2012 IEEE 16th International Enterprise Distributed Object Computing Conference Workshops

Towards a BPM Cloud Architecture with Data and Activity Distribution

Evert F. Duipmans
University of Twente
Enschede, The Netherlands
e.f.duipmans@student.utwente.nl

Luís Ferreira Pires
University of Twente
Enschede, The Netherlands
l.ferreirapires@utwente.nl

Luiz O. Bonino da Silva Santos
BiZZdesign
Enschede, The Netherlands
l.bonino@bizzdesign.nl

***Abstract*—Nowadays, many organizations use BPM for capturing and monitoring their business processes. The introduction of BPM in an organization may become expensive, because of the upfront investments on software and hardware. Therefore, organizations can choose for a cloud-based BPM system, in which a BPM system can be used in a pay-per-use manner. Opting for cloud-based solutions may normally raise concerns in organizations such as privacy, security, legal constraints and control. By combining cloud-based and traditional BPM, organizations can benefit from the best of both worlds. This paper proposes a distribution solution in which a business process is separated into individual business processes to be executed in the cloud and on-premise. This solution gives users the freedom to place sensitive data and non-computation-intensive activities within the borders of their organization, whereas less sensitive data and computation-intensive activities can be placed in the cloud. In our proposed approach, the business processes for both on-premise and the cloud are created by performing a transformation on the original business processes, guided by a distribution list in which the placement of each activity and data element is defined. This paper discusses the challenges of implementing this transformation.**

***Keywords*-BPM; cloud computing; process transformation; sensitive data; data distribution; activity distribution**

I. Introduction

Business Process Management (BPM) [1] has gained a lot of popularity in the last two decades. By applying BPM, organizations embrace a methodology for managing and optimizing their business processes. A business process consists of activities, which are performed by either humans or information systems. A Business Process Management System (BPMS) consists, amongst others, of a process engine, in which instances of a business process are coordinated and monitored. The introduction of the Service-Oriented Architecture (SOA) paradigm [2] has led to increased use of BPM, especially since the SOA paradigm provides standardized interfaces for defining services and communication between services. Consequently, executable process languages such as WS-BPEL [3], have been introduced for describing executable business processes that integrate existing services.

Purchasing a BPM system can be an expensive investment for a company. Not only the software itself needs to be purchased, but also hardware is required on which the process engine should run, and personnel needs to be hired for setting up and maintaining the hardware. In addition, scalability can be a concern for companies that use BPM, since a process engine is only able to coordinate a limited number of business process instances simultaneously. As a consequence, organizations might need to purchase additional servers, to ensure that all their customers can be served during peak load situations. Especially when these additional servers are only rarely needed, buying and maintaining the servers might become expensive.

Cloud computing [4] gives users the opportunity of using computing resources in a pay-per-use manner and perceiving these resources as unlimited. The NIST definition of cloud computing [5] mentions three service models for cloud computing: Infrastructure as a Service (IaaS), Platform as a Service (PaaS), Software as a Service (SaaS). For example, organizations may choose for cloud-based BPM systems, in which a BPM system is offered as a service (SaaS), over the Internet. Instead of having to buy hardware and software, the BPM system can be used in a pay-per-use manner. This cloud solution should also offer scalability to the organization, so that in peak load situations, additional resources can be instantiated relatively easily, and when the rush is over, the additional resources can be released. However, the fear for losing or exposing sensitive data by placing these data in the cloud is one of the biggest obstacles to the deployment of cloud-based solutions in organizations nowadays.

In this paper we investigate an architecture based on [6], in which traditional BPM is combined with cloud-based BPM. By splitting up a business process into individual collaborating processes to be executed on-premise and in the cloud, organizations can place their sensitive data and non-computation-intensive activities within the borders of the organization, whereas non-sensitive data and scalable activities can be placed in the cloud. In our approach, the original (monolithic) business process is transformed according to a user-defined activity distribution list. This gives organizations the possibility of distributing activities and data in a controlled way, depending of performance and sensitivity requirements. This paper introduces and justifies this approach and also identifies the technical challenges of automating the proposed transformation.

978-0-7695-4786-2/12 $26.00 © 2012 IEEE
DOI 10.1109/EDOCW.2012.30

IEEE computer society

The reminder of this paper is organized as follows. Section II identifies the benefits and challenges of BPM in the cloud, by identifying the general and the specific problems for each of the service models. Section III proposes an architecture in which cloud-based BPM is combined with traditional BPM. Section IV discusses the transformation we propose in our work. Section V introduces an example, on which the transformations are performed. Section VI discusses related work and Section VII draws our conclusions.

