Towards a Standardized Task Management

Tobias Unger[1], Thomas Bauer[2]

[1]University of Stuttgart, Institute of Architecture of Application Systems (IAAS)
Universitätsstrasse 38, 70569 Stuttgart, Germany
unger@iaas.uni-stuttgart.de
[2]Daimler AG, Group Research & Advanced Engineering (GR/EPD)
P.O. Box 23 60, 89013 Ulm, Germany
thomas.tb.bauer@daimler.com

Abstract: Business processes are increasingly controlled by IT-systems automatically, but they still consist of many tasks that have to be performed by people. Despite an appropriate IT-infrastructure is required for Task Management, until now this is a neglected topic in the research domain. In general, existing concepts and products for Task Management are not sufficient and, even worse, inter-partner aspects are not supported at all. For the first time, this paper analyzes the requirements for Task Management in a comprehensive way. Furthermore, we present an architecture for a Task Management Infrastructure that allows to fulfill these requirements even in inter-partner scenarios. This architecture was developed in the TAMPRO project and is based on emerging standards as WS-HumanTask and BPEL4People, which are discussed as well.

1 Introduction

Currently, there is a general tendency to consider business processes as an important success factor of enterprises. Despite of all efforts undertaken to increase process automation there still remain numerous tasks which have to be carried out by people. Hence, an integration of people in business processes becomes more important. The work that has to be accomplished by people is specified using (human) tasks. A task is assigned to an actor, which is able to complete the task; e.g., the design of a bracket has to be done by an engineer. The assignment is defined by a query on the business organization. Actors interact with their tasks using a worklist. Furthermore there is a trend to directly integrate partner enterprises in business processes. Therefore partner enterprises must have access to their tasks. These cooperations may appear in many different shapes; e.g., employees of the partner enterprise may perform single process steps whereas in other scenarios two or more enterprises are glued together to a virtual enterprise using one common, integrated business process; e.g., for developing cars.

Process-aware applications are increasingly implemented using the paradigms of service-oriented computing. Building applications using the design principles of a service-oriented architecture (SOA), enterprises are able to orchestrate their applications using business processes. Web services are the latest technology commonly accepted for implementing SOA applications. The Web Services Business Process Execution Language (WSBPEL or BPEL in short) [OAS07] is the de facto standard for composing Web services into a business process that in turn is made available to the outside as a Web service. Nevertheless, the focus of BPEL is to orchestrate automated services. For a long time the Web service standard architecture did not address the integration of humans in business processes. Recently, WS-BPEL Extension for People (B4P, [Ag07b, Kl05]) and Web Services Human Task (WS-HT, [Ag07a]) have been published, which address this topic.

In spite of WS-HT and B4P there is no concept for Task Management, which is able to deal with the issues shown above. Especially inter-partner task management is neglected. Additionally, each application has its own understanding of task management and its own implementation. Thus, people mostly have to use multiple worklists and multiple clients to get their work done. These topics are addressed in the TAMPRO project, in which we are developing a standardized Task Management and an appropriate infrastructure. The infrastructure allows unlike most of the Workflow Management Systems (WfMS) to manage tasks as separated entity using a dedicated task engine. All task-related functionality is extracted from applications. The Task Management acts as an integration layer for tasks and provides an integrated view to all tasks independent of the source application. Figure 1 shows a high level overview about the Task Management. The applications on the left side are using task functionality provided by the Task Management.

In this paper we discuss the requirements and the architecture for the Task Management. Furthermore, we compare the requirements with B4P and WS-HT in order to evaluate, if we can use these standards as a foundation for our Task Management Infrastructure. The paper is organized as follows: Section 2 discusses scenarios derived from case studies. In Section 3 we present important requirements for an Inter-partner Task Management and afterwards in Section 4 our preferred architecture. Section 5 compares our requirements with B4P and WS-HT and gives an overview about related work. Section 6 concludes our work and offers an outlook to future work.

