Business Process Modeling, Task Management, and the Semantic Link

Uwe V. Riss and **Ingo Weber** and **Olaf Grebner**
SAP Research, CEC Karlsruhe
Vincenz-Priessnitz-Str. 1, 76131 Karlsruhe, Germany
{uwe.riss, ingo.weber, olaf.grebner}@sap.com

Abstract

In the current paper we describe how to combine web services and task patterns in a joint infrastructure. The central intention is to make up executable business processes by including generic pattern based web services which establish a relation between the service infrastructure and task management. This approach significantly increases the flexibility of process modeling due to the flexible definition and introduction of new task patterns and their incorporation in the web service framework. Semantic technologies are the common ground which enables the combination of the two approaches. Thus gaps in service compositions can be closed and later completed by automated services if a demand is asserted.

Introduction

Today one of the major challenges for enterprises is the continuous competition in terms of innovation, growth, and competitive advantage. Keeping pace with these challenges requires continuous and fast adaptation of existing and development of new business processes as well as their swift realization in business applications. The introduction of Service-Oriented Architecture (SOA) in enterprise software solutions aims at meeting such demands (Sinur & Hill 2006).

However, the transformation of business-driven process descriptions into technology-based solutions is not straightforward. The mechanical derivation of executable business process models from high-level, graphical business process models is not yet achievable for most real-world processes. Moreover, it remains questionable whether it is even reasonable to completely cover all parts of the original process model by automatic services. Most likely there will be gaps, i.e., parts of the process model for which automated services are not available (in principle or temporarily); or the complexity goes beyond the possibilities of automation, e.g., if the number of process alternatives is too high; or there are exotic cases for which the cost of automation does not pay off. This is a particularly common scenario in knowledge-intensive work processes (Grebner *et al.* 2006).

In these cases an alternative approach can fill these gaps with the aid of human actors who step into the breach. In order to realize this approach, we enter the realm of human task management. Here we find a different portfolio of tools, e.g., task management systems that support users to organize and distribute work via email. From the process point of view the decisive difference is that a human actor (task owner) takes the responsibility for a task and the subprocess that results from it, e.g., subtasks that are defined and can be delegated to other persons.

However, task owners often require specific support in executing particular types of tasks, e.g., if they are not familiar with the proceeding to execute the task and require some guidance for this. To provide adequate support, the concept of task patterns has been introduced. Task patterns offer context-adapted information that help a human actor to accomplish the respective tasks at hand (Riss *et al.* 2005).

The intervention of human actors breaks the paradigm of automated service processing. To this end we refer to the concept of human task as it has been introduced in (Agrawal, A.; Amend, M. et al 2007a) and raise the question what an efficient integration of these human tasks in the infrastructure of automated processes can look like. Usually human tasks can be realized more easily than automated services to fill emerging gaps (see Figure 1). Of course, in suitable cases human tasks that follow a regular schema can be automated later by web services.

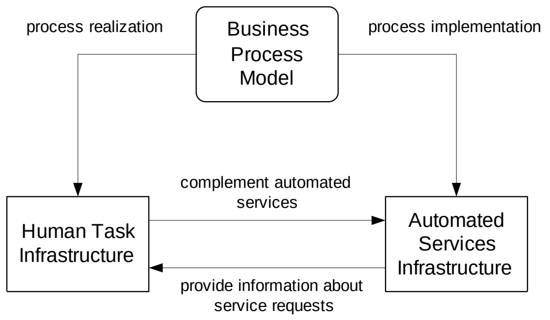
process realization
Business Process Model
process implementation
complement automated services
Human Task Infrastructure
Automated Services Infrastructure
provide information about service requests

Figure 1: Process realization by Automated Services and Human Tasks.

For our approach of integrating automated web service and human tasks to execute a given process model, semantic technologies play a decisive role. While in the former case the semantics are primarily used for service identification, in the latter case they help to find supportive information. The particular role of semantic technologies in this respect arises from the fact that it provided a common ground for both paradigms and thus decisively leverages the integration. In the following we shortly describe these two roles of semantics. Afterwards we sketch a possible approach to integrate both paradigms. After giving a detailed example we discuss the results.