II. BPM IN THE CLOUD

Data protection is the biggest obstacle regarding cloud-based BPM. By using a public cloud-based BPM solution, the process engine, activities and all the data that is used in the process is stored in the cloud. Sensitive data protection is no longer in hands of the organization itself, but instead, the cloud provider is now responsible for guaranteeing the safety of the data. For many organizations, this is a reason for not using cloud-based BPM solutions, simply because of policies specified by the management of the organization, or because of governmental regulations that have to be followed.

Furthermore, not all the activities involved in a business process can benefit from cloud computing. Non-computation-intensive activities might even become more expensive when being placed in the cloud [6]. For example, an activity that processes some data might take longer to execute in the cloud, since the data that needs to be processed by the activity has to be uploaded to the cloud first. This activity may not only take longer to execute, even the costs can become higher, since data might need to be exchanged between the cloud and the organization, and data transfer is one of the billing factors of cloud computing.

In [7], the consequences of moving an executable business process onto one of the service models of cloud computing has been investigated. Fig. 1 shows that the responsibilities for both the cloud provider and the cloud user change when a business process is used in each service model. We give an overview of the requirements and challenges that have to be faced when moving an application to one of the service models, as identified in [7].

In the IaaS service model, the cloud user is responsible for the operating system, the middleware and the applications running in the virtual machine. Only the hardware is managed by the cloud provider. A cloud provider offers to a cloud user a virtual machine, on which the user can install an operating system and software. Each cloud user has its own virtual machine, which means that there is no sharing of virtual machines among multiple users. The installation of a process engine is performed by the cloud user, and is comparable to the installation of a process engine on-premise. In addition to on-premise installation, the cloud user might need to take additional security measures, such as

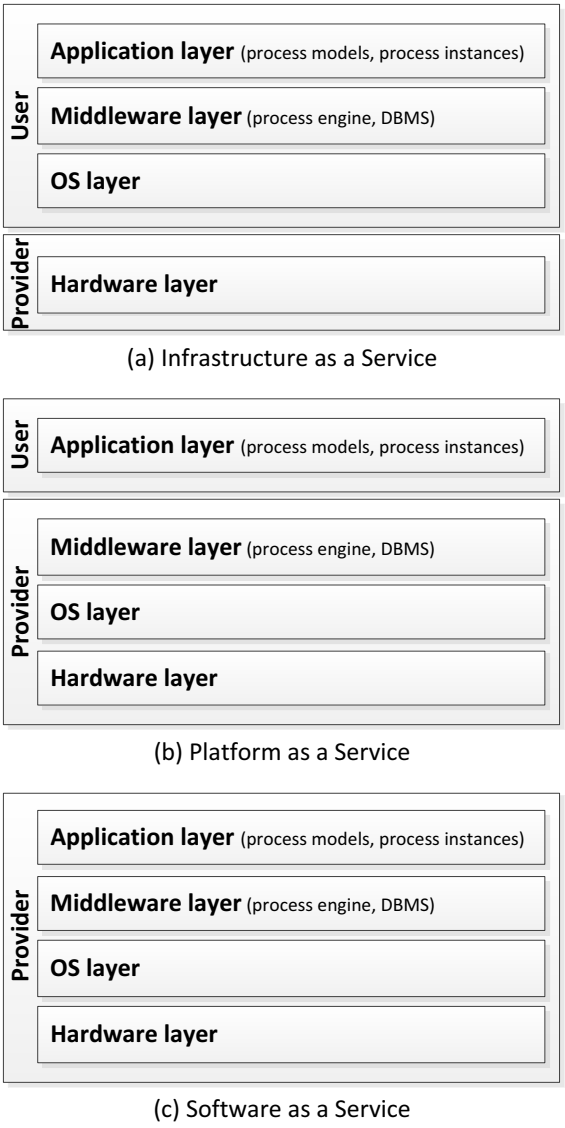

Figure 1. Responsibilities for cloud providers and cloud users for each of the service models of cloud computing, based on [7].

blocking ports, enforcing access control policies and keeping the software and operating systems up to date.

In the PaaS service model, the hardware, operating system and middleware including the process engine are offered by the cloud provider. The cloud provider is therefore responsible for the process engine. A cloud user can deploy a business process by uploading the specification of the process to the cloud. The process engine might be shared among multiple cloud users, since it is part of the platform. The responsibility of data storage and data management is no longer in hands of the cloud user, which leads to several security issues. In order to offer a secure BPEL engine in the PaaS service model, several security measures have to

be taken. Process models should be encrypted, to ensure that intruders cannot read or alter process models. By signing a process model, cloud users can ensure that a process model is only valid for a particular process engine. Whenever an intruder obtains a signed process model, this model cannot be deployed on a different engine.

Process engines often use an underlying Database Management System (DBMS) for storing the process structure and the state of each process instance. The process data stored in the database also has to be encrypted, in order to be unreadable for intruders.