2 Scenarios

In the TAMPRO project we have conducted case studies internally in DaimlerChrysler. From these case studies we derived three scenarios which are presented in the following: (i) a scenario stating, which applications types must be supported, (ii) a scenario showing, how people work together especially in inter-partner relationships, and (iii) a scenario showing, how these applications are deployed to the computing infrastructure.

Mostly, processes are supported through one or more applications. However, not all applications support processes directly; e.g., some types of applications only provide user interfaces for manipulating process relevant data, but do not track the execution of the process. In this case, the execution order of the tasks of a business process cannot be guaranteed. More and more, applications are realized with a notion of the running processes and are observing the right execution of the processes. Our analysis shows three different types of applications (cf. [BRB05]): (i) applications can be implemented as Process-aware Applications based on WfMS, (ii) as Process-oriented Applications implemented with conventional programming techniques, and (iii) as (Process-unaware) Applications carrying out process-related tasks, without being aware of the process. Additionally, many business processes are implemented mixing two or all types. All these types of applications have in common, that they are implicitly or explicitly generating tasks, which have to be performed by people.

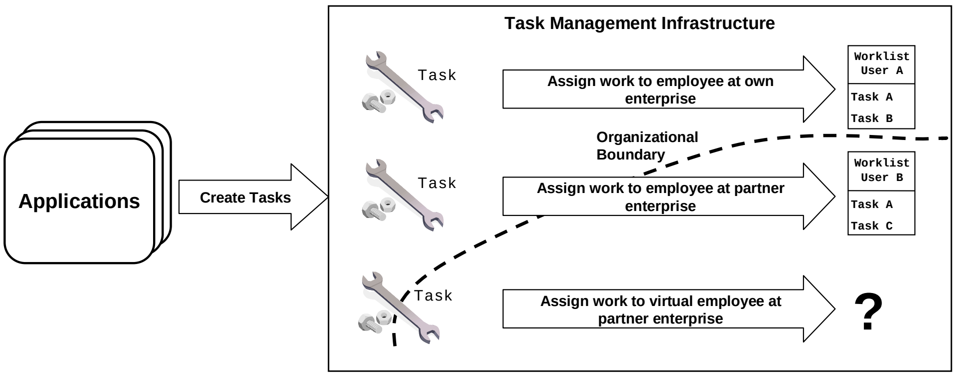
Task Management Infrastructure
Applications
Create Tasks
Task
Task
Task
Assign work to employee at own enterprise
Assign work to employee at partner enterprise
Assign work to virtual employee at partner enterprise
Organizational Boundary
Worklist User A
Task A
Task B
Worklist User B
Task A
Task C
?

Figure 1: Task Management Overview

3 Requirements

In this section we discuss important requirements for the Task Management derived from the TAMPRO project. During our analysis we recognized a division of the requirements into two groups: (i) functional requirements describing the intended behavior of the task management and (ii) requirements related to communication infrastructure. Table 1 gives an overview of the requirements that will be discussed in the following. However, implicit requirements exist, which are not discussed here (e.g. the existence of a worklist). These requirements also apply to existing WfMS (cf. [LR00]).

Requirement 1 defines the life-cycle of a task. A simplified model of our life-cycle is shown in Figure 2. We focus on the normal case and omit e.g. error-handling. The life of a task starts with the creation. Afterwards actors are assigned and the task transitions into the Ready sate. Now, the task can be claimed by a single actor and therefore only this actor can process the task, e.g. using a CAD-tool. During being processed, the task is in the Running state. The processing of a task can be intercepted. The intermediate results are kept and the task changes its state to Saved. On successful completion the task transitions into the Completed state.

Looking at the application types, we see that we need a possibility to model and execute a task standalone as its own entity (**Requirement 2**). In traditional WfMS tasks are modeled and executed as part of a process and the task functionality cannot be used from outside (e.g. tasks cannot be created by arbitrary applications). Additionally we must support applications that have no notion of the process during runtime. Therefore we must provide interfaces, so that the task management can be used by all types of applications. For process-unaware applications, a task must be offered like any other service. This means that the calling application does not know if the service is implemented by a human or not. The other types of applications need interfaces, that enable them to control the complete life-cycle of the task.