The role of semantics

It has been pointed out that an efficient implementation of processes by automated services can be leveraged through the support by semantic technologies (Hepp *et al.* 2005). In the same way, task management can essentially benefit from semantic technologies (Grebner *et al.* 2006). It is this very fact that inspires the idea of an integration of automated services and human tasks infrastructure (Riss & Grebner 2006) based on a common semantic foundation. In particular it is possible to transfer some ideas regarding the application of semantic technologies from one area to the other. To this end, we have a closer look at the ways how these technologies are applied in both cases.

Semantics for enterprise services

One motivation to use semantic technologies for composing business processes is the demand for identification of suitable web services to fit in given processes. Semantic service descriptions provide an appropriate means to enable such identification. The related technology is referred to as semantic web services (SWS), where web service interfaces are described using Semantic Web technology to enable machine reasoning (Cabral *et al.* 2004). For example, the similarity of an available and a requested web service can be assessed in this way. This enables informed semantic web service discovery - as opposed to purely syntactic techniques such as input/output-based match-making.

While there are multiple initiatives to SWS (OWL-S, WSMO, and SAWSDL are the most well-known projects), there is a common denominator: to allow for techniques such as discovery, composition, or mediation of SWSs, the states before and after the execution of a service are described through preconditions and postconditions, respectively. In addition, non-functional properties (NFPs) are a common place to describe further aspects, such as quality of service (QoS) parameters, security options and the like. In service composition and discovery, the preconditions can be used to find out if a service is executable in a certain state - in other words: if the given state satisfies the preconditions. If so, the service can be executed, and by applying the postconditions the state after the service execution can be determined. NFPs are mostly orthogonal: they can be used to evaluate if a service satisfies the required QoS parameters, or to find the preferred service out of a set of functionally equivalent services.

Semantics in business process models

Facilitating Business Process Management with the benefits offered by semantic technologies is the broad vision of Semantic Business Process Management (SBPM) (Hepp *et al.* 2005). In order to enable this vision, semantic information is attached to artifacts used in BPM and specialized reasoning techniques are developed to make use of it. If every (relevant) activity, conditional control flow link, and data entity in a process model was semantically annotated, the opportunities would be manifold: based on this information, the space of existing process models could be accessed more easily, e.g. through queries that display all processes that have to do with meat (and thus, by semantics-enabled subsumption reasoning, chicken, beef, pork, kangaroo, etc.); in contrast to the status quo, the relationships between the conceptual business process models and the executable processes would be observable from the models - allowing for changes on one of the layers to be (partially) projected to the other; and, most relevant in the context here, finding suitable implementation forms and configuring their usage within business process contexts can be facilitated. These tasks are currently addressed by (i) providing notations and tools for adding semantic information to BP models, and (ii) developing suitable algorithms and methodologies for applying this information successfully.

The latter point relates to the need for specialized reasoning: combining process models and semantic information requires extensions to standard ontology reasoning techniques in order to take the execution semantics of the process model into account. While such a combined view on the models is not necessary for all reasoning tasks of interest, there are a number of examples where only the combination leverages the full potential of SBPM. One example is the validation of a semantic process model, i.e., the question whether the semantic annotations and the standard parts of the process model together “make sense”: an isolated view on the structure of the process or the semantic annotations alone cannot answer this question.

Semantics for task management

One of the central objectives of today’s task management is the reuse of incorporated work experience. To this end we have introduced the concept of task patterns (Riss *et al.* 2005) that guide users through the planning and execution of specific types of tasks. These task patterns can be understood as abstractions of individual task descriptions. They offer ways of possible task executions but leave the details to the responsible user.

The implementation of the task pattern approach requires the successful identification of suitable task patterns by users. In this respect task management faces similar challenges as service composition and we regard semantic technologies as a suitable means to address them.

