Application of Subject-oriented Modeling in Automatic Service Composition

Erwin Aitenbichler and Stephan Borgert

Technische Universität Darmstadt, Hochschulstrasse 10, 64289 Darmstadt, Germany

Abstract. Next generation SOA systems promise to enable an "Internet of Services" (IoS) - an open environment, in which every participant is free to offer and consume services. Such an IoS gives businesses the opportunity to outsource parts of their internal processes and to replace them by using external services. However, businesses must ensure that external services are compatible with their processes and that they can quickly adapt if service offering changes on the market. This raises the need for a process definition language with a formal foundation and well-defined semantics. In this paper, we discuss the suitability of different process definition languages for automatic service composition, show that subject-oriented modeling with PASS is well-suited for this domain, and how automatic service composition is implemented in the Theseus/TEXO project.

1 Introduction

Service-oriented Architecture (SOA) is an architectural style that facilitates loose coupling of components, and consequently enables flexible selection and substitution of services. However, todays SOA systems are rather closed. They are only used within the boundaries of an enterprise, or sometimes within conglomerates of enterprises with long-standing cooperations. To match the reality of Business Value Networks, current systems must evolve towards open service environments.

A *Business Value Network* (BVN) emerges from dynamic interactions of loosely-coupled organizations, which are legally distinct but economically interdependent, performing different value-creating roles (e.g., suppliers, distributors, service providers, infrastructure providers) that leverage their core competencies in order to flexibly craft optimum response to rapidly changing markets and customer demands. Value is created via dynamic exchanges of shared information and resources among these organizations engaged in complex and co-evolving processes wherein dominant players can shape the network context [1].

The term *Future Business Value Network* (FBVN) inherits this concept and stands for a conceptional framework which describes organization models with configurations of value adding collaborations within cooperative social networks among enterprises, (public) organizations, and individuals. A further characteristic is the aim to achieve a common set of goals enabled through the *Internet of Services* (or any other upcoming technology framework). FBVNs are motivated by the marching processes of outsourcing, tertiarisation, globalization, and technical innovation.

The basis for such an Internet of Services is currently developed in the large-scale Theseus Programme [1]. Building on the notion of a SOA, interacting software components can be loosely coupled and distributed over the Internet. The Theseus/TEXO platform allows for a fully decentralized service provisioning, since service consumers and providers are communicating directly with each other, in a peer-to-peer manner. The market participants are brought together by a number of central entities, such as the service marketplace and the community portal.

Such an Internet of Services gives businesses the opportunity to outsource parts of their internal processes and to replace them by using external services. In an open service market, where anybody is allowed to offer services, it seems natural that there will be numerous offers for services providing the same functionality. Hence, the customer can leverage the effects of an open market, concentrate on his core business, and save costs. However, the services offered will still be different in many details, such as their quality and how their internal processes are realized. Consequently, the customer in a B2B scenario must ensure that his overall process and the processes of external services are compatible with each other.

For example, public institutions in Germany have to stick to a clearly defined buying process. It defines how the institution has to verify the past behavior of a supplier and that he has not been blacklisted, how offers have to be invited, how offer evaluation meetings have to be organized, how offers have to be evaluated, which order approvals are necessary, how orders have to be made, and how payment is made. It also defines that suppliers cannot ask for pre-payment and must not charge shipping costs. Now, if a supplier would insist on pre-payment, the buying process would fail, because the customer is not allowed to do so.

However, the main issue with this example is the time when the process incompatibility is discovered: in the middle of the process - which is much too late. Similarly, process compatibility is an important aspect in the automotive industry. The processes for ordering components at suppliers, shipping to the car manufacturer, payment, etc. must match and all activities must be executed in the correct sequence and at the right time, such that the overall process is successful and completes in the designated time.

Consequently, one important aspect is to verify - before a service is purchased and the process is executed - that all potential messages flowing between the process participants can be handled adequately, that all activities are executed in the correct sequence, and that the process eventually terminates. In order to test process compatibility automatically, this first raises the need for a suitable process description language. Because the intricacies of how process models are described and maintained are rooted in Business Process Management (BPM), we first discuss the current state of BPM in industry and the associated mainstream process description techniques in Section 2. Next, we describe Subject-oriented BPM and the PASS language, which are the basis of our automatic service composition approach presented in Section 3. In Section 4, we present the current state of the automatic service composition implementation in Theseus/TEXO. Related work is discussed in Section 5. Finally, the paper is concluded in Section 6.

