Analysing Dependencies in Service Compositions

Matthias Winkler[1], Thomas Springer[2], Edmundo David Trigos[1], and Alexander Schill[2]

[1] SAP Research Center Dresden, SAP AG, Chemnitzer Str. 48, 01187 Dresden, Germany
{matthias.winkler,edmundo.david.trigos}@sap.com
[2] TU Dresden, Faculty of Computer Science, Institute for Systems Architecture, Computer Networks Group, Nöthnitzer Str. 46, 01187 Dresden, Germany
{thomas.springer,alexander.schill}@tu-dresden.de

Abstract. In the vision of the Internet of Services (IoS) services are offered and sold as tradable goods on an open marketplace. Services are usually consumed as part of service compositions defining complex business processes. In a service composition the execution of one service depends on other services. Thus, changes or problems during the provisioning of services may affect other services. While information about dependencies is necessary to handle problems and changes in service compositions, this information is usually only implicitly available in the process description and SLAs. In this paper, we propose an approach where the dependencies between services in a composition are analysed at design time and captured in a dependency model. This information is used to validate the negotiated SLAs to ensure that proper collaboration between the services is possible. At runtime this model can then be applied for determining the effects of events such as service failure (SLA is violated) or SLA renegotiation on other services. Our major contributions are a classification of service dependencies in business processes and an algorithm for semi-automatic dependency model creation based on a process description and the related SLAs. We have evaluated our approach based on a logistics scenario.

1 Introduction

The IoS vision aims at creating an open marketplace where services can be offered and sold as tradable goods. Services may be provided fully automatic or involve human and machine tasks. They may be composed to complex business processes and resold as composite services. The involved services cooperate to achieve a common goal. The cooperation is regulated by service level agreements (SLA) negotiated between the service provider and the service consumer. A SLA consists of a set of *service level objectives (SLO)*. A SLO refers to a key performance indicator (KPI) and specifies a value or value range and unit together with an operator to express the expected value of a KPI. Example SLOs are $max.temperature \leq 15\,^{\circ}$ and $availability \geq 95\%$.

Due to the collaboration the execution of one service in a composition has dependencies on other services. SLAs need to be negotiated in such a way that the different SLOs of one SLA do not conflict with the SLOs of another SLA. The violation or renegotiation of one SLO by a service can also influence the execution of other services in

A. Dan, F. Gittler, and F. Toumani (Eds.): ICSOC/ServiceWave 2009, LNCS 6275, pp. 123–133, 2010.

the composition or the whole composite service. Thus, information about dependencies is necessary to support the handling of problems and changes in business processes at design time and runtime. However, dependency information is only available implicitly in the process description and SLAs.

In this paper, we propose an approach for analysing dependencies between services in a composition in a semi-automatic manner at design time and capturing them in a dependency model. This model can then be applied for validating negotiated SLAs and determining the effects of events such as service failure or SLA renegotiation on other services.

The paper is organised as follows: In section 2 we introduce the background of our work and a logistics scenario. From the scenario we identify a set of dependency types. Related work is then discussed in section 3, focussing on the detection and modelling of dependencies. As the first major contribution of our paper, we analyse the identified dependency types and extract their characteristics. These characteristics are then exploited in our approach for detecting and modelling dependencies which is presented in section 5 as the second major contribution. Our evaluation is based on a case study using a logistics scenario, the results are described in section 6. We conclude the paper with a summary and an outlook to future work.

2 Logistics Use Case

In the IoS vision services from various domains (e.g. logistics) can be traded via service marketplaces. In the logistics domain not only software services are performed. For example, the service of transporting a good is provided by a truck. The cooperation of several logistic companies to deliver goods can be modelled as business process and the contractual details are negotiated as SLAs. In the following we present a scenario from the logistics domain and derive relevant dependencies.

In our scenario Trans Germany Logistics (TGL) acts as organizer of a logistics process offered as composite service Transport Germany. The service is bought by Dresden Spare Parts (DSP), which needs to ship spare parts for cars from Dresden to Hamburg. During the pre-carriage phase goods are picked up by Truck DD (pallets P1 and P2) and AutoTrans DD (pallet P3 and P4) and are transported to a warehouse where they are loaded to a large truck for further shipment to Hamburg. Three different logistics providers are responsible for picking up goods from Warehouse HH and delivering goods to the customer. The process is illustrated in Fig. 1. During the provisioning of this composite service TGL manages the underlying process.