One way to support the identification of task patterns goes back to goal descriptions. For example, the task with the goal ’business travel to Paris on November, 20th 2007’ can be assigned to the task pattern with the goal ’business travel abroad’. In this way the task is not only related to the task pattern but also to all other tasks that use the same pattern. This decisively simplifies the exchange of information between these tasks. Nevertheless, to define well distinguishable task patterns we need clear goal descriptions. When drawing a comparison of a formal representation of a task and its goals with the formal description of web services, we see that these are similar and thus can benefit from ontological descriptions in a similar way. Moreover, we see this as the common basis that facilitates the integration of web services and task management.

Generally, semantic technologies provide further benefits in the area of task management. Semantic relations establish the connection between tasks and the information objects that are relevant for this task. In this way they enable users to find information directly or indirectly related to the task. The Social Semantic Desktop (SSD) (Groza *et al.* 2007) plays a decisive role for the integration of task management in this respect, enriching the work environment of users. In particular the SSD allows for a tight integration of task and other personal information objects on the desktop. This helps the users to find work related information and at the same time increases the value of the task management for them.

Proposed approach

The central idea that we follow in our approach to provide an efficient integration of human actors into automated business processes is to fill gaps in service compositions by task patterns which are embedded in the service infrastructure via a generic web service. This web service can consume invocations for any kind of task pattern as detailed below. In this way we can leverage an existing web service infrastructure for dealing with task patterns and, where applicable, avoid the implementation of completely new web services. We see the main advantage of specifying task patterns compared to implementing web services in a gain of flexibility since they are described by attributes and do not require development. Moreover, they can rely on the intelligence of a task owner so that inadequacies or ambiguities in their definition can be still rectified. Of course, in contrast to web services, they build on human activities as the core of service execution and thus can be more time consuming in some cases, e.g., if high amounts of data are processed in a standardized way.

Originally, task patterns have been designed to support inexperienced users in executing specific tasks (Riss *et al.* 2005), e.g., to plan a business trip where the information about the specific destination and period of time is provided by the task assigned to this pattern while the general proceeding is proposed by the pattern. To this end the user first starts with the task and then selects a task pattern that provides guidance during the task execution. In the present approach we invert the procedure since it is not a user who selects the task pattern but this is preset by the process designer. The process model determines that the generic task pattern web service is called and by the messages exchanged during runtime this service applies a specific task pattern. This means that we first have the task pattern to be executed but not yet the person who will execute the resulting task according to this pattern. To this end we need additional mechanism to identify such persons and transfer the task to them. How this can be done will be discussed later.

Whether it is preferable to implement a web service needed in a process model or process the respective step by human tasks via a task pattern, depends on a number of factors. Some of them are:

- implementation time and costs,
- urgency of the task availability,
- flexibility requirements of execution,
- complexity of the task, and
- expected execution frequency.

It depends on the character of the action of the business process, whether a machine or human actor is better suited. Human actors are often immediately available whereas the execution of process steps by web service often requires new implementations. This generally takes more time and is usually costlier. On the other hand, the implementation of a web service can help save costs in the long run, for example, if flexibility is not required and the task can be automated for more frequent execution. This decision generally depends on carefully weighing the different factors.

Implementation

The central goal of the implementation of task pattern based web services is to enrich the infrastructure for automated business process execution with semi-automatic integration of human actors. Thereby, we modify the service composition step of the business process execution infrastructure to incorporate task patterns as descriptions of tasks executed by humans instead of web services.

Below we sketch the integration of task management in a generic web service infrastructure.

The most characteristic information of a task pattern is its goal description. It explains which activity is to be performed and which results are to be achieved, e.g., which kind of document has to be written. Moreover, it describes the information or resources that are provided with a task that is generated from this pattern. This can be a specification document, a meeting room, or a customer order number. It also contains a description how the result is to be achieved, e.g., by providing a description or specific steps to be executed. Finally, it also provides a role description for a person that is able to accomplish such a task. Further details can be found in (Grebner *et al.* 2006).