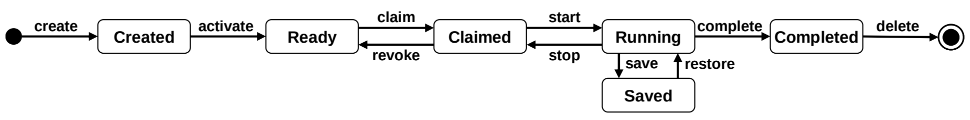

Figure 2: Task Life-Cycle

Despite of having standalone tasks, tasks and processes should be able to exchange context information (**Requirement 3**). As we see later, some information derived from the process context may be required by tasks (e.g. for staff assignment).

A powerful staff assignment mechanism (**Requirement 4**) is needed, because enterprises have very complex business organizations. A single employee may belong to multiple organizational entities (e.g. department, groups, projects). Furthermore many individual attributes can be related with each person (e.g. skills, position). During the staff assignment procedure the Task Management should be able to evaluate all these data in any combination; e.g., all persons who belong to a certain department and have certain skills should be allowed to perform a certain task. Partly, the applications themselves determine the potential persons on their own. Therefore the Task Management must be able to obtain the staff from the task or process context (e.g from the input data). Some processes require special staff assignment patterns like the 4-eyes principle; e.g., a decision should be made by two people independently of one another. In this case, the Task Management must be able to obtain the staff assignment of elapsed tasks in the current staff assignment. This data is mostly part of the process context.

Functional Requirements		**Infrastructure Requirements**	
Req1	Task Life-cycle	Req15	Integration of Partner Infrastructure
Req2	Standalone Tasks	Req16	Continuous Security
Req3	Exchange of Context between Task and Process		
Req4	Powerful Staff Assignment		
Req5	Delegation of Tasks		
Req6	Substitution of Tasks		
Req7	Notification of Tasks		
Req8	Escalation of Tasks		
Req9	Access to Process and Task Context Information		
Req10	Partner Autonomy		
Req11	Location Dependent Task Visibility and Access		
Req12	Integrated Worklists for Actors		
Req13	Multiple types of Implementations		
Req14	Adaptable, Customizable Client Framework		

Table 1: Task Management Requirements

Occasionally an actor wants to delegate some of his work to other actors; e.g., a manager delegates some work to one of its assistants. Thus the task management has to provide a delegation mechanism (**Requirement 5**). Partial results obtained before the delegation should be kept. Additionally, the Task Management should offer the possibility to restrict the delegation. If the processing of a a task requires a certain skill, the Task Management may have to assure that the receiver of the delegation has the required skills as well. Anyway, it must be assured that the task is not delegated to an actor, which is not allowed to work on the task; e.g., because of a 4-eyes principle. Occasionally the sender of a delegation intends to keep control over the delegation; e.g., the manager wants to observe the delegated work. In some cases he even wants to revoke the delegation. Hence we define two delegation types: (i) the revocable delegation allows the sending person to keep control about the delegation. The delegated task remains in the sender's worklist. Thus he is able to observe the completion of the task or to revoke the delegation. (ii) If the delegation is non-revocable the sender is not able to observe or control the task anymore. The task disappears from the actor's worklist.

If an actor is not available (for instance he is on vacation), his work must be taken over by another actor. Thus we need a powerful substitution mechanism (**Requirement 6**). The absence of an actor may have several reasons and therefore we need two activation mechanisms: (i) the substitution can be activated by an actor or administrator; e.g., before he/she goes on vacation he/she will explicitly activate his substitution policy. (ii) Substitution can be activated by the task management; e.g., when an user disconnects from the task management. The definition of a substitution policy can be done during modeling by a process modeler, during runtime by an administrator, or by an actor himself. A substitution policy uses a staff query to obtain the substitutes and refers to an organizational entity, a process, or a task; e.g., a manager may define a policy, that he is substituted for all task, which are assigned to him because he is in a role "project leader", by all other department managers which are in the role "project leader". For a special task an actor may define, that he is substituted by anyone of his department. Furthermore a substitution policy must define the activation mechanism. Analog to the delegation, the 4-eyes principle must be assured. Furthermore substitutions may also be revocable and non-revocable.

