In: Norton, Barry and Stojanovic, Nenad (Eds.). Proceedings of the 5th International Workshop on Semantic Business Process Management (SBPM2010) in conjunction with the 7th Extended Semantic Web Conference (ESWC2010), Heraklion, Greece. pp. 36-41.

AGILE PROCESS EXECUTION WITH KISSMIR[1]

Andreas Martin
Institute for Information Systems
University of Applied Sciences Northwestern Switzerland FHNW
Riggenbachstrasse 16
CH-4600 Olten
andreas.martin@fhnw.ch

Roman Brun
Institute for Information Systems
University of Applied Sciences Northwestern Switzerland FHNW
Riggenbachstrasse 16
CH-4600 Olten
roman.brun@fhnw.ch

ABSTRACT
In this paper, we describe an approach for agile business process execution and its developed prototype. In a rapidly changing environment an enterprise must be flexible and able to quickly react. Traditional business process modelling approaches are too rigid and inflexible. To achieve more agility, the modelling during build-time must be less strict and more open in a way that users are able to perform individual adaptations during run-time, which leads to more flexibility. Being able to react fast is also depending on the enterprise knowledge. Employees must be aware of it and able to access it in an easy way. The approach proposes to use ontologies to store information and appropriate services to receive context-relevant information to tackle these challenges.

Categories and Subject Descriptors
D.2.11 [**Software**]: Software Architectures - Domain-specific architectures. H.4.0 [**Information Systems Applications**]: Miscellaneous.

General Terms
Management, Design.

Keywords
Agility, Knowledge-intensive Activity, Process execution, Ontology

1. INTRODUCTION
With the accelerating innovation cycle in a globalised world, organisations need to adapt their IT solutions to the changed business requirements more frequently and unexpectedly. Agility has emerged as an important common characteristic of successful businesses. Conventional business process modelling and execution approaches have found themselves overstretched in such situations due to the lack of flexibility.

Therefore, business process models need to be agile and open for changes during execution. To achieve this, the strict distinction between build-time modelling and run-time execution is softened and process activities are represented to the users in a way that allows for individual adaptations. The implementation of this approach is described in this work at hand.

1.1 Short overview of KISSmir
KISSmir is a process handling and information sharing tool. The objective of it is to transfer and share knowledge and experience among knowledge workers in an organisation and to collaboratively learn and mature processes by executing them [1].

It focuses on the modelling of knowledge intensive business processes and aims at the following goal The individual knowledge and experience used when carrying out a process should be shared with other employees in an organisation. In KISSmir, the user should thus be able to add and exchange new knowledge and experience at any time. In order to reach these goals, business process models are made agile and open for changes during execution. The activities of the process models are represented in a way that allows end users to adjust them by attaching resources and experience deemed valuable for others in that specific context [2].

The paper is organised as follows: Section 2 describes the application scenario within which KISSmir has been applied and evaluated. Section 3 describes the architecture and Section 4 shows how it has been implemented. Then a scenario describes the four main parts in more detail before concluding the work and giving an outlook for future work.

2. Application Scenario
KISSmir is currently being applied in two scenarios – additional scenarios have already been considered but not yet realised. The first of these scenarios concerns the process of selecting students for matriculation in a master's programme at the FHNW while the second scenario is still in an early stage and supports the recruitment process at SAP Research. In this paper, only the first part of the matriculation process will be shown and explained (see Figure 1).

At the beginning of the process, a student has to send the application to the administration office. First, all of the data needed is entered by the administration office into KISSmir before the process continues with the next activity. Following the sequence flow, the sub-process 'Check Application' has to be performed with various activities. However, the execution sequence of these activities, and which of them is being executed at all, depends on the specific context of each process instance. and makes it adaptive. When finishing an activity, the user is asked whether the requirements for the particular activity are met

[1] This work has been conducted within the EU-funded Integrating Project MATURE (grant no. 216346) which investigates knowledge maturing as new form of learning in businesses and organisations, a dynamic view one-learning and knowledge management.

or not. As soon as all activities have been performed the process goes to the next activity 'Send rejection letter', 'Send acceptance letter' or 'Check Criteria List' depending on whether the requirements for an application are fulfilled or not. Hence context data used and adapted in an activity by a user is relevant for the execution of subsequent activities making a process adaptive. The process then would continue with other activities but the explained fragment is the most interesting part for KISSmir.