2 Current Issues in BPM

The mainstream process description languages used today lack a formal foundation and well-defined semantics (e.g., EPC, BPMN), or they are too low-level (e.g., BPEL). Consequently, such description formats do not permit computers to reason about processes. Beside these technical aspects, BPM also suffers from several other problems, as current practice in the implementation of BPM projects shows.

2.1 Lack of Process Governance

A first fundamental problem is that processes are not 'lived' as they have been designed and modeled. Practitioners report that the vast majority of decisions made in a business are still based on gut feel, intuition and experience. "We think the process works like this, so we should do X?" or "Customer orders were delayed in the past primarily because of Y, so go fix that!" [2].

If processes had initially been modeled, then the corresponding models are often not kept up-to-date. A recent study by Gartner reveals that many BPM projects will fail after implementation because the proper supporting disciplines are not implemented: "Too many user organizations are adopting BPM technologies without applying BPM disciplines via the competency center, and find that their efforts do not deliver the promised results, and their BPM initiatives are disbanded." [3]. Similarly, Forrester underlines that process support is not only about technology: "Too many organizations believe they can implement BPM with nothing more than a comprehensive set of tools and a good return on investment story" [4].

The lack of up-to-date process models also impedes the assurance of process quality, analysis of process efficiency, and process improvement. Alone the discovery of how a process actually works in a business can cause significant costs: BPM consultants claim that they spend around 40% of the project time finding out how processes in a business actually run.

2.2 One-shot Transformations

When parts of a process are implemented as software components, such as Web Services, then often so-called one-shot transformations are made. For example, the implementation starts with a business analyst creating a BPMN model of the process, which emphasizes the business aspects. However, from a technical point of view, these models are abstract, inexact, and omit many essential technical details that would be necessary to be able to execute the process directly on a computer. Next, a software engineer creates an executable BPEL process model based on the BPMN. This is either done manually or by means of automatic transformations. However, because BPMN lacks formal semantics, such a transformation can at best produce a "BPEL skeleton", which contains the structure of the process, but the engineer has to fill in all the technical details manually. Consequently, engineers transform from abstract to more concrete models and add details to the model. The relationships between model elements from the concrete to the abstract model get lost and it is not possible to automatically update the abstract model containing the business perspective.

2.3 "Outlook Processes"

Another common implementation of processes are so-called "Outlook processes". Such processes contain activities like "send email to financial accounting", meaning that an employee uses ordinary email to perform a process step. Such processes have two major drawbacks.

First, the progress of process instances cannot be monitored directly. If a customer asks for the current state of a process instance, it boils down to locating the person who had last acted on the process and asking her. While mail server products, such as MS Exchange, support message tracking, the relationships between mails and process instances cannot be discovered easily.

Second, because the email client allows unbounded communication, new communication paths between process participants can emerge easily. In reality, this changes the process, but this change will usually not make it back into the process model.

3 Process Modeling

In the following section, we discuss the requirements for a process modeling language as needed for our automatic service composition approach.

3.1 Requirements

Some important aspects of this process modeling language are:

- The process description language needs **formal semantics**. This property is needed to test the compatibility of processes, e.g., of the main process with the subprocess implemented by an outsourcing partner and to verify with formal methods that the process is correct.
- The description must have a **subject-oriented perspective**. Beside the description of *what* is done in the process, it must be clearly stated for each activity, *who* is responsible for it. This is important to decompose the overall process and to identify its constituent subjects for which services are inserted.
- The process model must be **directly executable** or it must be transformable to an executable process format without the need to manually add details to the generated model.
- The model must be **hierarchable**, i.e., it must be possible to move up and down in terms of the abstraction level. It should enable arbitrary refining or clustering of behavior without the need to leave the model. This is the key feature to eliminate one-shot model transformations.
- The language should equally support **software services** as well as **human services**.

To enforce governance, a valid process model must be in the information system that corresponds to the execution of the process in real world at all times. This is not only a requirement for automatic service composition, it is rather a general requirement to drive the next generation of BPM.

Instead of "Outlook Processes", execution platforms for the business process are needed, which do not allow interactions between participants beside the process. This forces the business to stick to the modeled process. There are no costs for discovering how the process works when it should be evaluated or made more efficient. Of course, this comes with a cost. For example, the business has to install several key users that are responsible for maintaining the process model. This approach has been successfully shown in [5].

Given these requirements, we have chosen subject-oriented modeling and the description language PASS.

