People system that *manages human aspects* while transparently interacting with an arbitrary BPEL engine.

This paper is structured as follows: In this section we have defined the problem of BPEL without an appropriate extension that provides descriptive support for people. The architecture of VieBOP, a BPEL4People system that enables traditional BPEL engines to handle BPEL4People processes transparently, will be introduced in Section 3. Section 4 will enumerate respective requirements for BPEL4People on a conceptual level together with ideas derived from [3] and will introduce the standards [2] and [1] by elaborating and comparing them to some of our work. Section 6 discusses the results and Section 8 gives a conclusion by referring to further work.

2. Background

The business process execution language (BPEL) is an XML[1] subset for specifying and executing business processes [20]. As interactions are realised with Web services for maximum interoperability between various heterogeneous systems, BPEL permits *orchestration* of Web services. BPEL4People, as an extension to BPEL, describes

[1]Extensible Markup Language (XML) [6]

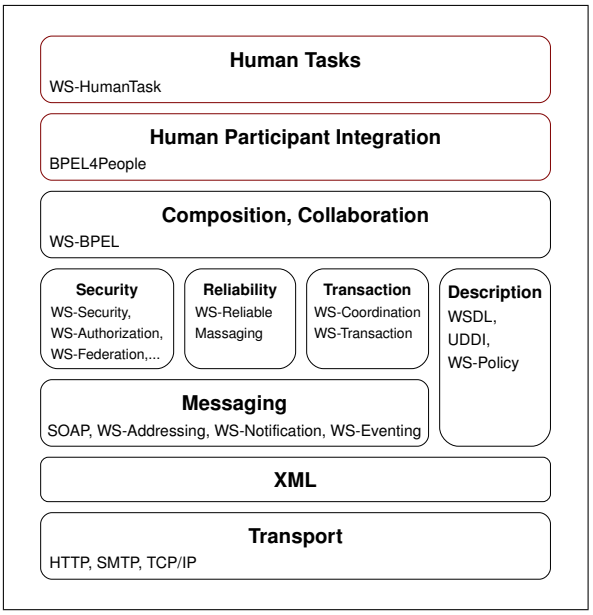

Figure 1. BPEL4People and WS-HumanTask within the Web Service Stack

scenarios where *users* are involved in business processes. Figure 1 shows BPEL4People within the Web Service stack.

Section 14 in [20] declares *extensibility* for WS-BPEL by using *foreign namespaces*. Moreover, an *extension* element can be used to indicate a BPEL engine that it must support a foreign XML namespace's WS-BPEL extension.

3. System Architecture

As BPEL4People really is an extension to BPEL, we chose to design a system that interacts with *arbitrary* BPEL engines while hosting BPEL4People information and managing BPEL processes on a *people* level. VieBOP thus *encapsulates* a traditional BPEL compliant engine transparently while offering specific interfaces to clients.

We will first have a look at the mapping of BPEL4People to BPEL and then present some components of VieBOP, that we implemented in Java using JAX-WS[2] technology. Appropriate business and data objects for and interfaces of our system have been defined[3] using XML schemata [32, 5] and WSDL[4]. In order to administrate VieBOP we developed a web application using HTML[5], CSS[6], JavaScript [26], JavaServer Pages [24], together with Java Servlet technologies. For persistence we chose a native XML database and

[2]Java API for XML - Web Services [7]
[3]`http://xml.taid.holmes.at/ns/bpel4people/`
[4]Web Services Description Language (WSDL) [29]
[5]Hyper Text Markup Language [31]
[6]Cascading Style Sheets [30]

implemented a persistence layer with *revision control* for generic *BusinessObject*s. A simplified UML[7] package diagram of VieBOP is illustrated in Figure 2.