At some points during the execution of a process there is a need to notify other persons; e.g., in the change management process the responsible persons of impacted parts can be notified about problems during the realization (**Requirement 7**). As notifications are process relevant informations, actors should receive notifications via their worklist. Unlike notification using email, all process-relevant information can be accessed using the worklist. Notifications can be initiated automatically by a process or ad-hoc by an actor performing a task. The receivers of a notification may be defined directly or using a staff query.

Often tasks are assigned with deadlines; e.g., a task must be completed within 6 hours. If the task is not completed after 5 hours an escalation (**Requirement 8**) is triggered. The escalation may inform people using a notification or the task may be reassigned to another actor. Escalations can be chained; e.g., after 5h a notification may be triggered. If the task is still not completed after another 30 minutes a reassignment is triggered.

Actors should be able to see only these tasks on their worklist, they are currently interested in. For instance an actor may only be interested in tasks belonging to a certain car series. Therefore, the car series should be displayed in the worklist and other tasks should be filtered. Mostly, parts of information for displaying and filtering must be obtained from the process or task context (**Requirement 9**). Thus the task management must offer the functionality to query tasks using context information and functionality to define the context data that is displayed in the worklist.

Partner-integration is an important issue of task management. As already described, there are different types of people assignments. Especially if a task is completely outsourced, it cannot be assigned directly, because the name of the actor at the partner's side is not known. In such scenarios the partner has the autonomy (**Requirement 10**) to perform its own staff assignment and even not to publish information about the real actor. Additionally the partner may have the autonomy to disconnect from the life-cycle of a task. If an assessment task within the change management process is skipped, because the process is terminated, the partner is free to complete the task, if he needs the results internally, without returning any data.

Requirement 11 claims, that particular tasks are not visible to an actor and consequently the task is not processable if the actor is outside a specific physically location. For instance, prototypes of cars are kept in secret by an automotive company. This also applies also to related tasks and data. No data concerning the prototype may cross the boundary of the enterprise, independent of the enterprise the actor of the task is employed. The actor must be physically inside a specific location in order to process the task.

One limitation of today's application architectures is that each application comes up with its own client and worklist. Users must check multiple worklists to get an overview about his work. If we look at the deployment scenarios described above, we see that there does not exist a single instance of the task management infrastructure. The task management should offer an integrated worklist to an actor (**Requirement 12**).

If an actor starts a task using his worklist, the task management has to start the implementation of the human task (**Requirement 13**). The working environment of an actor should be prepared by the infrastructure, so that the user immediately can start working. That means data must be loaded, programs must be started, etc.. Typically, two types of implementations can be distinguished: e.g. in form-based clients, the implementation is realized as a single page or a page flow. However, forms are not always sufficient: Some tasks are more interactive, such as the design of a part of an automobile. Typically, worklist and implementation are displayed within the same client. Partly, the presentation of the worklist and the implementation of a task must be separated in two distinct clients (e.g. for integrating existing tools). In this case the task management must be able to start and interact with the separated client.

It is impossible to realize a single client that fits to all requirements of all applications. A client used by an engineer is completely different to a client used during manufacturing. Thus the task management delivers a client framework instead of a complete client. Each application or set of applications can use the framework to develop a customized client (**Requirement 14**). The framework should also be adaptable to any client architecture (e.g. rich clients, thin clients). Furthermore established programming platforms (e.g. Java or .NET) and interface styles (e.g. synchronous or asynchronous) should be supported.