3.2 Subject-oriented Modeling

Subject-orientation introduces an approach that gives balanced consideration to the actors in business processes (persons and systems as subjects), their actions (predicates), and their goals or the subject matter of their actions (objects) [6, 7]. It is based on the fact that humans, machines, and software services can be modeled in the same manner. Every one of them receives and delivers information by exchanging messages. Humans, e.g., exchanges emails, office documents, or voice messages.

3.3 PASS

The *Parallel Activities Specification Scheme* (PASS) [8] language is an implementation of subject-oriented modeling. It is founded on top of the process algebra CCS [9] (Calculus of Communicating Systems) and all language constructs of PASS can be transformed down to pure CCS. Process algebras provide a suitable means for modeling distributed systems. They offer well-studied algorithms for verification and for determining behavioral equivalences. In addition, the CCS composition operator facilitates a hierarchization and modularization of the model, allowing to handle business processes of arbitrary size. At the basic level, PASS only distinguishes between three basic types of activities: *send message*, *receive message*, and *function*.

3.4 PASS Extensions

To describe process patterns, the PASS language was extended. In contrast to regular PASS graphs, process patterns do not have to be fully connected graphs and may contain wildcard operators. Process patterns are used for service matching and their modeling differs from that of fully-specified processes in the following two aspects:

- In the model of a pattern, only activities are specified, which are essential for the process. This simplifies modeling, because the service engineer does not have to specify all functionalities and does not have to take care about each detail activity. E.g., he could omit modeling the payment branch of the process (, because its details might not be vital from a customer's point of view). If services have such branches, they would still be included, unless the engineer explicitly models the exclusion of certain behavior.

– The order of activities can be defined in a more general way as in usual process models. The wildcard operator can be used in conjunction with multiple isolated subgraphs to express a logical order between activities, instead of a single sequential order. This is useful, e.g., to enforce a certain behavior or communication pattern, while only concentrating on the essential parts of a process.

4 Automatic Service Composition

The desired result of service composition is specified by the *composition goal*. The goal consists of a description of the overall behavior, functional, and non-functional properties. In the following, we concentrate on the description of the behavior.

The behavior of the composition is described by a *fragmented* PASS process model. This model is underspecified, i.e., it only specifies the essential and basic parts of the desired process, but omits unimportant details. Consequently, it contains all subjects participating in the process and for each subject, it may contain a *process pattern* instead of a fully-specified process. Process patterns are used later to search for suitable services. We denote a model as being *fully specified*, if the behavior of all its subjects is fully specified. Models containing one or more process patterns are denoted as *fragmented models*.

Fragmented models should neither be overspecified nor be underspecified. If a pattern is overspecified, then the likelihood to find suitable services implementing this process diminishes. On the other hand, if a process is underspecified, then service candidates may bring unwanted behavior into the composition.

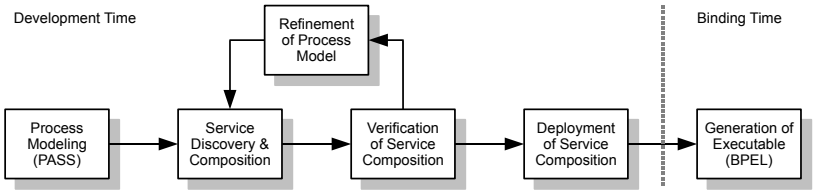

Fig. 1. Development process from fragmented model to executable process.

To specify the composition goal, we start with modeling an underspecified process and then refine it as far as needed. These steps are supported by tools. The development process is shown in Figure 1 and involves the following steps:

1. **Modeling:** In the first step, the initial fragmented PASS model is created. This model is typically underspecified.
2. **Discovery and Composition:** All suitable services are discovered according to the process patterns and constraints specified by the functional and non-functional properties. Then, a list of possible compositions is constructed.

3. **Verification:** While the compositions constructed in the previous step already match the structure of the desired process and its constraints, some compositions might still violate formal properties. In this step, each composition is entirely transformed into a CCS expression and then verified with formal methods.
4. **Refinement:** The developer inspects the service compositions found by the system in the refinement editor. Because the model is initially underspecified, the system might pull in services that expose unwanted behavior. The developer can exclude such behavior by refining the fragmented model.
5. **Deployment:** Once the fragmented model has been sufficiently refined, it can be deployed on the automatic service composition server. The server generates an executable BPEL process based on the fragmented PASS model and the candidate services. The server can also periodically repeat the discovery, composition, verification, and generation steps to take new services that appear on the market into account.

4.1 Modeling

The process is modeled using the Eclipse-based editor jPASS by jCOM1 [10]. Figure 2 shows the Subject Interaction Diagram of the TEXO EcoCalculator demonstrator.