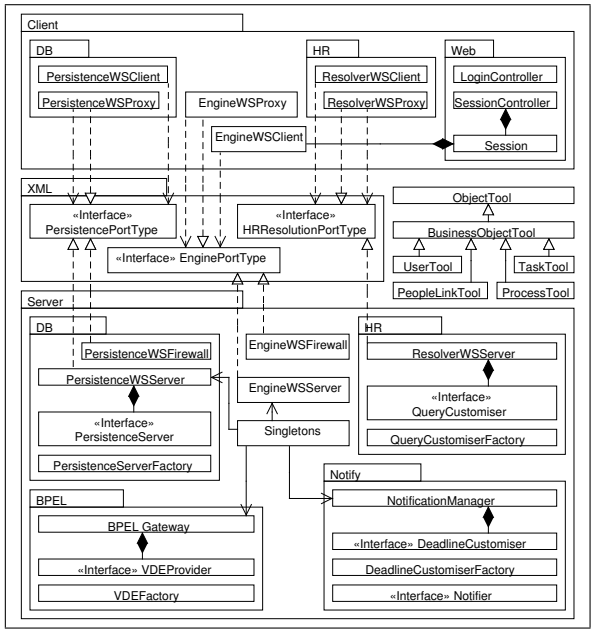

Figure 2. Package Diagram of VieBOP

3.1. Enabling BPEL engines for BPEL4People

When parsing processes containing BPEL4People elements, BPEL engines can forward deployment to VieBOP. Thus: besides direct process deployment and execution, VieBOP can also be used *reversely* by BPEL engines. A BPEL4People process is parsed by VieBOP and converted to BPEL that will be deployed on the BPEL engine. VieBOP thus maps BPEL4People to ordinary BPEL by *extracting* all BPEL4People specific data and by *transforming* people activities into ordinary BPEL activities.

3.2. Mapping BPEL4People to BPEL

BPEL4People extends BPEL by additional vocabulary that uses it's own namespace. A *people activity* from BPEL4People that participates in the process workflow as a simple activity can be transformed to a pair of invoke and receive *BPEL activities*. The *invoke* activity calls the people activity that itself calls back the *receive* activity when terminating. We kept our definition of BPEL4People simple so that a mapping of people activities suffices in order to support BPEL4People using a traditional BPEL engine

[7] Unified Modeling Language (UML) [15]

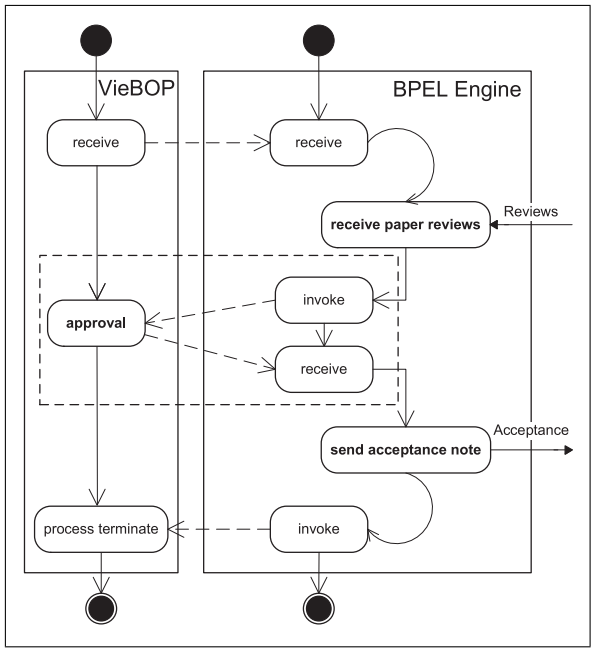

Figure 3. A BPEL4People approval process as deployed on VieBOP and an BPEL engine

for deployment and execution together with VieBOP that manages the people aspects.

Besides people activities [2] also permits to copy variables from BPEL4People to BPEL variables and vice–versa using *assign* activities. Thus: appropriate operations have to be exposed and BPEL4People fragments have to be transformed accordingly using invoke and receive activities for communication with VieBOP in order to realise the desired procedures. We can accomplish that by replacing BPEL extended elements into a set of invoke and receive activities for deployment on the BPEL engine. As with the people activity, the extension, that is hosted on VieBOP, then is transparently called by the BPEL engine.

Besides the transformation of people activities, process invocation simply is forwarded to the BPEL engine and an invoke activity, appended at the end of the process, notifies VieBOP of the terminated process. Figure 3 demonstrates how the approval example containing a people activity, that invokes a human task for the approval, will be deployed and executed both on VieBOP and the BPEL engine.