To enable partners to work remotely on tasks, their communication infrastructures must be connected. Unfortunately this is a very complex issue, since each enterprise has its own communication infrastructure and integration style; e.g., some enterprises are using messaging as integration style [HW03] whereas other enterprises are using remote procedure calls as integration style (**Requirement 15**).

The task management should integrate itself smoothly into a broad range of security infrastructures (**Requirement 16**). Thus, encoding, authentication and authorization policies can be defined in a central directory; otherwise they must be defined individually for the tasks of each application.

4 Architecture

In this section we shortly introduce the architecture of our task management infrastructure. Indeed existing WfMS (e.g. [IBM06, He06, Ora07, Act07]) have integrated task management functionality, but mostly this functionality is not available standalone and w.r.t. to several further requirements it is insufficient; e.g. none of these systems support partner autonomy. WS-BPEL, B4P, and WS-HT focus on the functional description and only denote an architecture for the runtime infrastructure. Furthermore some architectural aspects are left open consciously. Altogether, this results in a lack of suitable existing architectures for task management. Therefore, in the following we present an architecture that allows to realize the requirements discussed before.

Components: The architecture must be suitable to execute standalone tasks, created by arbitrary applications. Hence, we define a separated *Task Engine* for executing tasks, that is able to control the life-cycle of the tasks (cf. Figure 3). *Arbitrary applications* interact with the *Task Engine* using defined interfaces. The support for WfMS-based applications is achieved, by providing a dedicated *Process Engine* that executes the business processes. The Staff Assignment requires a dedicated *Organizational Directory*. Through the separation from the *Task Engine*, the *Organizational Directory* may be used by *arbitrary applications* as well as by the *Process Engine*. These components also need access to organizational data; e.g., for evaluating whether an actor is authorized to start a process. To view their worklists and to start tasks, actors interact with the task management by using *Clients*. A single *Client* can connect to multiple Task Engines. This enables the infrastructure to present an integrated worklist to an actor even if the tasks are located on several *Task Engines*. A dedicated component concerning security is not provided since each component must implement security functions.