In the SaaS service model, cloud providers are responsible for the hardware, operating system, middleware and the applications running in the cloud. The business process of a cloud user is no longer part of the users organization, but is instead completely offered by the cloud provider. The process is offered to multiple users and can be offered either as a single-tenant, or as a multi-tenant architecture. In a multi-tenant architecture, a business process is used by multiple users, whereas in a single-tenant architecture a process engine is deployed for each tenant and for each process model. In multi-tenant architectures, protection of data is an issue, since data of multiple users is stored in the same database. As a solution, providers can choose to store data for each user in a separate database, or to add a column to each database table where the identifier that uniquely identifies the tenant is stored.

III. Controlled Decomposition

In [6], four distribution patterns for the process engine, activities and data are identified, as shown in Figure 2. The first pattern corresponds to the traditional on-premise BPM solution and the fourth pattern is a pure cloud-based BPM solution. In the second and third pattern, the process is coordinated from either the end-user side, or from the cloud side. In our work, we identified a fifth pattern in which process engines are explicitly placed on both sides.

The architecture proposed in [6] also considers that process engines can be placed on both sides, but the decomposition of the original monolithic process onto cooperating on-site and cloud processes is not addressed in [6]. In our approach, we want to make use of two separate process engines to minimize the amount of data that has to be exchanged between the cloud and on-premise. A process engine regulates both the control-flow and data-flow of a process. Consider a process in which some activities are executed in the cloud, whereas the process engine executes on-premise. In case the output generated by an activity is the input of the next activity, when the execution of the first activity is finished, the output data of this activity is first sent to the process engine, which in turn, sends the data to the next activity. This problem is visualized in Fig. 3. By introducing a second process engine, we can avoid this problem. Activities do not have to send their data from cloud

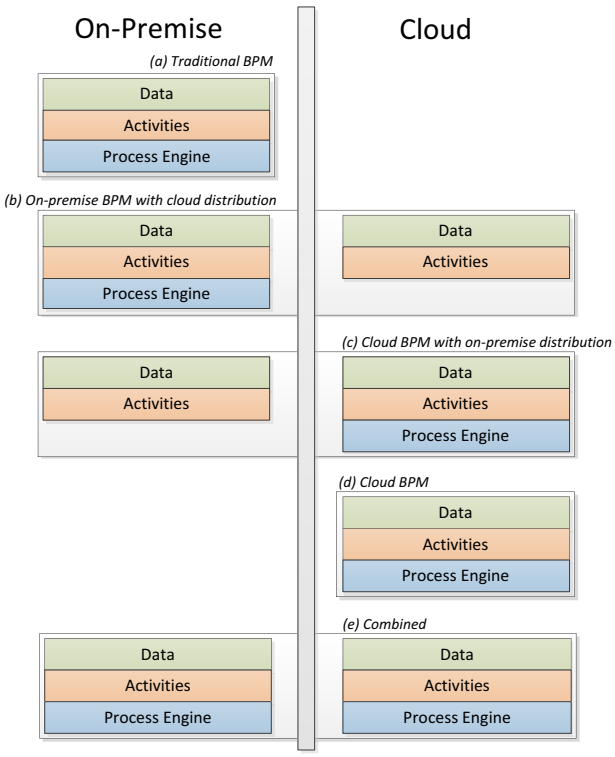

Figure 2. Distribution patterns based on [6], extended with process decomposition.

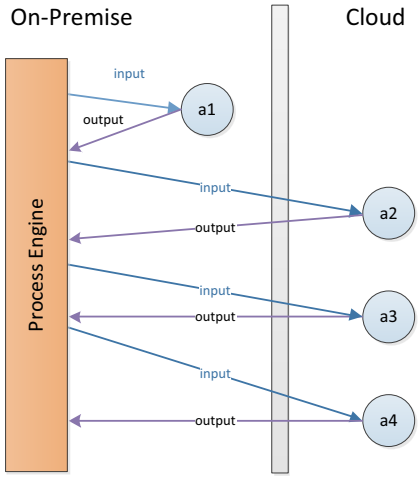

Figure 3. Schematic representation of a process with one coordinator.

to on-premise, or vice versa, since the coordination can be performed by the process engine placed on the same side as these related activities.