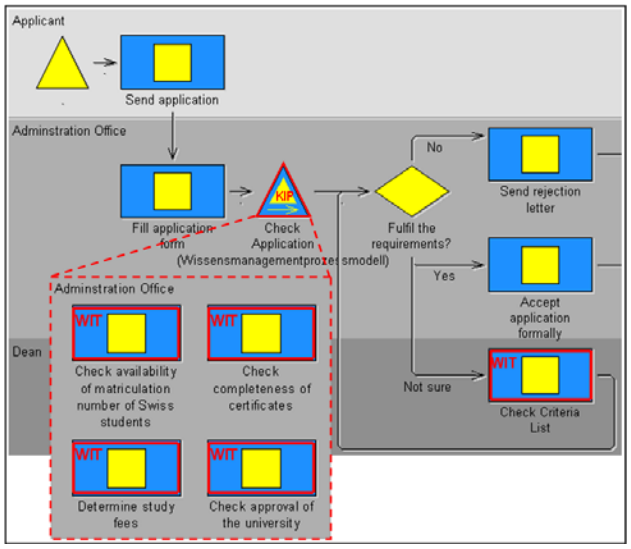

Figure 1: Matriculation process

The above application scenario has been used to evaluate the approach of KISSmir. The evaluation consisted of a first phase where the implemented approach had been explained to the end users and a guided walkthrough (including subsequent interviews) had been conducted afterwards. The results were promising and valuable feedback could be received. Based on it, KISSmir has been further developed and evaluated in a second phase, where the system has been used in a productive environment during a period of four weeks. Subsequent interviews have been conducted in order to verify the process support of KISSmir, the practical use of the adaptivity services, personal taks management, knowledge maturing and sharing and to gain new cognitions. The analysis of the interviews is currently in progress.

3. ARCHITECTURE

Figure 2 gives an overview of the architecture. The business process serves as backbone and orchestrator of the whole approach. The first activity represents a normal one, whereas we call the second activity a Knowledge Intensive Activity (KIA). KIAs can consist of sub-activities for which no sequence is defined and the execution is optional, depending on specific information for the certain process instance [3].

When an instance of an activity in the process is executed, ① it calls various services. These can be adaptivity services which endorse the agility of the process and also provide context dependent relevant resources, or application partner services which have been developed for their specific purposes. The yet implemented adaptivity services have been developed specifically for the application partner but it is intended to implement generic ones reusable in other context. As a first step a study fee calculation service has been implemented which calculates the study fee of the applicant based on the given data and influences the workflow. These services should not be mixed with the reasoning services in section 5.3 - they use the ontology for inferencing. ② A so called task pattern [4] is related to each activity. It consists of additional information such as websites that could be helpful to execute an activity, the executor of the activity, a link to the applicants application data or documents (e.g. templates) needed to execute an activity. The task pattern proposes needed resources and steps to perform an activity. However the user does not have to follow or use these suggestions. He/she is able to adapt the task pattern to his/her needs. These adaptations are stored locally so that the execution of the same activity will be proposed in the adapted way. Furthermore, the adaptations can also be shared with colleagues working on the same activities. ③ The utility web service is responsible for the instance handling and also for the management of property data. ④ An ontology is used to store necessary and helpful information for the task execution along with a relation to the activity instance, e.g. websites, experts or historical cases which might be useful in the activity execution. This information was earlier collected by web services in step one. Also related is its task pattern which includes the information as described in step two. ⑤ The communication web service is triggered. It is a service that sends an e-mail to the user which has to execute the activity. This service was developed by other partners in the MATURE project[2].