3.3. BPEL Gateway

The BPEL gateway of VieBOP is responsible for communicating with a BPEL engine. It *deploys* processes by converting BPEL4People to plain BPEL and *exposes* a Web service endpoint with operations for invoking people activities and for notifying the engine of process termina-

tion. It *receives* invocations for people activities from the BPEL server and *activates* the corresponding tasks. If a task reaches its *closed* state, the BPEL gateway invokes the corresponding *receive* activity on the BPEL process by passing the tasks output.

3.4. HR Server

This independent component hosts human resource related information and can be run individually by an organisation while offering a Web service towards the BPEL4People engine for *authentication* and *resolving* purposes. The HR server returns *user objects* for user names, people links and for successful logins. It can moreover implement individual *customisation* that can be applied during people link resolution.

Customisers Query and deadline customisers permit the customisation of people queries and deadlines. They take *parameters* as their input that can be interpreted in an individually implemented way and return *customised* people queries or deadlines. The following table shows examples for deadline parameters using a simple grammar, that a customiser would have to interpret:

key	value
owned	1d
completed	owned + 1w
completed	2010-01-02

Customisers are not found in [2] nor [1]. On the other hand it is possible to specify *expression languages* that are associated with a namespace for the *interpretation* and *evaluation* of expressions and values. By applying *namespace-prefixes* to the key names of customisation parameters it becomes possible to apply several customisers.

3.5. Notification Manager

The responsibility of a notification manager is to watch deadlines of tasks and raise *escalations* in case of a missed deadline. Apart from polling, the notification manager may directly be invoked from VieBOP for delivering notifications. It does so by consulting the HR service for people resolution. Besides sending notifications to *observers*, the notification manager also creates escalation tasks, sends out escalation notification to *escalation recipients* of the original task and manipulates the escalation tasks state in case the original task is completed. Escalation tasks are tasks that do not have corresponding people activities. They are fully managed within the manager.

4. People Integration Requirements

In order to gain an understanding of the new requirements that arise from the formal integration of people into processes, we want to summarise key human–process interaction scenarios and conclude with goals for BPEL4People. We derived the following requirements from [3]. We then introduce the two standards BPEL4People and WS-HumanTask and compare them to the identified interaction scenarios.

4.1. Human $\rightleftharpoons$ Process Interaction

Unidirectional interaction that is initiated by a *human* can be categorised as following:

Instantiation A user *instantiates* a process.

Supply Data Data can be supplied by a user for the process. This can be the *result* of a performed task as we will see as well as any *annotation* that may be interesting information for the continuation of the process. This may take place as early as when instantiating the process.

Notification is the only *unidirectional* interaction that can be performed by a *process*.

Notification A user is *notified* by the process that continues its workflow. Notification thus does not block the process.

Bidirectional interaction between a process and a human as shown in Figure 4 consists of:

Request Data A user is notified by a process and further data is required for the process execution to proceed. Upon notification the user typically will *perform* a task and *submit* data to the process that blocks for the people activity to complete.

Provide Data A task is performed by a user and results are sent back to the process. An approval, in fact, can be understood as the most simple case of a user providing data: a binary decision. Nonetheless it may be enriched by an additional annotation documenting the decision.

4.2. People Activity: a Human Task

The execution of a task, that represents a small unit of work and that can be accomplished in an *atomic* way, is delegated to a person only. A task may be equal to another task that is assigned to a different user[8] but the same task must not – and by definition actually cannot – be performed twice.

[8] see the description of the four eyes principle (4.4) for an example

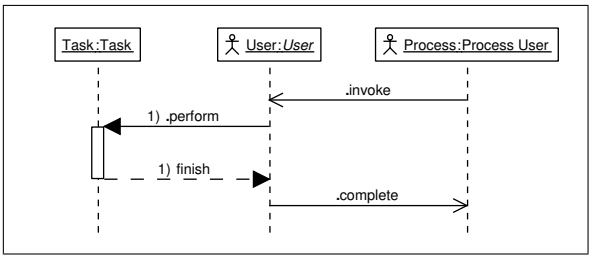

Figure 4. Sequence Diagram for invoking a People Activity

4.3. Role-based Interactions of People

In everyday life we are used to play several *roles* when interacting with our environment. For this, we do not require hierarchical structures for the role definitions of individuals. It can be observed in particular that responsibilities during an interaction generally are distributed between parties. UML use case diagrams and descriptions for actors of different roles are found in Section A.