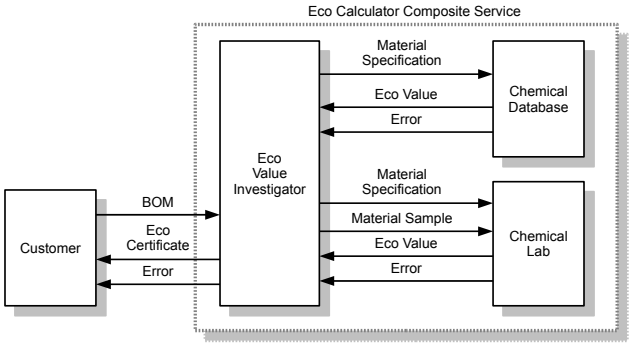

Fig. 2. Subject Interaction Diagram of EcoCalculator.

This diagram describes the relationships between subjects and the types of messages exchanged. In this scenario, a government agency establishes a new "eco label" for cars meeting certain ecological requirements. One of the requirements is a concept for disassembling and recycling and the restricted use of certain environmentally harmful materials. The service provides a compliance check and cost simulation. Its process involves the following subjects:

– **Customer:** An OEM can invoke the EcoCalculator service by sending a Bill Of Material (BOM) to the service.

– **Investigator:** The Investigator is the main part of the EcoCalculator composite service. It implements the government policy and orchestrates additional services. The functionalities Chemical Database and Chemical Lab are provided by external services.
– **Chemical Database:** Third party service that provides detailed chemical information about materials.
– **Chemical Lab:** Third party service that provides a chemical analysis of (physically) provided material samples.

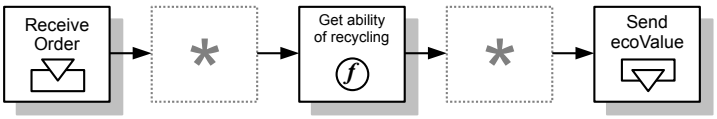

Fig. 3. Fragmented process model of the subject Chemical Database.

Figure 3 shows the fragmented process model of the subject Chemical Database. A fragmented model only describes the basic and essential parts of a process. Later, during service discovery, it is used as a search pattern to identify matching candidate services. In contrast to fully specified process, the fragmented model may contain *wildcard activities*. Such a wildcard matches any sequence of activities in the fully specified model of a service.

To make the matching of activities work, it is also vitally important that the activities in the search pattern and in the process description of a service are modeled using the same vocabulary. To ensure a consistent modeling, we introduce a so-called Activity Catalogue. It is a taxonomy of possible service functions and is based on the NACE catalogue [11] which is a statistical classification of economic activities in the European Community.

4.2 Discovery and Composition

The first step in composition is to find matching service candidates. To match the fragmented process with service descriptions, we use the programmed graph rewriting system GRL [12]. GRL stands for Graph Rewrite Library and is a Java library that provides the core functions of a graph rewriting system by supporting queries and rewrite operations. Rewrite rules are described in the declarative language GRL-RDL (Rule Description Language). GRL operates on directed, attributed graphs, whose data structures are defined by the respective application. Nodes and edges of the graph can be attributed by arbitrary Java objects. Its basic building blocks are predicates (tests) and productions (rewrite rules). Rewrite rules are specified textually. Complex attribute tests and transformations can be performed by calling Java methods from inside RDL programs. RDL programs are compiled, optimized using a heuristic, and then executed on a virtual machine. Hence, GRL provides highly efficient graph matching.

The service descriptions are used as work graphs and the goal specifications are translated into query expressions in the language GRL-RDL. To match the pattern with services, it is required that each service comes with a fully-specified PASS description. Applying graph algorithms leads to candidate lists for each specified pattern. The fragments defined in the first step are used to discover candidate services.

4.3 Verification

In order to verify the correctness of a possible service composition, the first test is to check the statical interfaces between the services. This involves the comparison of the message types exchanged between the respective services. Next, the dynamic interface is checked. This represents the communication behavior during runtime, such as the order of messages that are exchanged.

While the graphical representation is suitable for matching process patterns with service candidates, it is not suitable for verification. Hence, the PASS graphs are transformed to a pure CCS description, which is then used as input for the verification algorithms.

We currently use the CWB-NC Workbench [13] for running the verifications. CWB-NC supports various behavioral equivalences as well as model checks. Firstly, this allows us to identify services that expose equivalent behavior. At runtime, such services might be used as a replacement, in case that the original service fails. Secondly, a choreography conformance check can be performed. In a valid composition, it must be ensured that the involved services are able to communicate with each other.