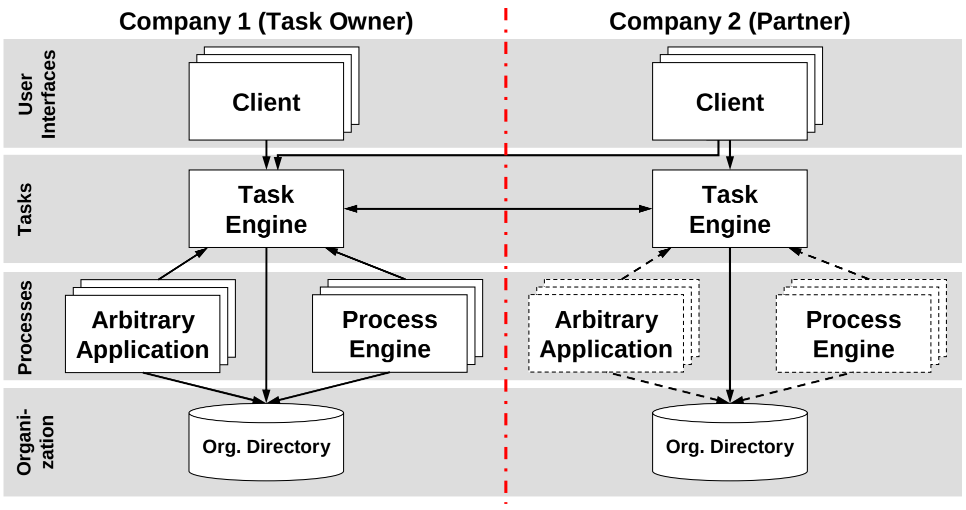

Figure 3: Task Management Architecture

Partner Integration: If the partner has no autonomy only a client is needed (except he is planning further integrations with his infrastructure). To support partner-autonomy the partner needs an own component for task handling, especially for controlling the life-cycle and performing staff assignment (cf. Req10). Thus, the partner requires a *Task Engine* as well. This results in the challenge to integrate the partner's *Task Engine* with the task management of the task owner. There exist two possible solutions: (i) the applications or the *Process Engine* at the task owner side can directly create tasks at the partner's *Task Engine*. An advantage of this solution is that existing interfaces can be used for connecting the applications and the *Task Engine*. But this solution results in inflexibility, since it is not possible to support all collaboration scenarios: If the task is only located at the partner's *Task Engine* internal actors cannot work on the task concurrently. (ii) Our solution is based on an interaction between the *Task Engine* of the task owner and the partner's *Task Engine* (cf. Figure 3). To determine whether a task should be federated to a partner or not, we use the normal staff assignment process in combination with a resource virtualization: each partner is registered as virtual resource in the organizational directory (e.g. as user). The task is transferred to a partner either using a pull or push mechanism. By executing a second staff assignment, the partner determines actors, who are finally allowed to work on the task by using its own Organizational Directory. Because of the two required staff assignments, we call this approach 2-stage staff assignment.

After the task is transferred to the partner, instances of the same task exist in every involved Task Engine. If one partner is involved, there are two instances and for every further partner the number of instances increases. Thus the *Task Engines* must exchange coordination messages in order to synchronize the task state. Thereby, the *Task Engine* of the task owner acts as coordinator. If a task is claimed at one Task Engine, the task has to be locked at all other *Task Engines*, in order to avoid redundant work.

Furthermore if a process terminates a task, the task must be terminated at each involved Task Engine.

The *Task Engine* at the partner's side cannot only be used to integrate the partner with the task owner. Similarly to the task owner, the partner can use the *Task Engine* to control tasks from any application. Hence the partner may connect its own applications and *Process Engines* to the *Task Engine*.

5 Discussion

As mentioned in the introduction, until now, researchers and software vendors have addressed task management aspects sparsely. Recently by publishing WS-BPEL Extension for People (B4P, [Ag07b]) and (Web Services Human Task (WS-HT, [Ag07a]) the issue is addressed by a public available specification. In this section we discuss related work to task management and evaluate, if B4P and WS-HT fulfill our requirements.

5.1 Related Work

The Workflow Management Coalition proposes a reference architecture for Workflow Management Systems [Hol95]. It allows to connect WfMS from different vendors by standardized interfaces. In contrast to our architecture, the WfMC promotes a monolithic architecture for WfMS. [PPC97] proposes a distributed workflow system for the internet. By using a generic workflow framework based on well-defined interfaces, components can be distributed cross-boundaries. For assigning tasks to an actor, a component based on the worklist interface can be built that is similar to our Task Engine. However, resource virtualization is not supported and most of the functional requirements (e.g. delegation, substitution) are not mentioned. [LR00] presents an algorithm for federating workitems to a remote workflow system. However resource virtualization is not supported. [LK06] proposes an activity manager, which is similar to our task engine. Unlike our focus on partner integration, the activity manager focuses on ad-hoc and collaborative tasks.

Furthermore, there is related work addressing the topic staff assignment. As we do not focus on staff assignment, this complements our work. Russel et al. [Ru05] describe various patterns, how staff can be assigned to a task w.r.t. the task's life-cycle. However, most of the allocation patterns and some of the more sophisticated patterns like delegation are currently supported by the TAMPRO project. [Ha98] discusses push and pull principles for assigning actors with tasks. The task management focuses on the push principle, however, using suitable staff assignments combined with the query capabilities the pull principle may also be supported. Furthermore, [Ha98] describes a concept for a multistage assignment, but this concept does not focus on partner integration. Using RBAC [Fe01] staff assignments can be defined based on roles. Users are made members of roles and roles are assigned with permissions. However, RBAC focuses on security aspects and an assignment only based on roles is too coarse-grained for task management. As roles can have several thousand members, we need further mechanisms to limit the assigned users. PBDM [ZOS03] extends RBAC with a flexible delegation mechanism, but the impact of a delegation to the life-cycle of a task is not covered. [BJ95] identifies many limitations of Workflow Management Systems in staff assignments and organizational structures which apply to the Task Management as well.