4.4. People Scenarios

Scenarios that must be covered by BPEL4People as additional requirements as mentioned by [3] are:

Nomination A potential owner may be *nominated* to become the actual owner of a task. In such a scenario, even though there might be other potential owners, only nominated owners shall be allowed to *claim* a task.

Four Eyes Principle A task may be given to persons that are *independent* in order to obtain different opinions and for comparison. In such a scenario, although the set of potential owners might be the same, we do not want the owner of one task to also claim another equal task. If a potential owner claims one of the tasks he must be *excluded* as a potential owner for all other equal tasks. This scenario requires thus that a potential owner might only claim one out of multiple equal tasks. Also the information of the actual owners *must not be disclosed* by the system in order to avoid any collusion between them.

Escalation An escalation takes place if deadlines are not met. In such a case escalation notifications to *escalation recipients* are sent.

Chained Execution Chained execution is an execution of a process fragment where the owner of one task, by successfully *completing* a task, automatically requests *ownership* for the following *chained task*.

4.5. Goals

We want to summarise the goals for BPEL4People: Within the context of a business process BPEL4People must support

- *role based interaction* of people with processes,
- provide means of *assigning* users to roles,
- *delegate ownership* of a task to a person only,
- *support scenario* as mentioned above

by extending BPEL

- with *additional orthogonal syntax and semantics*.

5. Evaluating the Standards

We now want to evaluate the standards BPEL4People and WS-HumanTask to the previously introduced goals and compare them to our work.

5.1. Human $\rightleftharpoons$ Process Interaction

Instantiation [2] specifies that a process will be associated with its process initiator at runtime. However it does not support constraints on who *may* initialise a process. We have implemented a more comprehensive definition of a process initiator that also permits declaration of *potential process initiators* that may initiate new processes that have been defined by BPEL4People administrators. Furthermore sets of the following parameters that we have defined can be stated during initiating that will be applied to the process instance for *customisation*:

Resolution Parameters Resolution parameters can be specified for *customising the people queries* that will be interpreted by a HR service during people resolution. As the implementation of the interpretation of the parameters can be realised in any way, there is maximum flexibility regarding the use of these parameters.

Deadline Parameters Deadline parameters *customise* deadlines that specify for example during when a task must be claimed or completed. If deadlines are missed, escalation takes place. Deadline parameters have to be interpreted by a deadline customiser.

Supply Data The operations *addAttachment* and *addComment* as defined in [1] permit the supply of ad hoc attachments and comments.

Notification Notification is defined in Section 5 of [1].

Request Data A process may request data by *invoking* a people activity that specifies an *outputVariable* as specified in section 4.1.1 of [2].

Provide Data The operation *setOutput* defined in [1] permits the submission of results to the process.

5.2. Protecting Human Tasks as Critical Resources

While the tasks execution is a *critical section* for owners, the task itself becomes the critical *resource* that must be protected accordingly as we do not want two persons to perform the same task unnecessarily. Using *stateful* tasks this is very easily accomplished by the two operations *claim* and *release* that a human task has to expose according to [1]. By claiming a task the user becomes the exclusive owner of the task. The owner may then revoke ownership of a task.

5.3. Assigning People to Roles

Generic human roles are defined in section 3.1 both in [2] for processes and [1] for tasks. People assignment can take place via *logical people groups*, via *literals* or via *expressions*. These people queries will be executed at runtime during people resolution and result in a set of users.

We have applied similar structures for people queries in oder to establish the relation between roles and users. Yet our people queries for a role can be combined set–theoretically using the *operation* attribute.

5.4. Supporting Scenario

Nomination As we wanted to distinguish between a set of users that *may* claim a task and the subset of users that *should* claim a task we realised *nomination* with an operation and a new role, the *nominated owner*. While nominated owners are *potential* owners in order to claim a task, only nominated owners may do so in case of a present nomination. [1] does not support nomination out of a set of potential owners, instead nomination simply equals to the use case of people assignment for the potential owner role.