The advantage and at the same time the problem of using task patterns is that everything is to be described in terms of attributes including documents and possible subtasks. This makes it more difficult to deal with the service infrastructure but, on the other hand, opens flexibility for the executors. To technically integrate the task pattern in the service composition we propose a generic task pattern web service that connects the web service infrastructure with the task management infrastructure. This web service receives information from other web services and transfers it to the task that results from the service call. The task pattern must describe how this information is finally provided to the executor of the task.

Web services can process high amounts of data at little cost, given they are used often. For example, if we consider a customer order, in which a specific customer is involved, the web service can directly retrieve the customer data and process them according to the given requirements. Applying a task pattern that results in a human task only the customer ID and a link to a system which can provide the customer data will be provided to the task executor. This can be extended by additional information about the further proceeding. The actual processing, however, must then be performed by the human executor on the basis of this information.

In this example the task pattern must be specified in a way that the generic task pattern web service extracts the customer order number from the customer order object that is transferred with the web service call and stores this number in a specific task attribute where executors find this information that they need to work on this customer order in the business system specified in the task pattern.

The users then execute these tasks in the usual task management infrastructure (Grebner *et al.* 2006) such as the SSD which allows them to organize their work in a flexible manner and start various subtasks to support the execution, e.g., to put on additional enquiries.

The generic task pattern web service is not only responsible for the preparation of data for the executor but also for the initiation of tasks triggered by web service calls. To this end the service must first identify an appropriate executor for the task and then send a task request to him. This is done by a suitable role description provided by the task pattern. Such a work distribution via roles is a usual conception in workflow management (Kumar, van der Aalst, & Verbeek 2002) and can also be applied here.

The task pattern web service must also recognize that the corresponding task is finished. The task management already has means to exchange messages between tasks to transfer information about status changes. These mechanisms can be also used for this purpose.

Example

In the following we want to discuss the example of an organization-internal software approval process as described in Figure 2. The figure shows a process model that controls the processing of requests for new software. An increased demand for flexibility comes into play at the 'security assessment' step. Here it is often difficult to decide in one step whether new software can cause security problems. The check can involve different experts who must finally agree on the issue and can become rather complicated. Therefore it might appear favorable for the process designer to enable a flexible process handling via task management at this point, which is handled by the identification of a suitable task pattern. To this end the same semantic structure for identification of suitable task patterns can be used as for web services. The process designer only has to determine in which form the information from the workflow is transferred to the respective task. Messages used by the web services must be mapped onto task attributes, related documents and so on.

When an explicit security assessment is initiated by the workflow at runtime, the set of information as defined at design time is transferred to the corresponding service. An example of such information is given in Figure 3. The individual message parts are checked by the task pattern web

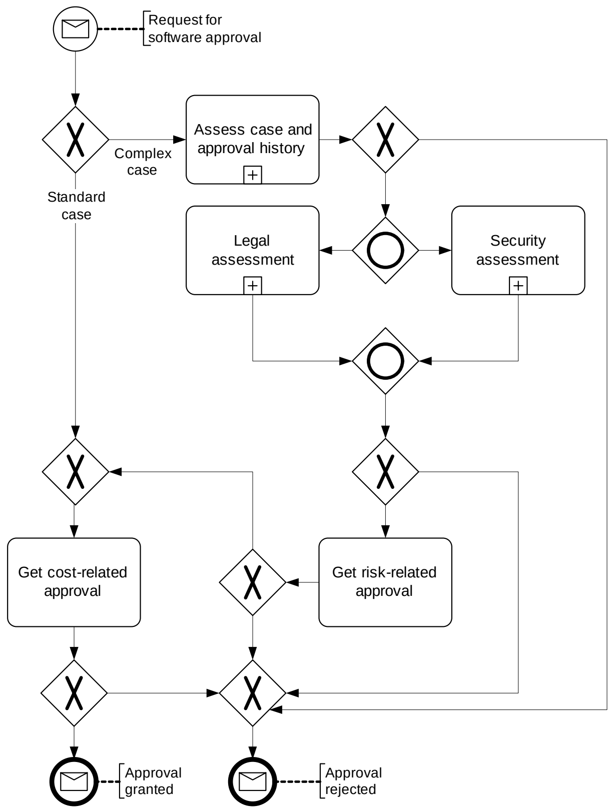
Request for software approval
Complex case
Standard case
Assess case and approval history
Legal assessment
Security assessment
Get cost-related approval
Get risk-related approval
Approval granted
Approval rejected