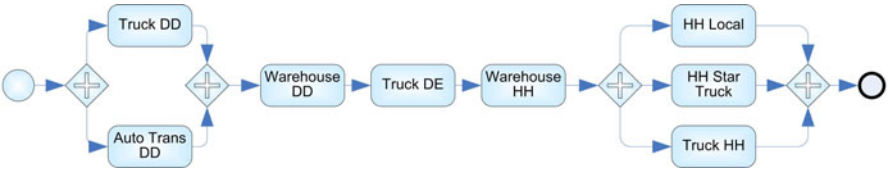

Fig. 1. Collective Goods Transport Scenario

In this use case a number of different dependencies occur. Resource dependencies occur when two services handle the same goods at different points in time. This is for example the case for Truck DD and Warehouse DD with regard to pallets P1and P2. Time dependencies occur between two interacting services when their interaction is bound to a certain time. The time for making goods available for pickup (Warehouse DD) and the pickup time (Truck DE) depend on each other. Location dependencies occur, similar to time dependencies, when two services interact at a certain location. Price dependencies occur between the composite service and its contained atomic services. The price of the composition is calculated from the price of the atomic services.

These aspects are regulated by SLAs. It is the responsibility of composite service providers to ensure that the SLAs negotiated with the different providers enable the smooth execution of the composition. If the location for delivering goods negotiated with the composite service consumer does not match the delivery location of the atomic services, problems occur. Requests for changes (e.g. renegotiating the number of goods to be transported by the composite service) or problems (e.g. delivery time cannot be met) during provisioning of one service need to be managed due to their effects on other services. If e.g. DSP changes its order and wants to ship fewer goods, TGL needs to adapt the SLAs with its subservices. To automate this process, information regarding the different dependencies needs to be captured. The different types of dependencies need to be considered.

3 Related Work

Numerous research projects are focusing on the creation of dependency models including the modelling as well as the discovery of dependencies. Dependency information is used to describe service interactions (e.g. [1]), to optimize sequencing constraints in service compositions (e.g. [2,3]), to do root cause analysis in the case of service failure (e.g. [4,5]), and to manage composite service level agreements [6]. Depending on the application domain and the purpose of the dependency information, the approaches to creating dependency models vary greatly.

Wu et al. [2] present an approach for modelling and optimizing the synchronization dependencies of activities in business processes. A synchronization model, which contains dependency information, is used to support activity scheduling in business processes. Dependency information is modelled manually. The automatic discovery of dependencies is not supported.

The MoDe4SLA [5] approach supports the handling of response time and price dependencies between composite and atomic services. The goal of the system is to support root-cause analysis. The dependency models for each type of dependency are created by a modelling approach. The discovery of dependencies is not supported.

In [4] the authors present an approach to determine the effects of service failure on other services in a composition as well as for the analysis of the root-cause of problems. The discovery of service dependencies is based on message logs. The approach assumes the execution of services as base for the dependency analysis. This is not feasible for our approach, since all dependencies need to be captured prior to service execution.

In [3] the authors use dependency information to support the scheduling of activities in business processes. They discuss the discovery of control and data dependencies

from semantically annotated business activities by evaluating their pre-conditions and effects as well as input and output parameters. This approach differs from our approach in several ways. While our resource dependencies are similar to the data dependencies of their work, we also support dependencies regarding time, location, QoS, and pricing information. Furthermore, their approach is limited to dependencies between atomic services (horizontal dependencies) while our work also supports dependencies between atomic and composite services. Another difference is that Zhou et al. use semantically annotated business activities as a base for their discovery while our work focuses on BPMN process descriptions and SLAs negotiated between services.

The COSMA approach [6] supports service providers to manage composite service SLAs. All information about the different SLAs of the composite service as well as constraints (dependencies) regarding different SLA elements are expressed within a COSMADoc document. For example, composite QoS values are calculated based on aggregation formulas from atomic service QoS values. While the aggregation formulas for the different QoS values are automatically derived from the process description, further constraints need to be added manually or from configuration files. In contrast to our work, COSMA focuses on the relationship between composite services and atomic services. Dependencies between different atomic services are not handled. Dependency types such as resource, location, and time are not covered. However, our approach to QoS and price dependency discovery is based on COSMA.

4 The Nature of Service Dependencies

Service dependencies are the underlying cause for the fact that effects of events regarding single services are not limited to these services but influence other services in service compositions. In this chapter a definition of service dependencies is provided. Different types of service dependencies are classified with regard to their occurrence in service compositions and the underlying cause of these dependencies.