Four Eyes Principle [2] covers this scenario in the sample demonstration. The *secondApproval* activity overwrites the tasks default people assignment for *excludedOwners*.

In contrast to this solution we implemented a simple accounting for *exercising* roles. The four eyes principle scenario is covered as following:

The people links optional attribute *useCredits*, if greater zero, indicates how often this link may be used[9]. Otherwise, there is no limit on how often a role may be executed. Thus: by setting the *useRoleCredits* attribute of two people activities potential owner's referenced people link to *one*, the four eyes scenario is realised.

When a user exercises a role, the people query within the people link is extracted by VieBOP. Its *useRoleCredit* attribute is then decremented by one and if it becomes zero the attribute *operation* is set to *exclude*.

Escalation Escalation is covered by Section 4.6 of [1]. Not only the escalation recipients but also a condition for the escalation can be specified. We realised the concept of *observers* for notifications and different types of *event–based changes* – observers may be subscribed to – and modelled the escalation recipient as a special type of observer that becomes notified in case of a missed deadline.

In case of an escalation, the *notification manager* creates a new escalation task and sends out notifications to escalation recipients. When the original task (that may be an escalation task as well) is completed, the escalation task terminates as well. This way, unnecessary work that might be performed by escalation recipients can be minimised.

Chained Execution Chained execution makes use of a logical link that may have been specified when modelling the process. Our people activities can specify a following people activity, using the *following* attribute for indicating that their tasks are related and that it makes sense to carry them out together.

A BPEL4People client may *complete* and *claim* a following task manually. With the explicit *claimFollowing* attribute of our *complete* element that is sent in a Web service message to VieBOP however, the operations of complete and claim can be executed in an *transactional* way ensuring the following task not to get claimed by another potential owner in case this additional security is beneficial.

[2] and [1] do not support chained execution in an atomic, transactional way, and do not provide means for linking people activities beyond the implicit sequential order of the workflow.

6. Discussion

Our presented VieBOP system permits usage of *existing* BPEL engines for process *deployment* and *execution* while managing the people aspects of BPEL4People. As a consequence, traditional BPEL engines *do not have to be extended* for BPEL4People in order to host processes where people are involved. Particularly they do not need to understand BPEL4People definitions. Instead *interpretation* and *handling* of BPEL4People data and aspects is done by VieBOP.

[9] A people link is used when a role's use case is exercised.

The HR service *hides the complexity of people links and queries* toward callers. Everything lower than *user objects* is managed and abstracted by this service. It permits customisation of people queries by offering an *extension point* for interpreting resolution parameters. The HR service can fully be *delegated* to a third party and does not have to reside on the engine. This brings the following advantages: *Sensitive* and individual related data can be directly hosted by an organisation and does not have to be placed externally. A huge, international, organisation that may run an instance of VieBOP, may choose to *delegate* the HR service towards it's national branches that run their respective HR servers *autonomously*. Information as the *locality* of the HR server or other criteria can subsequently be applied for people resolution permitting *individual customisation*.

Although all goals of BPEL4People have been met in [2] and [1] as well as in our work we have seen several opportunities for precising their realisations as in the case of the nominated owner, the customisation or the interpretation of people queries and deadline parameters. Particularly we introduced interesting accounting or notification concepts that are missing in the current specification of BPEL4People and WS-HumanTask.

7. Related Work

IBM and SAP published a white paper [3], that addresses various BPEL4People requirements and scenarios. We have analysed and taken this as a basis for our work. In June 2007 [2] and [1] have been released by various companies.

Conceptual *foundations* for process–aware collaborative WfMS are presented in [9]. As such the work – that focuses on artifacts, business processes and resources – is decisive for technologies such as BPEL4People and, in contrast to the latter, not only covers teams but also processes that have *ad hoc* character.

While the formal integration of humans into modern business process languages as BPEL is still a novel topic, process modelling has evolved over years. As a matter of fact, plenty of workflow languages exist nowadays.

Petri nets represent an abstract way of specifying processes [13] that BPEL can be transformed to [17]. BPEL itself evolved out of the Web Services Flow Language (WSFL) [14] and XLang [27]. [18] compares BPEL with BPMN [22] and with Yet Another Workflow Language (YAWL) [28]. XML Process Definition Language (XPDL) [35] supports complete graphical representation of BPMN and there are ambitions to transform BPMN into BPEL [34, 23].