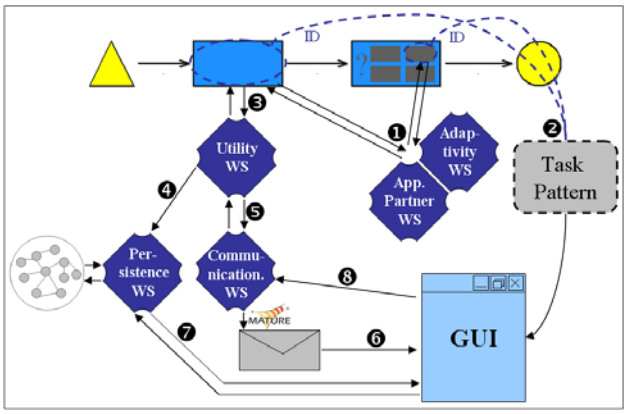

Figure 2: KISSmir Architecture overview

⑥ As soon as the user accepts the activity by clicking on a link provided in the e-mail, it is loaded into the GUI of KISSmir. In the current implementation we have used the Kasimir UI [5]. There, the user finds necessary information to execute the activity. On the one hand, information which was already defined in the task pattern during build-time. On the other hand, this information that has been collected from the previously mentioned services and stored in the ontology during run-time. ⑦ From there the information is retrieved. Some queries might not be executed until the user accepts the activity. This guarantees that the latest information is provided, for example if it takes two weeks until the user accepts the activity, it is useful to search for historical cases at the time when the activity is accepted and executed but not a priori. ⑧ When the activity has been finished,

[2] MATURE IP: http://www.mature-ip.eu/

the user terminates it by clicking the appropriate button. This triggers a return message over the communication web service back to the utility handling web service, which communicates with the running process instance.

4. IMPLEMENTATION

4.1 Overview

The modelling of an adaptive business process could be performed in a semantic modelling environment like ATHENE [6]. The activities of the processes are linked to the related task patterns and resources (files, roles, etc.) which are stored in semantic repositories ① (see Figure 3). Further on, the process model can be enhanced by adaptivity services ②. After modelling the process in ATHENE, it could be transferred to the execution framework and stored in a semantic repository ③. In the case of an automatic transformation, the model represented in a knowledge representation language like RDF/S[3] or OWL[4] could be transformed by a transformation service into a process execution language like BPEL[5] or XPDL[6].

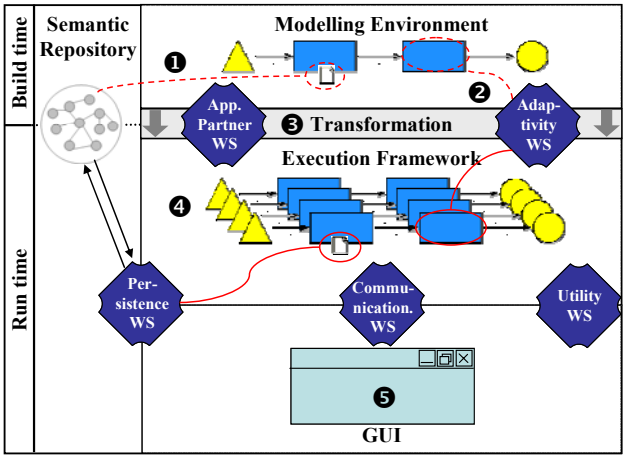

Figure 3: Architecture

The transformed process can be executed in the execution framework (e.g. BPEL workflow engine) ④. The process can access the linked resources which are stored in the semantic repositories and invokes the defined adaptivity services during run-time. The communication web service, which handles the whole instance management and communication, invokes the task GUI (graphical user interface) ⑤. The utility web service is used to define and access any kind of settings, options and preferences.

4.2 Execution framework technology stack

The KISSmir approach is implemented as follows (see Figure 4):

Data storage: KISSmir uses the open source framework Sesame[7] as semantic repository for the storage and querying of ontological data. The Sesame framework consists of a server container that runs on an Apache Tomcat[8] web server. For internal configuration and logging data, the open source database MySQL[9] is used.