4.1 Defining Service Dependencies

A general definition of the word *dependence* can be found in [7]. It is described as "the state of relying on or being controlled by someone or something else". Dependencies are an important aspect in many different scientific disciplines such as economics, organization theory, and computer science [8]. The following definition illustrates dependencies in the light of services and forms the base for our work. *A service dependency is a directed relation between services. It is expressed as a 1:n relationship where one service (dependant) depends on one or multiple services (antecedent). A service S1 is dependent on a service S2 if the provisioning of service S1 is conditional to the provisioning of service S2, i.e. if a property of service S1 is affected by a property of S2.*

4.2 Classifying Service Dependencies

In composite services dependencies occur regarding different service properties. Examples are the resources handled by services, different quality of service attributes (e.g.

max. temperature during transport), start and end times of service execution, as well as the location of service provisioning. Dependencies that occur regarding these different service properties can be classified with respect to their occurrence either between the individual services within the composition or with the service composition as a whole. We call dependencies that occur between the individual services of a service composition *horizontal dependencies* because they affect services on the same hierarchical level of composition. Dependencies can also have direct effects on the overall composition. We call these dependencies, which occur across a hierarchical level of composition, *vertical dependencies*.

Furthermore, dependencies can be classified according to the underlying cause of the dependency. In their study on coordination Malone and Crowston distinguish different classes of dependencies which occur in different disciplines (e.g. economics, computer science)[8]. *Task-subtask* dependencies occur when a process of several subservices is created as a refinement of another (composite) service. The goal of a composite service is broken down into sub-goals which are achieved by other services. Task-subtask dependencies cause vertical dependencies between the composite service and its atomic services. Two services have a *producer-consumer* relationship (dependency) when the outcome of one service is required by another service. Requirements of a service include its input resources as well as its pre-conditions. Producer-consumer relationships occur as horizontal dependencies. The execution of services may underlie certain temporal constraints between services. These constraints are called *simultaneity constraints*. Two services may need to occur at the same or at a different time. Simultaneity constraints occur as horizontal dependencies. They are often a result of a producer-consumer relationship but may also exist e.g. due to preferences of the process developer. In Table 1 an overview of the different dependency types and their classification with regard to horizontal and vertical dependencies is provided.

Quality of service: The QoS attributes of services (e.g. maximum temperature during a transportation process) describe quality aspects of service provisioning. QoS dependencies occur due to task-subtask dependencies. Thus, within a service composition QoS dependencies occur usually as vertical dependencies. Violations or changes to the QoS values of atomic services affect the composite service. Changing the composite QoS values requires a modification of all the respective atomic QoS values.

Table 1. Classification of service dependencies

Dependency class	Attribute	Horizontal	Vertical
Task-subtask	QoS	-	X
	Price	-	X
	Resource	-	X
	Start/end time	-	X
	Start/end location	-	X
Producer-consumer	Resource	X	-
	Start/end location	X	-
Simultaneity constraints	Start/end time	X	-

Price information: A price dependency exists when the price of a service depends on the price of other services. For example, the price of a composite service is calculated based on the prices of its atomic services or the prices of the atomic services are negotiated based on breaking down the price of the composite service. Price dependencies occur due to task-subtask dependencies and, thus, only occur as vertical dependencies. Renegotiating the price of an atomic service only affects the composite service.

Time information: Time dependencies express temporal relationships regarding the execution of services. They can occur as task-subtask relationships (vertical dependency) or simultaneity constraints (horizontal dependency). If the consumer of the composite service renegotiates the time when goods should be picked up, this affects the atomic services in the composition.

Location: Location dependencies occur between two services that need to be executed at the same or at a different location. Location dependencies occur as a result of task-subtask or producer-consumer relationships. Thus, there are horizontal and vertical location dependencies. If e.g. the consumer of the composition decides to renegotiate the location of delivery, this affects the atomic providers of the composition.

Resource: Two services have a resource dependency, when the availability of a resource, which is needed by one service, depends on another service. Resources include electronic data as well as material goods. Resource dependencies occur due to provider-consumer relationships as well as task-subtask dependencies. Thus, they occur as vertical or as horizontal dependencies. If the goods being transported are damaged, the SLAs for all services, which should have handled the same resource later in the process, should be adapted, because further handling is not possible. This is an example of a horizontal provider-consumer based dependency. An important aspect is that resource dependencies have a transitive nature. Thus, the failure of one service may not only affect the direct consumer but all other services which handle the same resource.