As model driven development and engineering [21] become increasingly important, the application on processes represents an interesting undertaking. [25] discusses the suitability of UML activity diagrams for business process modelling. A framework for generating XPDL specifications from UML activity diagrams is presented by [12].

8. Summary

We have presented the architecture of VieBOP, a generic BPEL4People system that manages human aspects and enables traditional BPEL engines to handle transformed BPEL4People processes transparently. Moreover, we have defined goals for BPEL4People as derived from [3] and evaluated the standards by comparing them to our work.

Our syntax definition and our system enables users to specify BPEL4People definitions by assigning people to roles, by creating tasks for BPEL process activities that are encapsulated by people activities and by creating processes. It enables users to work with BPEL4People instances by instantiating processes, by querying tasks and processes, by altering a tasks state and by submitting work and ad hoc attachments. It interacts proactively with users by notifying users of changes or events, hosts people activities that encapsulate human tasks and that are invokable transiently as common activities by BPEL processes and engines via BPEL gateways. It interacts with a BPEL engine by managing the people activities states and results within the context of the process, by deploying a BPEL process to it, while extracting and conserving BPEL4People specific information, that will be associated with the process and by handling inputs and outputs.

9. Further Work

We plan to support the standards [2] and [1] in our future work by integrating them into conceptual models for describing human aspects of business processes within VieBOP.

9.1. Human Roles and User Interfaces.

User interfaces that may be specified with a task are not interpreted yet by our webclient. This is because a standard has to be defined how these user interfaces should be used and how the binding of input data takes place. XForms [33] was suggested for these user interfaces by [3] as an enabling technology. The topic of customised user interfaces for BPEL4People needs to be addressed properly.

9.2. Process Analysis.

Human business actors play key roles in the executional behaviour of business processes. Further work to trace down the impacts of actions on different levels of the process needs to be undertaken. Besides these tracing techniques, metrics for human interactions need to be defined.

A. BPEL4People Actors

The following definitions have been derived from [3] and not necessarily equal to the definitions as found in [2] and [1].

Business Administrator A business administrator (see Figure 5 for a UML use case diagram) is interested in all process instances of a specific process class.

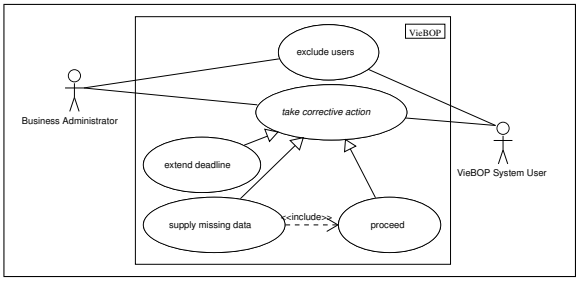

Figure 5. Business Administrator

Escalation Recipient An escalation recipient is a person (see Figure 6 for a UML use case diagram) that receives a notification if specified deadlines have not been met.

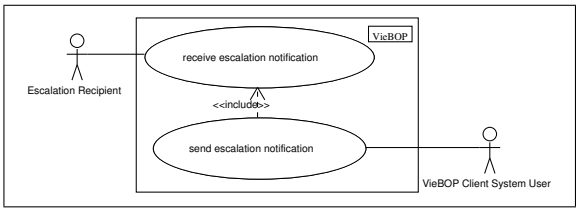

Figure 6. Escalation Recipient

Owner A potential owner (see Figure 7 for a UML use case diagram) that successfully claimed a people activity.

Potential Owner A potential owner (see Figure 8 for a UML use case diagram) may claim and complete a people activity.

Nominated Owner A nominated owner is a potential owner that has been nominated for ownership.

Process Initiator A process initiator is a person (see Figure 9 for a UML use case diagram) that may create an instance of a process or the person who actually initiated the process.

Process Stakeholder A process stakeholder (see Figure 10 for a UML use case diagram) can observe and influence a process instance.