4.4 Refinement

After the verification step, there may still be unwanted compositions, because the fragmented model of the service composition is initially underspecified. For each possible composition, the system now generates a fully-specified process graph by combining the process descriptions of its constituent services. The resulting process graphs are displayed in the refinement editor, where the service engineer can now annotate activities and eliminate unwanted behavior. An activity can have one of the following four states in the refinement editor:

- **required**: The activity is required and must be part of the resulting process.
- **forbidden**: The activity must not be part of the resulting process.
- **allowed**: The activity can be tolerated. It is not considered to be an essential functionality of the process.
- **unclassified**: The activity does not carry an annotation. This is the initial state of each activity.

4.5 Deployment

To determine all possible combinations of services, the first step was to discover all candidate services using process graph pattern matching. Next, these combinations were checked in the verification step and all incorrect combinations were discarded. Finally, for each valid combination, an executable BPEL process is generated, which orchestrates the constituent services.

5 Related Work

Several recent research efforts have focused on dynamic service composition techniques. Most of them are working on the execution level and extend the functionality of the BPEL standard by using proxy services or additional annotations or descriptions. A representative work is described in [14], where the authors introduce the VxBPEL language, which is an extension of BPEL by incorporating variability. It enables rebinding of services during runtime, substitution of service for optimizing purposes or in case of in sudden unavailable services. In contrast to our approach, choreographies are not supported. In addition, services provided by humans are not considered and there is a shortage of formal verification techniques.

A Petri Net is a formal language for modeling concurrent systems and has been widely accepted as formal foundation for business process modeling. Furthermore, it provides a graphical and easily understandable notation. Petri Nets are object of research for many years and current efforts are focusing on suitable constructs for automatic composition and choreography descriptions. For example, Huangfu et. al. [15] present an approach that addresses the issue of dynamic service composition by modeling service behavior by Object Petri Nets. A service consists of a set of operations and the paper introduces mapping rules from services to Object Petri Nets. A main drawback of using Petri Nets is that the entire process has to modeled in a single net. In contrast to this, we use a process algebra that supports parallelism. That allows to model each service separately and then compose them simply by using the parallel operator.

Process algebras like π-calculus provide strong means for modeling concurrent systems like service compositions and are based on formal terms. Choreography modeling, refining, and clustering are inherently supported. In addition, a rich theory to analyze processes for equivalence is provided and also the capability to perform reasoning on system properties and to verify process behavior. For this reason, current research efforts in this area focus mainly on approaches for formal verification of services and business process. Work on compliance and compatibility checks investigate the issue of when a service can be replaced by another one [16, 17]. This is necessary when a service of a process fails during runtime or for finding redundant services. COWS [18] and SOCK [19] are designed for the purpose of automatic service composition. Furthermore, process algebras are often combined with other formalisms in order to be able to specify more aspects of a service in a formal manner. E.g., some extensions exists, that combine the π-calculus with ontologies [20, 21] and formal logics [20, 21] to describe non-functional properties and access control policies. While these formal approaches are also capable of formal verification and matchmaking, they usually do not consider other aspects, such as the execution of the models, or the seamless integration of human services. In contrast to Petri Nets, process algebras lack a graphical representation.

Human-centered process modeling is another area of growing research interest. Previous approaches for automatic service composition take mostly only software-based services into account. Work on human-centered process modeling is very technology-oriented and lacks formal methods for verification [22–24]. Also, choreography is not supported, since most approaches are based on extending BPEL.

6 Conclusion

In this paper, we have presented a novel approach for automatic service composition, based on process pattern matching.

An important prerequisite for this approach is a suitable process description language. While business analysts are very comfortable with visualizing business processes in a flow-chart format, most such formats used today in industry can only establish an interoperability on the human level. This creates a technical gap between the format of the initial design of the business process and the formats for verification and execution. In contrast to this, the PASS language fulfills three important properties:

- The formal foundation based on CCS allows formal verifiability.
- Its well-defined semantics allows direct execution.
- Its graphical representation is easily comprehensible by humans.

We have extended the PASS language with constructs to describe fragmented processes. This allows an engineer to describe the goal of a service composition, while focusing only on the essential and basic aspects of the process. In addition, we have presented a method for automatic service composition based on matching such process descriptions.

Acknowledgments This work was supported by the Theseus Programme, funded by the German Federal Ministry of Economy and Technology under the promotional reference 01MQ07012.