Many products [IBM06, He06, Ora07, Act07] are delivered with support for task functionality. But the products miss most of our requirements. However standalone tasks are supported by [He06, Ora07]. A sufficient support for task implementations (Req13) is provided by [IBM06]. [He06] comes with sufficient escalation support. Actually, no product supports WS-HT and B4P.

5.2 Evaluation of BPEL4People and WS-HumanTask

The topic integration of human beings in service-oriented applications is divided in two complementary specifications: Web Services Human Task (WS-HT) [Ag07a] provides a notation for human tasks, a behavioral description, and interfaces for interacting with human tasks. The life-cycle of WS-HT conforms in a large part to our life-cycle (Req1); except that WS-HT uses different terms, and tasks may be suspended additionally in state Claimed and Ready. Unlike most WfMS, tasks can be used standalone and are exposed as callable Web services. The B4P specification [Ag07b] describes the integration of human tasks and BPEL [OAS07]. Therefore BPEL is extended with a PeopleActivity. B4P describes three major scenarios, how tasks can be integrated in a BPEL Process using a PeopleActivity: (i) tasks can be defined inline as part of the BPEL process. (ii) Tasks can be defined standalone without a callable Web service interface. In this case the communication between the people activity and the human tasks happens using an implementation-specific, non-standardized protocol. (iii) An external task can be integrated by invoking its callable Web service interface. Optionally, this invocation can be coordinated by a defined protocol. Accordingly Req2 is covered as WS-HT allows defining standalone tasks. Arbitrary applications or dedicated Process Engines can optionally use the standardized coordination protocol to coordinate the invoked task. Thus, a task can be terminated as a result of the termination of the calling process.

Exchange of context between tasks and processes is only provided if tasks are modeled inline. In contrast, our architecture separates the Task Engine and the Process Engine on a conceptual level. In order to keep the separation, standalone human tasks must be used. In this case no context can be passed from process to tasks and vice versa. Hence Req3 is not fulfilled.

In WS-HT people can be assigned with tasks using Literals, Logical People Groups, or Expression. Req4 is not fulfilled, as complex staff queries referring to multiple attributes of users or groups cannot be defined. In order to define such complex queries, we have to develop additional X-Path functions or provide a custom expression language.

WS-HT allows defining delegation policies which govern to whom a task can be delegated, and the behavior of a task’s life-cycle when executing a delegation operation. As there is no statement whether the delegations are recoverable and only fixed policies can be defined for delegations, Req5 is not fully supported. Unfortunately, substitutions (Req6) are not even mentioned in WS-HT. However WS-HT defines a notification and escalation mechanism, which completely covers our requirements (Req7 and Req8).

In order to specify information that is displayed in the worklist, WS-HT provides Presentation Elements. The presentation can be enriched using context information. In contrast, querying tasks based on the process context is not supported. Therefore Req9 is not completely supported.

Furthermore, WS-HT defines no mechanism to connect two or more Task Engines in order to support partner autonomy (Req10). The introduced coordination protocol focuses on the coordination between a process and a task; e.g., a state change from claimed to running cannot be propagated form one Process Engine to another. Also location-based task-visibility is not supported (Req11).

For the interaction of actors with a task multiple renderings can be specified which are displayed to the actor by an arbitrary client; e.g., it can be specified, how the task can be rendered as form, portlet, etc. Beside the description of the rendering WS-HT defines no mechanisms, how clients deal with these renderings or external applications are invoked and controlled. Hence Req13 is not completely covered.

WS-HT only denotes the architecture and functionality of the client. It is only mentioned that clients may be separated in a task-description UI and a task-list UI and that clients should be able to connect to multiple Task Engines. Therefore we cannot evaluate Req12 and Req14. The same applies for Req15 and Req16, as security is out of scope of WS-HT and B4P, and assumptions about the (communication) infrastructure are not introduced.