Figure 2: Exemplary process for software approval in an enterprise.

service and used to define a suitable task.

Thus the first message part is used to identify the task pattern that corresponds to the security check and has to be used. After the task pattern has been identified in this way the task pattern web service interprets the remaining message parts according to predefined rules. As a first step, however, a task is created that is based on the assigned task pattern, i.e., it obtains a structure such as possible subtasks and initial data such a general description for the proceeding for the human actor who works on the task (task executor). The data that describe the software to be approved are attached to the task as a document. Since the message parts are already known the designer of the task pattern can decide in which form they are to be associated to the task and provide corresponding rules.

Another message part specifies the people who are involved in the process. They can be added to the task as collaborators. Other data such as customer or partner IDs are added to the task as parameter. According to a rule defined in the task pattern the generic task pattern service uses this parameter to create a link that allows the users to access the customer or partner data in the respective enterprise system. Thus the task executors get access to additional information about these customers and partners.

Before the task can actually be performed, a task owner must be identified. This can be done by the task pattern ser-

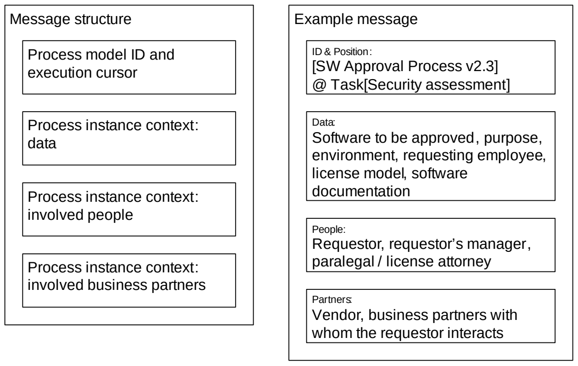
Message structure
Process model ID and execution cursor
Process instance context: data
Process instance context: involved people
Process instance context: involved business partners
Example message
ID & Position:
[SW Approval Process v2.3] @ Task[Security assessment]
Data:
Software to be approved, purpose, environment, requesting employee, license model, software documentation
People:
Requestor, requestor's manager, paralegal / license attorney
Partners:
Vendor, business partners with whom the requestor interacts

Figure 3: Messages to the task pattern handling service. Left-hand: general message structure; right-hand: an example for the 'security assessment' task of the process in Fig. 2.

vice, which may apply rules that determine the task owner. Alternatively the assignment of the task owner can be done by a human dispatcher who finds a suitable person. When the task has been transferred to the task owner the proceeding is independent of the predefined process related to the task pattern. Now the task owner can create new subtasks, and delegate these to other people or involve new persons as co-workers in the current task. The process is open and the task owner has full responsibility for orderly task processing. Even if additional rules can be introduced to restrict the action possibilities of the task owner it is exactly this feature that adds most flexibility to the approach and too restrictive limitations of the task owners' autonomy would be counterproductive. The task pattern service finally checks the task state when the task owner sets it to '*finished*'. Then the task pattern service checks whether the required postconditions are fulfilled, e.g., that a security assessment has been attached and a decision by the responsible task owner has been made. If this is the case the process can proceed in the predefined way. On an abstract level we can thus rely on certain facts after the execution of a task has been accomplished, namely that the postcondition applies. In the example we can rely on the assessment being finished, but we cannot a priori say (at process design time) whether the assessment will finish with a positive or a negative result. Unless the postcondition is fulfilled, the task remains active. For example, if the task owner does not come to a clear decision in the legal assessment, the task can be escalated to the responsible manager.