References

1. BMWi: TEXO – Business Webs in the Internet of Services. http://theseus-programm.de/scenarios/en/texo.html (2009)
2. Lees, M.: BPM Done Right: 15 Ways To Succeed Where Others Have Failed. Software AG (March 2008)
3. Olding, E., Cantara, M.: Highlights from BPM Summit. Gartner, Inc., London (March 2009)
4. Savvas, A.: Cultural Resistance Main Cause of BPM Project Failure. Computer Weekly (March 2005)
5. Konjack, G., Heckmaier, M.: AST – Order Control Process. In: this volume. (2010)
6. Fleischmann, A., Lippe, S., Meyer, N., Stary, C.: Coherent Task Modeling and Execution Based on Subject-Oriented Representations. In: Task Models and Diagrams for User Interface Design (TAMODIA). Volume 5963 of LNCS., Springer (2009) 78–91
7. Schmidt, W., Fleischmann, A., Gilbert, O.: Subjektorientiertes Geschäftsprozessmanagement. HMD - Praxis der Wirtschaftsinformatik (266) (April 2009)
8. Fleischmann, A.: Distributed Systems: Software Design and Implementation. Springer (1994)
9. Milner, R., ed.: Communication and Concurrency. Prentice Hall PTR (1995)
10. jCOM1: Welcome to the Future of BPM: S-BPM. http://www.jcom1.com (2010)
11. NACE: Revision 2. http://ec.europa.eu/eurostat/ramon/nomenclatures/index.cfm?TargetUrl=LST_NOM_DTL&StrNom=NACE_REV2 (2010)

12. Aitenbichler, E.: Entwurf und Implementierung eines programmierten Graphersetzungssystems in Java. Master's thesis, Institut für Technische Informatik und Telematik, Johannes Kepler Universität Linz (2000)
13. CWB-NC: The Concurrency Workbench of the New Century. http://www.cs.sunysb.edu/∼cwb/ (2000)
14. Koning, M., Sun, C., Sinnema, M., Avgeriou, P.: VxBPEL: Supporting Variability for Web Services in BPEL. Information and Software Technology **51**(2) (2009) 258–269
15. Huangfu, X., Shu, Z., Chen, H., Luo, X.: Research on Dynamic Service Composition Based on Object Petri Net for the Networked Information System. Fifth International Joint Conference on INC, IMS and IDC (2009) 1075–1080
16. Wu, Z., Deng, S., Li, Y., Wu, J.: Computing Compatibility in Dynamic Service Composition. Knowledge and Information Systems **19**(1) (2008) 107–129
17. Bordeaux, L., Salaun, S., Berardi, D., Mecella, M.: When are Two Web Services Compatible. In: Technologies for E-Services. Volume 3324 of Lecture Notes in Computer Science., Springer (2005) 15–28
18. Lapadula, A., Pugliese, R., Tiezzi, F.: A Calculus for Orchestration of Web Services. In: Programming Languages and Systems. Volume 4421 of Lecture Notes in Computer Science., Springer (2007) 33–47
19. Guidi, C., Lucchi, R., Gorrieri, R., Busi, N., Zavattaro, G.: SOCK: A Calculus for Service Oriented Computing. In: Service-Oriented Computing (ICSOC). Volume 4294 of Lecture Notes in Computer Science., Springer (2006) 327–338
20. Agarwal, S., Rudolph, S., Abecker, A.: Semantic Description of Distributed Business Processes. In: Proceedings of AAAI Spring Symposium – AI Meets Business Rules and Process Management. (2008)
21. Markovic, I., Pereira, A.C., Stojanovic, N.: A Framework for Querying in Business Process Modelling. In: Multikonferenz Wirtschaftsinformatik. (2008) 1703–1714
22. Canfora, G., Penta, M.D., Lombardi, P., Villani, M.L.: Dynamic Composition of Web Applications in Human-Centered Processes. In: Proceedings of the ICSE Workshop on Principles of Engineering Service Oriented Systems. (2009) 50–57
23. Schall, D., Truong, H.L., Dustdar, S.: Unifying Human and Software Services in Web-Scale Collaborations. IEEE Internet Computing **12**(3) (May 2008) 62–68
24. Soriano, J., Lizcano, D., Hierro, J.J., Reyes, M., Schroth, C., Janner, T.: Enhancing User-Service Interaction through a Global User-Centric Approach to SOA. Fourth International Conference on Networking and Services (icns 2008) (2008) 194–203