6 Summary

Task Management is a neglected topic in the research domain. Especially the inter-partner aspects mostly are neglected. In this paper we have presented requirements for a (Inter-partner) Task Management and a resulting architecture. We have analyzed, if WS-HT and B4P are a suitable foundation for (Inter-partner) Task Management. Currently some of our requirements especially related to partner autonomy are not covered in WS-HT and B4P. However, in the TAMPRO project we plan to develop the (Inter-partner) Task Management using these standards. Therefore, we have to extend WS-HT and B4P at certain aspects. Current work is about extending WS-HT for partner autonomy. Furthermore, we are currently evaluating products from different vendors against our requirements. Therefore, we have implemented prototypes showing the feasibility of our requirements; e.g., we have implemented a prototype for evaluating the 2-stage staff assignment.

References

[Ag07a] A. Agrawl et al. Web Services Human Task. Technical report, Active Endpoints, Adobe, BEA, IBM, Oracle, SAP AG, June 2007.

[Ag07b] A. Agrawl et al. WS-BPEL Extension for People Specification. Technical report, Active Endpoints, Adobe, BEA, IBM, Oracle, SAP AG, June 2007.

[Act07] Active Endpoints. ActiveBPEL for People, 2007.

[BJ95] C. Bussler and S. Jablonski. Policy resolution for workflow management systems. In HICSS ’95. IEEE Computer Society, 1995.

[BRB05] R. Bobrik, M. Reichert, and T. Bauer. Requirements for the Visualization of System-Spanning Business Processes. In DEXA Workshops, pages 948–954. IEEE Computer Society, 2005.

[CC06] Frederick Chong and Gianpaolo Carraro. Architecture Strategies for Catching the Long Tail. Technical report, Microsoft, April 2006.

[Fe01] D. F. Ferraiolo et al. Proposed NIST standard for role-based access control. ACM Trans. Inf. Syst. Secur., 4(3):224–274, 2001.

[Ha98] J. Hagemeyer et al. Arbeitsverteilungsverfahren in Workflow-Management-Systemen: Anforderungen, Stand und Perspektiven. In A.-W. Scheer, editor, Veröffentlichungen des Instituts für Wirtschaftsinformatik, 1998.

[He06] B. Heumann et al. WebSphere Process Server V6.0 Business Process Choreographer Programming Model. Technical report, IBM Corporation, 2006.

[Hol95] D. Hollingsworth. The Workflow Reference Model. Technical report, Workflow Management Coalition, 1995.

[HW03] G. Hohpe and B. Woolf. Enterprise integration patterns: designing, building, and deploying messaging solutions. Addison-Wesley, 2003.

[IBM06] IBM Corporation. IBM MQSeries Workflow: Concepts and Architecture, 2006.

[Kl05] M. Kloppmann et al. WS-BPEL Extension for People - BPEL4People. Technical report, IBM, SAP AG, 2005.

[LK06] P. Loos and H. Krcmar. Architekturen und Prozesse. Strukturen und Dynamik in Forschung und Unternehmen: Strukturen Und Dynamik in Forschung Und Unternehmen. Springer, Berlin, 2006.

[LR00] F. Leymann and D. Roller. Production Workflow: Concepts and Techniques. Prentice-Hall, Upper Saddle River, New Jersey, 2000.

[OAS07] OASIS. Web Services Business Process Execution Language Version 2.0 – Committe Specification. Published via Internet, Jan 2007.

[Ora07] Oracle Corporation. Oracle BPEL Process Manager, 2007.

[PPC97] S. Paul, E. Park, and J. K. Chaar. RainMan: A Workflow System for the Internet. In USENIX Symposium on Internet Technologies and Systems, 1997.

[Ru05] N. Russell et al. Workflow Resource Patterns: Identification, Representation and Tool Support. In CAiSE, pages 216–232, 2005.

[ZOS03] X. Zhang, S. Oh, and R. Sandhu. PBDM: a flexible delegation model in RBAC. In SACMAT ’03. ACM Press, 2003.