5 Dependency Model Creation

A dependency model is an explicit representation of dependency information between services. It was specified via a metamodel, which describes services and the different types of dependencies between these services. Services can play the role of a *dependant* or *antecedent*. Each dependency is described in more detail depending on its type. More details about the dependency model can be found in [9].

5.1 Lifecycle for Managing Dependencies

In order to manage the service dependencies in service compositions, a lifecycle consisting of four phases was developed (see Fig. 2). During the *Creation and Recomputation* phase the dependency model is created based on the process structure and SLA information. Information can be added or changed if conflicts are detected, SLAs change, or the process description is adapted. The *Validation* phase is necessary to ensure that the created dependency model is valid. It is also necessary to validate the negotiated SLAs to ensure that they enable proper collaboration between the different services. In the case

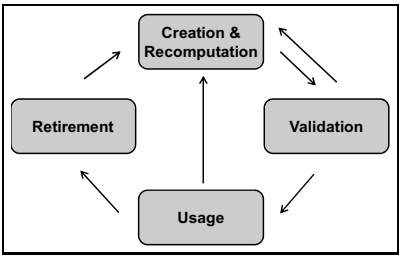

Fig. 2. Dependency Model Lifecycle

of problems either the dependency model or the SLAs need to be adapted. During the *Usage* phase the dependency model supports runtime dependency evaluation tasks such as the determination of SLO violation effects or handling SLA renegotiation requests. In the case of renegotiation the model may need to be adapted accordingly. During the *Retirement* phase the dependency model is discarded.

This paper is mainly concerned with the creation phase. The creation process of a dependency model was realized as a semi-automatic approach consisting of automatic dependency discovery and the explicit modelling of dependencies. This was necessary because it is not possible to automatically discover all dependencies based on the process and SLA descriptions. For example, it is not possible to discover time dependencies between services in parallel paths of the process or location dependencies, because there is not enough information available to discover these dependencies. Thus, it is necessary to enable the manual modelling of dependencies in addition to automatic dependency discovery. In this section the approaches to automatic dependency discovery as well as dependency modelling are presented.

5.2 Dependency Discovery

The automatic discovery of dependencies consists of two major steps, the determination of services relevant for analysis and the execution of the analysis. In order to determine the relevant services regarding which the dependency analysis is executed, all linear paths leading from the start node to the end node of a process are determined. All services, which are connected via a valid path in the process, may have horizontal dependencies. Creating all possible subpaths of a process facilitates the analysis of dependencies. For the discovery of vertical dependencies, the composite service is analysed with regard to different atomic services. Time dependencies exist between the composite service and each first or last atomic service within a path. Resource dependencies may exist between the composite service and any atomic service in a path.

As a second step the different services are analysed for dependencies. The approach for analysis of dependencies is specific for each type of dependency. Time dependencies are discovered based on the composite service process structure. Resource dependencies evaluate the SLAs of the different services. An excerpt from a simplified sample SLA is shown in listing 1 in pseudo-code notation. An example of the analysis is presented in the validation section.

```
service {
  serviceName Transport Germany
  description Transport of goods within Germany
  ..
  startLocation Teegasse 83, Dresden, Germany
  startTime 2009-02-10T10:00:00Z
  required resources P1, P2, P3, P4
  ..
  endLocation Im Tal 1, Hamburg, Germany
  endTime 2009-02-10T12:00:00Z
  provided resources P1, P2, P3, P4
  ..
  maxTemperature 35°C
  }
```

Listing 1. Sample SLA

Horizontal time dependencies: Two atomic services ($s1_{atom}, s2_{atom}$) which are directly connected via an edge in a process have a *finish-to-start time* dependency: $s1_{atom}.endTime$ finish-to-start $s2_{atom}.startTime$.

Horizontal resource dependencies: Two atomic services ($s1_{atom}, s2_{atom}$), which are directly or indirectly connected in a process, have a *resource* dependency, if two conditions are met: (1) a subset of the output resources of $s1_{atom}$ matches a subset of the input resources of $s2_{atom}$ and (2) the matching resources are not provided to $s2_{atom}$ by another service $s3_{atom}$, where $s3_{atom}$ occurs between $s1_{atom}$ and $s2_{atom}$ in this path. $s2_{atom}.inputResource$ resourceDependent $s1_{atom}.outputResource$.