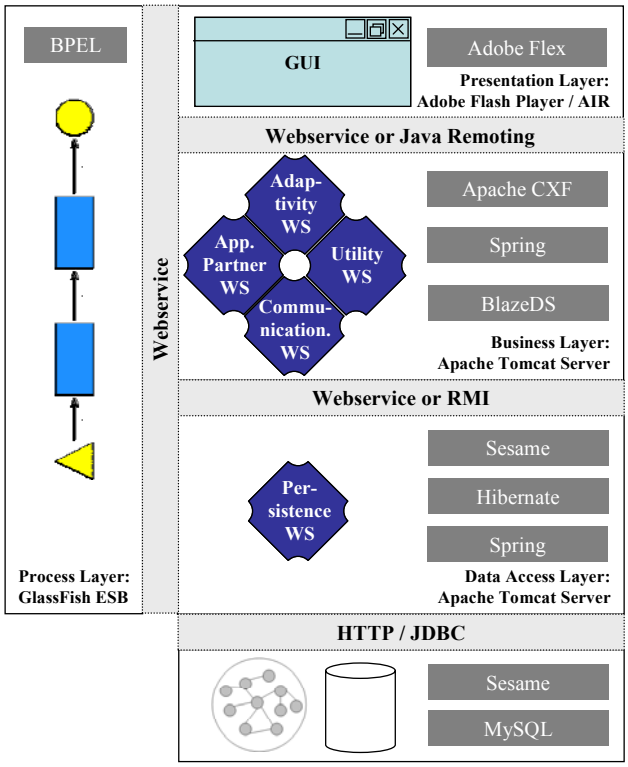

Figure 4: KISSmir technology stack

Data access layer: The data access layer is represented by the persistence web service. This service contains the data access procedures to access the KISSmir ontology using the OpenRDF Sesame Framework written in Java. There exist generic and application partner specific data access implementations. Apart from the ontological services, the logging and configuration services are implemented using the object-relational mapping features of the Hibernate[10] framework.

Business layer: The business layer contains several KISSmir web services that are explained in Section 3. These services are exposed as Enterprise Java Beans using the Spring[11] framework. The beans are deployed as JAX-WS[12] web services using the Apache CXF[13] framework. For fast and lightweight Java/Flex

[3] RDFS: http://www.w3.org/TR/rdf-schema/

[4] OWL: http://www.w3.org/TR/owl-features/

[5] WS-BPEL: http://docs.oasis-open.org/wsbpel/2.0/OS/wsbpel-v2.0-OS.html

[6] XPDL: http://www.wfmc.org/xpdl.html

[7] Open RDF Sesame: http://www.openrdf.org/

[8] Apache Tomcat: http://tomcat.apache.org/

[9] MySQL: http://dev.mysql.com/

[10] Hibernate: http://www.hibernate.org/

[11] Spring: http://www.springsource.org/

[12] JAX-WS: http://www.jcp.org/en/jsr/detail?id=224

[13] Apache CXF: http://cxf.apache.org/

remoting and messaging the open source framework BlazeDS[14] is used.

Presentation layer: The graphical user interface is implemented by using the Adobe Flex[15] rich internet application platform. This enables a fast and easy to use approach for creating application partner specific forms (see 5.1 Application handling GUI)

Process layer: The KISSmir process is described in the execution language BPEL, running on the Sun GlassFish ESB server, invoking several KISSmir web services.

We strongly focus on the ability to use the KISSmir approach in a real world scenario using existing and state of the art technologies. The mentioned technology stack gives the possibility to adapt this approach in an existing environment. The only requirement is the possibility to work with webservices.

5. SCENARIO

In this chapter the four main components of KISSmir (the application handling GUI, the BPEL process, the ontologies and the web services) are described in more detail on the basis of the implemented application scenario. For the time being the business process itself has been modelled in PROMOTE [7] using the graphical modelling tool Adonis[16], but any other business process modelling tool and notation could have been used as well. It is then transferred into a BPEL process. The Kasimir UI has been used to manage activities [5].

The sequence of the description follows the typical flow of working with KISSmir. However, the development process for an application scenario would be different sequence. After modelling the business process with the end users, the ontologies that cover the shared view of the scenario have to be defined. Next, (reasoning) web services that interact with the ontology and are used during process execution need to be implemented. Then the BPEL process has to be developed following the business process from the application scenario. The business processes often include some human activities. Therefore a GUI has to be created in order to facilitate the communication among the process instance and the user.