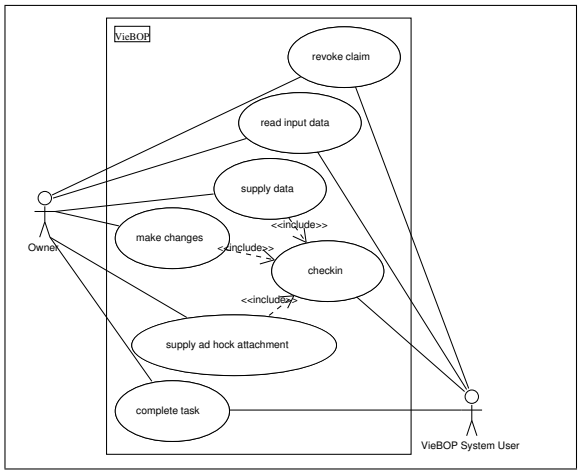

Figure 7. Owner

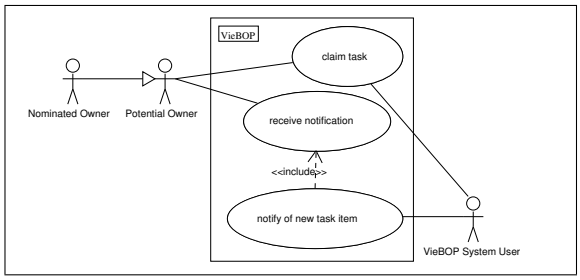

Figure 8. Potential Owner

Supervisor A supervisor monitors a process and may perform ownership nomination.

BPEL4People Administrator A person that defines new processes, is able to assign roles and has full control over BPEL4People specific data.

References

[1] *Web Services Human Task (WS-HumanTask), Version 1.0*, June 2007.

[2] *WS-BPEL Extension for People (BPEL4People), Version 1.0*, June 2007.

[3] a joint IBM-SAP whitepaper. *WS-BPEL Extension for People*, August 2005.

[4] R. Anzböck, S. Dustdar, and H. Gall. Software configuration, distribution, and deployment of web-services. In *SEKE '02: Proceedings of the 14th international conference on Software engineering and knowledge engineering*, pages 649–656, New York, NY, USA, 2002. ACM Press.

[5] P. V. Biron and A. Malhotra. *XML Schema Part 2: Datatypes*, October 2004.

[6] T. Bray, J. Paoli, C. M. Sperberg-McQueen, E. Maler, and F. Yergeau. *Extensible Markup Language (XML) 1.1*, August 2006.

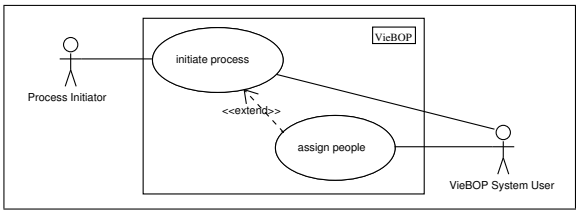

Figure 9. Process Initiator

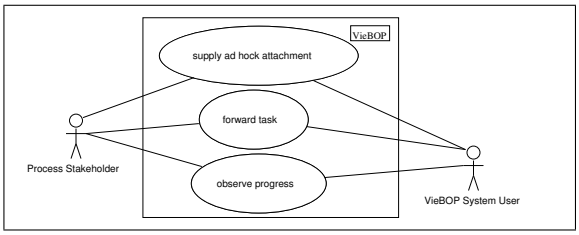

Figure 10. Process Stakeholder

[7] R. Chinnici, M. Hadley, and R. Mordani. *The Java API for XML-Based Web Services (JAX-WS) 2.0*, April 2006.

[8] F. Curbera, R. Khalaf, N. Mukhi, S. Tai, and S. Weerawarana. The next step in web services. *Commun. ACM*, 46(10):29–34, 2003.

[9] S. Dustdar. Caramba – a process-aware collaboration system supporting ad hoc and collaborative processes in virtual teams. *Distrib. Parallel Databases*, 15(1):45–66, 2004.

[10] T. W. Ferratt, A. L. Lederer, S. R. Hall, and J. M. Krella. Information technology and competitors: a case for collaborative advantage. In *SIGCPR '95: Proceedings of the 1995 ACM SIGCPR conference on Supporting teams, groups, and learning inside and outside the IS function reinventing IS*, pages 139–147, New York, NY, USA, 1995. ACM Press.