Vertical time dependencies: Each first atomic service in the process has a *start-to-start* time dependency on the composite service: $s_{comp}.startTime$ start-to-start $s_{atom}.startTime$. Each last atomic service has a *finish-to-finish* time dependency on the composite service: $s_{atom}.endTime$ finish-to-finish $s_{comp}.endTime$.

Vertical resource dependencies: An atomic service has a resource dependency on the composite service regarding its input resources if (1) a subset of the composite service input resources matches a subset of the atomic service input resources and (2) the atomic service does not have a horizontal resource dependency regarding the matching resources: $s_{atom}.inputResource$ resourceDependent $s_{comp}.inputResource$.

The composite service has a resource dependency on an atomic service regarding its output resources if (1) a subset of the composite service output resources matches a subset of the atomic service output resources and (2) if the matching resources are not provided by another atomic service occuring after s_{atom} in the same path: $s_{comp}.outputResource$ resourceDependent $s_{atom}.outputResource$.

The discovery of price and QoS dependencies is based on the work of Ludwig and Franczyk [6]. A formula for calculating the respective composite service attribute value from the atomic service values is created.

Vertical price dependencies: A composite service s_{comp} has a *price dependency* on all atomic services $s1_{atom}..sn_{atom}$ which have a price. $s_{comp}.price$ priceDependency $s1_{atom}.price..sn_{atom}.price$.

Vertical QoS dependencies: A composite service s_{comp} has a *QoS dependency* on all atomic services $s1_{atom}..sn_{atom}$ which have the same QoS property: $s_{comp}.maxTemp$ qosDependency $s1_{atom}.maxTemp..sn_{atom}.maxTemp$.

5.3 Modelling of Service Dependencies

The dependency discovery algorithm produces a valid dependency model. However, there are dependencies between services which cannot be discovered. The relevant information cannot be gained from the process description or the negotiated SLAs but is available as knowledge of domain experts only. To support the handling of these dependencies, a dependency model editor enables the manual adaptation of the generated dependency model. Optional modelling steps cover complex time dependencies e.g. between services in parallel paths, location dependencies (e.g. equals, not_equals), and dependencies for QoS attributes where no automatic detection is supported, or the user wants to model a calculation formula.

6 Evaluation

As part of the validation the approach for creating dependency models was implemented. It was integrated into the Eclipse-based service engineering workbench called ISE, which was developed within the TEXO project. The automatic dependency discovery as well as the dependency editor were implemented as Eclipse plug-ins. In the current implementation time and resource dependencies are discovered. Location, price, and Qos attributes can be modelled. The implementation of QoS and price dependency discovery is not planned, since its functioning has been demonstrated [6].

Furthermore, we demonstrate the dependency model creation approach based on the logistics scenario. Due to space limitations only few samples of horizontal and vertical time dependencies are presented.

As a first step of the discovery, the composite service process is decomposed into six paths leading from the start node to the end node. Table 2 shows two of the six paths.

Table 2. Linear process paths

Truck DD - Warehouse DD - Truck DE - Warehouse HH - Truck HH
Truck DD - Warehouse DD - Truck DE - Warehouse HH - HH Star Truck

As a second step these paths are analysed regarding the different dependencies. Horizontal time dependencies of type finish-to-start are created between all pairs of services which are directly connected in the process. From the first path these pairs are e.g. Truck DD and Warehouse DD as well as Warehouse DD and Truck DE. Horizontal resource dependencies are discovered based on these paths as well. For example Warehouse DD depends on AutoTrans DD regarding pallets P3 and P4. Two examples of horizontal dependencies are depicted in table 3.

In order to analyse the vertical dependencies the composite service Transport Germany is compared with the different atomic services in the process paths. Time dependencies of type start-to-start are created between Transport Germany and the first services in the process, i.e. Truck DD and AutoTrans DD. Finish-to-finish time dependencies are created between Transport Germany and the last services in the process, i.e. Truck HH, HH Star Truck, and HH Local. Vertical resource dependencies are also discovered based on the different paths. However, their analysis is not limited to the services directly connected to the start and end node but continues until all input and output resources are matched according to our specification. An example of a vertical resource dependency is shown in table 3. AutoTrans DD depends on Transport Germany regarding its input resources (P3, P4).