5.1 Application handling GUI

To serve the purpose of the application case, an Adobe Flex form has been developed which triggers the BPEL process. In the second activity of the process in Figure 1, the administrative office receives the application and needs to enter the data into KISSmir. The secretary enters all necessary data into a Flex form and attaches certificates and the letter of motivation. This information is available during the whole process execution to any user that is working on the case.

5.2 BPEL Process

There exist different approaches for business process execution, e.g. BPEL [8], XPDL [9] or YAWL [10]. For KISSmir the OpenESB[17] including NetBeans[18] IDE and GlassFish ESB server has been chosen, which includes a design and run-time environment for BPEL processes.

As aforementioned, the business process modelled in Adonis needs to be transferred into a BPEL process according to the application scenario. At the beginning of the process, necessary variables that build the context of the current instance are set. The process can be seen as a backbone from which various web services (WS) that are needed for each activity are called. For example, the activity 'Check completeness of certificates' first calls a WS to get an ID that is unique for this activity instance. Then a WS is called to receive experts according to the context (e.g. the applicant's country) and another WS is called to get historical cases (i.e. instances of the process which have already been executed and have similarities to the current instance) - again regarding the context. All the gathered information is stored in an ontology, using another WS. Then an e-mail is sent using a WS to notify the users about an activity that needs to be handled. Afterwards, the process waits for a callback (which is equivalent to the termination of an activity) and then chooses the next activity to be executed. Most of the web services call other web services, e.g. because of persistence issues or to access the ontology or property data.

In the BPEL process the instances also need to be managed. Again, web services are used to register the instances and handle the callbacks.

5.3 Reasoning Services

Having the data stored in an ontology offers many advantages. They can be listed according to the definition of Studer and colleagues [11]: A *formal* specification is machine readable (not just machine processable), can be validated, and there is no room for interpretation. *Explicit* means that concepts, functions and axioms are explicitly defined. *Shared conceptualization* indicates a common understanding about the concept, the modelled world and, last but not least, reasoning can be applied. In this section the two services for which reasoning mechanisms have been used are explained.

5.3.1 Expert Service

When a secretary is working on the application of someone coming from China, there might be some certificates attached which he/she is not able to understand as they are not (fully) translated. However, in the university there are also lecturers from China that are able to read the written Chinese language. As described in Section 5.4, the ontology also describes skills of persons. By searching in the ontology, it can be reasoned that the Chinese professor might be able to help the secretary although these two people haven't been aware of each other before. In such a way, the awareness and of usage organizational knowledge and skills can be increased.

5.3.2 Historical Case Service

It might be helpful to get information about historical cases when working on specific activities. The historical cases give information about how the activities and processes have been performed in the past and also offer an additional basis for decision making. A historical case represents a finished

[14] BlazeDS: http://opensource.adobe.com/blazeds/

[15] Adobe Flex: http://opensource.adobe.com/flex/

[16] Adonis: http://www.adonis-community.com/

[17] OpenESB: http://open-esb.dev.java.net/

[18] NetBeans IDE: http://netbeans.org/

instantiation of a process. The service looks for a historical case that is similar to the given case. There can be a similitude regarding the applicant's university or country of origin - searching for more analogies is conceivable. The service returns the historical case with all information as applicant data, attachments, decisions made, notes made on it, etc. which can be viewed by the users.

5.4 Ontologies

The entire ontology used in KISSmir can be divided into various namespaces or ontologies respectively (see Figure 5). The arrows indicate the relation from the dependent to the independent ontology. It was the objective to integrate already existing ontologies as far as possible in order not to reinvent the wheel. Therefore the ISO 3166[19] ontology has been built according to the existing standard of country names and their alpha codes. From the Dublin Core Metadata Initiative[20], the fifteen core elements (dcelements), such as title, language, creator etc. and the refinements (dcterms) as license, rightsHolder, modified, etc. have been integrated into the ontology.