Discussion

Web service automation based on semantic technologies may become as the decisive step for the transition to more flexible of business processes. However, automated web services cannot provide a sufficient flexibility. There will be a remainder of tasks for which it is preferable that persons take care of them, especially if these tasks are complex or rarely used.

In most business processes we find a mixture of human tasks and automated services. An efficient processing, however, requires that both are closely integrated, e.g., to prevent the loss of information that is already available. Moreover, we avoid media disruptions that might lead to transformation errors.

The current approach especially aims at the integration of knowledge intensive tasks in web service compositions. This distinguishes it from other approaches such as BPEL4People (Agrawal, A.; Amend, M. et al 2007b) and other workflow extending initiatives of the workflow coalition, which try to close the gap between web services and tasks. These approaches see tasks as steps in a workflow and do not achieve the required flexibility. However, such fixed workflow tasks which have to be performed by users could be used in our approach much in the same way as services, given they are can be described using the same kind of semantic annotations.

Web services are characterized by the fact that certain specifics are taken as premise. It is this *freezing* of specifics of the processing that leads to increased efficiency. However, it is exactly this predefinition of services that causes inflexibility in the web service approach. The task pattern approach aims at enduing knowledge workers with full flexibility even at the expense of loss of system control. This means that the task management leaves it completely open to the respective user how to achieve the task goal and which additional subtasks and deviations may be introduced to this end. With the transition to the task management the user is only guided by task patterns but not controlled. However, some constraints can be checked, e.g., that some mandatory fields are filled.

From our point of view it makes sense to align both approaches. For example, the task management that we have developed in the Nepomuk project foresees the transfer of task specific information between different tasks belonging to the same pattern. This can also be realized in designed processes. On the other hand designed processes work with specific dependencies between tasks. This is something that could be included in the task pattern management, best in a light-weight form. Also, the information related to tasks in a modeled process should be seamlessly integrated in the semantic network, as this is the case in the Nepomuk task management, to make a better use of this valuable information source.

Looking for related work we find a similar idea in the *Pockets of Flexibility* approach (Sadiq, Sadiq, & Orlowska 2001). Although this approach refers to traditional workflow and not to web services it also aims at breaking predefined processing schemes by the introduction of ad hoc task capabilities.

What we generally want to achieve is flexibility at runtime. Such a vision has already been formulated at the level of web services, i.e., the dynamic discovery and invocation of web services based on semantic technologies (Haller *et al.* 2005). Runtime composition and, to a lesser degree discovery, of web services is, however, rarely applicable in an enterprise domain today (Weber, Markovic, & Drumm 2007). Discovery and composition take place on models, those models are by nature an abstraction of the reality, and with abstraction comes imprecision. While the rate of errors can probably be kept down through fine-tuning, it is highly unlikely that it can be reduced to zero with state of the art techniques. Therefore, any process that is important for a company to the least degree should not rely on automatic runtime discovery/composition of Web services. However, this technology may be applied to nice-to-have features such as search for latest news or the weather forecast.

Therefore under the current technological circumstances we can only achieve dynamic composition by the inclusion of human executors who can value the situation and decide on a suitable processing, especially in critical and complex situations. Domain ontologies play a decisive role in this respect. As Chandrasekaran et al. (Chandrasekaran, Josephson, & Benjamins 1999) have pointed out, ontologies can be considered as means of knowledge sharing. This does not only hold for knowledge sharing between different persons but in the figurative sense also between persons and services. This means that in the present approach ontologies establish the information transfer between different web services as well as between web services and persons.