[11] D. Georgakopoulos, M. Hornick, and A. Sheth. An overview of workflow management: from process modeling to workflow automation infrastructure. *Distrib. Parallel Databases*, 3(2):119–153, 1995.

[12] N. Guelfi and A. Mammar. A formal framework to generate xpdl specifications from uml activity diagrams. In *SAC '06: Proceedings of the 2006 ACM symposium on Applied computing*, pages 1224–1231, New York, NY, USA, 2006. ACM Press.

[13] R. Hamadi and B. Benatallah. A petri net-based model for web service composition. In *ADC '03: Proceedings of the 14th Australasian database conference*, pages 191–200, Darlinghurst, Australia, Australia, 2003. Australian Computer Society, Inc.

[14] IBM Software Group. *Web Services Flow Language (WSFL) version 1.0*, 2001.

[15] ISO/IEC. *Unified Modeling Language (UML), v1.4.2*, April 2005.

[16] B. Ives and G. P. Learmonth. The information system as a competitive weapon. *Commun. ACM*, 27(12):1193–1201, 1984.

[17] N. Lohmann, P. Massuthe, C. Stahl, and D. Weinberg. Analyzing interacting bpel processes. In *Business Process Management*, pages 17–32, 2006.

[18] V. Martin and S. Dustdar. A view based analysis of workflow modeling languages. In *Lecture Notes in Computer Science*, pages 276–290, 2006.

[19] B. Medjahed, B. Benatallah, A. Bouguettaya, A. H. H. Ngu, and A. K. Elmagarmid. Business-to-business interactions: issues and enabling technologies. *The VLDB Journal*, 12(1):59–85, 2003.

[20] OASIS Web Services Business Process Execution Language (WSBPEL) TC. *Web Service Business Process Execution Language Version 2.0*, January 2007.

[21] Open Management Group. *MDA Guide Version 1.0.1*, June 2003.

[22] Open Management Group. *Business Process Modeling Notation Specification*, May 2006.

[23] C. Ouyang, M. Dumas, A. H. M. ter Hofstede, and W. M. P. van der Aalst. From bpmn process models to bpel web services. In *ICWS '06: Proceedings of the IEEE International Conference on Web Services (ICWS'06)*, pages 285–292, Washington, DC, USA, 2006. IEEE Computer Society.

[24] M. Roth and E. Pelegrí-Llopart. *JavaServer Pages™ Specification Version2.0*, November 2003.

[25] N. Russell, W. M. P. van der Aalst, A. H. M. ter Hofstede, and P. Wohed. On the suitability of uml 2.0 activity diagrams for business process modelling. In *APCCM '06: Proceedings of the 3rd Asia-Pacific conference on Conceptual modelling*, pages 95–104, Darlinghurst, Australia, Australia, 2006. Australian Computer Society, Inc.

[26] Standard ECMA-262. *ECMAScript Language Specification*, December 1999.

[27] S. Thatte. *Web Services for Business Process Design*. Microsoft Corporation, 2001.

[28] W. M. P. van der Aalst and A. H. M. ter Hofstede. Yawl: yet another workflow language. *Inf. Syst.*, 30(4):245–275, 2005.

[29] W3C Note. *Web Services Description Language (WSDL) Version 1.1*.

[30] W3C Recommendation. *Cascading Style Sheets, level 2 CSS2 Specification*, May 1998.

[31] W3C Recommendation. *XHTML™ 1.0 The Extensible HyperText Markup Language (Second Edition)*, August 2002.

[32] W3C Recommendation. *XML Schema Part 1: Structures*, October 2004.

[33] W3C Recommendation. *XForms 1.0 (Second Edition)*, March 2006.

[34] S. A. White. Using bpmn to model a bpel process, February 2005.

[35] The Workflow Management Coalition Specification. *Process Definition Interface – XML Process Definition Language*, October 2005.

[36] O. Zimmermann, V. Doubrovski, J. Grundler, and K. Hogg. Service-oriented architecture and business process choreography in an order management scenario: rationale, concepts, lessons learned. In *OOPSLA '05: Companion to the 20th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pages 301–312, New York, NY, USA, 2005. ACM Press.