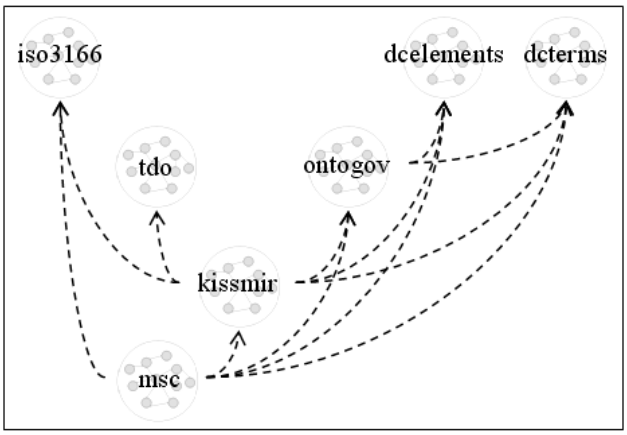

Figure 5: Ontologies and their dependencies

Next, ontologies from the OntoGov project [12] have been chosen. They are used to structure elements from the organisational domain, the process and activity domain, type elements as well as resource elements. Only concepts relevant for KISSmir have been selected. The two following ontologies have been designed specifically for KISSmir. TDO stands for 'Task Description Object'. It is used to store any information which is retrieved from the web services as described in Figure 2, step ④. This can be resourelink, weblink, expert, task pattern id, problem and solution. The kissmir ontology refines the resources from the ontogov ontology by distinguishing between knowledge resource, information resource, parts and tools. It also defines different roles that can be related to a person as executor, responsible, involved, informed and expert. Skills, which again can be related to persons, are defined and activities are further refined according to the KISS approach [3]. The msc ontology is only used for application specific needs. This means that every application partner will have to replace this ontology with their own specific concepts. MSc stands for 'Master of Science' for which persons apply in the matriculation process. It includes concepts such as the degree programme or the applicant and refines the information source concept with multiple data element concepts.

There exist different approaches for the categorization of different ontologies as the ones of Mizoguchi and colleagues [13], van Heijst and colleagues [14], Lassila and McGuinness [15] or Guarino [16] among others. Following the categorization of Guarino, iso3166, dcelement and dcterms can be classified as top-level ontologies, tdo, ontogov and kissmir as domain ontologies and msc as application ontology.

Not indicated in the overview are the instances which can be related to any ontology. The ontology has been loaded into the OpenRDF Sesame Repository[21] from where it can be accessed. A specific service has been implemented based on the given basic functionalities to create, read, update and delete data.

6. FUTURE WORK

6.1 Integration into the MATURE infrastructure

One next goal is the KISSmir integration into the MATURE infrastructure [2]. This infrastructure provides four levels of integration. KISSmir will be integrated in to the following ones:

User interface integration: The KISSmir GUI is based on the Adobe Flex rich internet application framework. This enables an easy integration into the MATURE user interface layer and the usage of intercommunication services. One benefit of the user interface layer is the possibility to use event management functionality as described in [17].

Functional integration: The demonstrator will be integrated into the MATURE infrastructure by invoking the KISSmir services over the MATURE Bus and using more MATURE services. Calls to these services can be triggered by a so-called query resource link (provided by KISSmir) or by the user who is provided with relevant recommendations for a given work situation.

Data integration: The data, stored in a Sesame repository, will be integrated by using the MATURE Sesame repository.

6.2 Mining

Process and task mining service(s) are planned to be implemented in the coming year. When implemented, the service will mine process and task instances for execution variants. Execution of knowledge intensive processes with KISSmir gives users the possibility to adjust task handling to their needs, e.g. creating subtasks or delegating tasks. Those adaptations - stored in the TDO - will be analysed. In addition, the system based selection of knowledge intensive activities will be mined: how often has one activity been chosen, has this selection been appropriate etc. [18].

7. CONCLUSION

The KISSmir approach addresses the current challenges of being flexible and agile with semantic technologies in an integrative manner. Through the usage of state of the art technologies companies will be able to react on chances more rapidly. Of course, the KISSmir approach requires an initial effort. An

[19] ISO 3166: http://www.iso.org/iso/country_codes.htm

[20] DCMI: http://dublincore.org/

[21] OpenRDF: http://www.openrdf.org/

ontology has to be created or enhanced, executable processes have to be adapted and persistence services need to be created, but the two application scenarios deliver remarkable results. The semantic based suggestions and recommendations empower the users to perform knowledge intensive activities with increased flexibility and economy of time.