Domain ontologies do not only help to compose web services in a suitable way (Hepp *et al.* 2005), they also help to find suitable task patterns. Moreover, they also help users to find information that they need for their current task, e.g., experts, documents, events and others. Both aspects are closely related and task patterns provided by users in the course of their work can also be used by those who look for task patterns to fill gaps in service compositions. In this respect we expect a fruitful exchange in both directions.

Acknowledgement

This work has in part been funded through the European Union's 6th Framework Programme, within Information Society Technologies (IST) under the SUPER and the NEPOMUK projects (http://www.ip-super.org, http://nepomuk.semanticdesktop.org), respectively.

References

Agrawal, A.; Amend, M. et al. 2007a. Web services human task (ws-humantask). http://www.activeendpoints.com/documents/documents/1/WS-HumanTask-v1.pdf. Access 22-01-2007.

Agrawal, A.; Amend, M. et al. 2007b. Ws-bpel extension for people (bpel4people). e.g., http://www.activeendpoints.com/documents/documents/1/BPEL4People-v1.pdf. Access 22-01-2008.

Cabral, L.; Domingue, J.; Motta, E.; Payne, T.; and Hakimour, F. 2004. Approaches to semantic web services: an overview and comparisons. In *Proc. ESWS*, LNCS, Vol. 3053, 225–239. Springer.

Chandrasekaran, B.; Josephson, J. R.; and Benjamins, V. R. 1999. What are ontologies, and why do we need them? *IEEE Intelligent Systems* 14(1):2026.

Grebner, O.; Ong, E.; Riss, U. V.; Brunzel, M.; Bernardi, A.; and Roth-Berghofer, T. 2006. Nepomuk Deliverable D3.1 - Task Management Model. http://nepomuk.semanticdesktop.org/xwiki/bin/view/Main1/D3-1. Access 10-10-2007.

Groza, T.; Handschuh, S.; Moeller, K.; Grimnes, G.; Sauermann, L.; Minack, E.; Mesnage, C.; Jazayeri, M.; Reif, G.; and Gudjonsdottir, R. 2007. The nepomuk project - on the way to the social semantic desktop. In Pellegrini, T., and Schaffert, S., eds., *Proceedings of I-Semantics' 07*, pp. 201–211.

Haller, A.; Cimpian, E.; Mocan, A.; Oren, E.; and Bussler, C. 2005. Wsmx - a semantic service-oriented architecture. In *Proc. IEEE International Conference on Web Services (ICWS)*, 321–328.

Hepp, M.; Leymann, F.; Domingue, J.; Wahler, A.; and Fensel, D. 2005. Semantic business process management: A vision towards using semantic web services for business process management. In *Proc. IEEE ICEBE*, 535–540.

Kumar, A.; van der Aalst, W. M. P.; and Verbeek, E. M. W. 2002. Dynamic work distribution in workflow management systems: How to balance quality and performance. *J. Management Information Systems* 18(3):157–193.

Riss, U. V., and Grebner, O. 2006. Service-oriented task management. In Calude, C. S.; Maurer, H.; Salomaa, A.; and Tochtermann, K., eds., *Proceedings of the 6th International Conference on Knowledge Management (I-KNOW'06)*, 354–358.

Riss, U. V.; Rickayzen, A.; Maus, H.; and van der Aalst, W. M. P. 2005. Challenges for business process and task management. *J.UKM* 0(2):77–100.

Sadiq, S. W.; Sadiq, W.; and Orlowska, M. E. 2001. Pockets of flexibility in workflow specification. In *ER '01: Proceedings of the 20th International Conference on Conceptual Modeling*, 513–526. London, UK: Springer.

Sinur, J., and Hill, J. B. 2006. Align BPM and SOA Initiatives Now to Increase Chances of Becoming a Leader by 2010. Gartner Predicts 2007.

Weber, I.; Markovic, I.; and Drumm, C. 2007. A conceptual framework for composition in business process management. In Abramowicz, W., ed., *Proceedings of the 10th International Conference Business Information Systems (BIS 2007)*, LNCS, Vol. 4439, 54–66. Springer.