In addition to the dependency discovery users can then model e.g. location dependencies or further time constraints, e.g. specifying that the two trucks in Dresden are required to pick up the goods at the warehouse at the same time. In table 3 we show a location dependency example which requires that the end location of Truck DD is the same as the start location of Warehouse DD.

Table 3. Dependencies of logistics process

Antecedent - Dependant	**Dependency**	**Description**
Truck DD - Warehouse DD	time	endTime *finish-to-start* startTime
AutoTrans DD - Warehouse DD	resource	P3, P4
Transport Germany - Truck DD	time	startTime *start-to-start* startTime
Transport Germany - AutoTrans DD	resource	P3, P4
Transport Germany - all atomic services	QoS	max(TruckDD.temperature, WarehouseDD.temperature,. . .)
Truck DD - Warehouse DD	location	endLocation *equals* startLocation

Based on this example it becomes clear that even in a relatively small use-case there are many dependencies. The semi-automatic dependency model creation process facilitates the process of capturing these dependencies.

7 Conclusion and Outlook

The explicit availability of information about dependencies between services in a service composition is important for the handling of changes requested by customers or violations of service level objectives. As illustrated in the paper, dependencies can occur vertically between a composite service and its atomic services as well as horizontally between different atomic services. Moreover, dependencies can be of different types, e.g. related to time, resources, location, price and quality of service attributes.

We have shown that these dependencies have different characteristics. Exploiting the knowledge about the characteristics of different types of dependencies we introduced a set of algorithms for discovering the dependency types relevant in logistics scenarios. The application of these algorithms is demonstrated in the evaluation section for our logistic use case.

In future work we will extend our evaluation to more complex use cases involving multiple levels of composition. Moreover, we plan to analyse the influence of process loops on the discovery of dependencies and to verify SLAs with respect to the negotiated times, locations, resources, QoS, and price based on the dependency model.

Acknowledgements

The information in this document is proprietary to the following Theseus Texo consortium members: SAP AG and Technische Universität Dresden. The information in this document is provided "as is", and no guarantee or warranty is given that the information is fit for any particular purpose. The above referenced consortium members shall have no liability for damages of any kind including without limitation direct, special, indirect, or consequential damages that may result from the use of these materials subject to any liability which is mandatory due to applicable law. Copyright 2009 by the Theseus Texo consortium.

References

1. Zhou, J., Pakkala, D., Perala, J., Niemelä, E.: Dependency-aware service oriented architecture and service composition. In: Proceedings of IEEE International Conference on Web Services, ICWS 2007 (2007)
2. Wu, Q., Pu, C., Sahai, A., Barga, R.: Categorization and optimization of synchronization dependencies in business processes. In: Proceedings of IEEE 23rd International Conference on Data Engineering (ICDE 2007), pp. 306–315 (2007)
3. Zhou, Z., Bhiri, S., Hauswirth, M.: Control and Data Dependencies in Business Processes Based on Semantic Business Activities. In: Proceedings of iiWAS 2008 (2008)
4. Basu, S., Casati, F., Daniel, F.: Toward Web Service Dependency Discovery for SOA Management. In: SCC 2008: Proceedings of the 2008 IEEE International Conference on Services Computing, Washington, DC, USA, pp. 422–429. IEEE Computer Society, Los Alamitos (2008)
5. Bodenstaff, L., Wombacher, A., Reichert, M., Jaeger, M.C.: Monitoring Dependencies for SLAs: The MoDe4SLA Approach. In: IEEE SCC, vol. (1), pp. 21–29. IEEE Computer Society, Los Alamitos (2008)
6. Ludwig, A., Franczyk, B.: Cosma–an approach for managing slas in composite services. In: Bouguettaya, A., Krueger, I., Margaria, T. (eds.) ICSOC 2008. LNCS, vol. 5364, pp. 626–632. Springer, Heidelberg (2008)
7. Miller, G.A., Fellbaum, C., Tengi, R., Wakefield, P., Langone, H.: Wordnet. WordNet Online Database (2009) (Online: accessed 21-July-2009)
8. Malone, T.W., Crowston, K.: The interdisciplinary study of coordination. ACM Computing Surveys 26, 87–119 (1994)
9. Sell, C., Winkler, M., Springer, T., Schill, A.: Two dependency modeling approaches for business process adaptation. In: Karagiannis, D., Jin, Z. (eds.) KSEM 2009. LNCS, vol. 5914, pp. 418–429. Springer, Heidelberg (2009)