8. REFERENCES

[1] C. Bradley, S. Braun, R. Brun, J. Cook, K. Hinkelmann, B. Hu, C. Kunzmann, T. Ley, A. Martin, A. Mazarakis, T. Nelkner, A. Ravenscroft, U. Riss, A. Schmidt, K. Schöfegger, B. Thönssen, and H.F. Witschel, "D2.2 / D3.2 Design and Delivery of Demonstrators of PLME / OLME and Tool Wrapper Infrastructure," 2010.

[2] V. Blažević, S. Braun, R. Brun, H. Eichner, A. Martin, and R. Woitsch, "D5.3 Infrastructure and Integrated 1st MATURE System Prototypes," 2010.

[3] D. Feldkamp, K. Hinkelmann, and B. Thönssen, "KISS – Knowledge-Intensive Service Support: An approach for agile process management," 2006.

[4] U.V. Riss, A. Rickayzen, H. Maus, and W.M. van Der Aalst, "Challenges for Business Process and Task Management," *Journal of Universal Knowledge Management, Special Issue on Knowledge Infrastructures for the Support of Knowledge Intensive Business Processes*, vol. 0, 2005, pp. 77-100.

[5] O. Grebner, E. Ong, and U. Riss, "KASIMIR – Work process embedded task management leveraging the Semantic Desktop," *'Multikonferenz Wirtschaftsinformatik' MKWI*, Berlin: 2008, pp. 715-726.

[6] K. Hinkelmann, S. Nikles, and L. von Arx, "An Ontology-based Modeling Tool for Knowledge-intensive Services," *MeTTeG07, Proceedings of the 1st International Conference on Methodologies, Technologies and Tools Enabling E-Government*, Camerino, Italy: 2007, pp. 43-56.

[7] D. Karagiannis and R. Telesko, "The EU-Project PROMOTE: A Process-oriented Approach for Knowledge Management," *Methodology*, 2000, pp. 30-31.

[8] T. Andrews, F. Curbera, H. Dholakia, J. Klein, F. Leymann, K. Liu, D. Roller, D. Smith, S. Thatte, I. Trickovic, and S. Weerawarana, "Business Process Execution Language for Web Services," 2003.

[9] Workflow Management Coalition, "XML Process Definition Language (XPDL)."

[10] W. van Der Aalst and A. ter Hofstede, "YAWL: yet another workflow language, Technical Report," *Information Systems*, 2002.

[11] R. Studer, V.R. Benjamins, and D. Fensel, "Knowledge Engineering: Principles and Methods," *Data & Knowledge Engineering*, vol. 25, 1998, pp. 161-197.

[12] B. Thönssen, L. Stojanovic, and T. Pariente, "OntoGov, Ontology-enabled e-Gov Service Configuration. Description of Ontologies, Addendum to D2," 2004.

[13] R. Mizoguchi, J. Vanwelkenhuysen, and M. Ikeda, "Task ontology for reuse of problem solving knowledge," *Towards Very Large Knowledge Bases*, 1995.

[14] G. van Heijst, A. Schreiber, and B. Wielinga, "Using explicit ontologies in KBS development," *International Journal of Human-Computer Studies*, 1997, pp. 183-292.

[15] O. Lassila and D. McGuinness, "The role of frame-based representation on the semantic web. Technical Report KSL-01-02," *Linköping Electronic Articles in Computer and ...*, 2001.

[16] N. Guarino, "Formal ontology and information systems," *Proceedings of FOIS'98*, Trento, Italy: IOS Press, 1998, pp. 3-15.

[17] T. Nelkner, *Best Practices for the Knowledge Society. Knowledge, Learning, Development and Technology for All*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[18] T. Ley, K. Schöfegger, and N. Weber, "D4.2 Maturing Services Prototype V1," 2010.