Using two process engines also implies that the business process should be decomposed into at least two separate business processes. Decomposition of business processes has been studied in, for example, [8, 9, 10]. The reason for

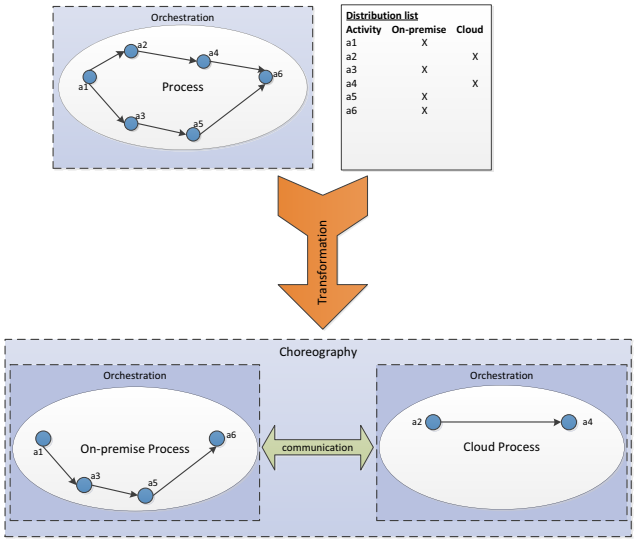

Figure 4. Schematic representation of the proposed business process transformation.

decomposition in these papers has been to reduce the amount of data that needs to be exchanged between the participants and the orchestrator. By creating separate processes for each of the participants, direct communication between participants is enforced, instead of using the orchestrator as a mediator. This leads to better data throughput and less data communication over the network. In our case, the number of processes that need to be created is fixed, and the reason for decomposition is not only the reduction of data transfer between the cloud and the organization, but also the enforcement of security rules.

We opted for a solution in which a business process is automatically transformed into separate processes. This transformation is guided by a distribution list, in which for each activity in the original process, the deployment location is defined. By using a transformation engine, we expect to obtain individual processes and a collaboration specification, in which the communication between the processes is described. Fig. 4 shows an overview of our approach.

In order to realize this approach, we need to overcome some practical issues. First, we need to define transformation rules, in order to split up the monolithic business process. The newly created processes should be formally correct, to ensure that the behavior of the decomposed business process corresponds to the original business process. Second, business process monitoring will have to consider the distribution and collaboration of the processes. Normally, a monitoring tool monitors a single business process, but since the original business process is performed by multiple cooperating but individual processes, monitoring tools should be modified to monitor the business processes as if they were a single one.

IV. Transformation

Most work on the decomposition of business processes [8, 9, 10] focus on the implementation level of the BPM lifecycle, and define transformations at the level of BPEL processes. In these references, the choice of partitioning a business process is based on performance issues and therefore can be made by implementers. The choice of distributing data and activities in the cloud, however, is not only influenced by performance issues, but also by costs and governance rules. Therefore, the distribution decision may take place in the design phase of the BPM life-cycle.

Instead of focusing on a specific language, we decided to use a more abstract model for our transformation. This gives us the advantage of purely focusing on the distribution problem, instead of also having to deal with language-specific problems, such as the ones caused by Dead Path Elimination (DPE) [3], which determines the behavior of links in BPEL, and has been identified as a big challenge in the decomposition of BPEL processes. In each of the investigated decentralization approaches, data analysis has been performed to identify the data dependencies between the activities in a process. We also need to perform data analysis in order to assess the consequences of splitting up a process.

Process transformations can be implemented using either graph transformations [11] or model-to-model transformations [12]. Model transformations are defined at metamodel level, i.e., in terms of abstract syntax. However, these transformations should result in semantically correct decompositions. In our particular case, the cooperation of the processes on-premise and in the cloud should have the same observable behavior as the original (monolithic) process. This means that either the transformations are ‘correct by construction’, or they are performed at semantic level, via auxiliary lifting and grounding transformations (from syntax to semantics and vice-versa, respectively). We propose to use an intermediate graph-based representation in which activities are represented by nodes, and control flow and data flow are explicitly modeled by the edges between the nodes. In addition, the structure should support conditional and parallel flows.

After implementing the transformation on the graph-based model, we need to define two transformations for transforming a business process defined in a certain business process modeling language into our graph-based model and back (lifting and grounding, respectively). The lifting transformation gives us the opportunity to apply our approach on real business processes models, without having to convert the business process models manually to the intermediate model.

A. Intermediate representation

Our intermediate graph representation for describing business processes consists of different types of nodes and

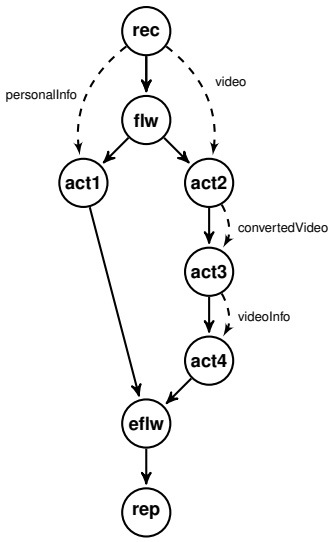

Figure 5. Intermediate representation of business process.

edges. Activity nodes represent the execution of single activities in the process. Communication nodes (invoke, receive and reply) are used to model either synchronous and asynchronous interactions between processes. The behavior of the communication nodes is comparable to the communication nodes in BPEL. Parallel behavior is modeled by using flow-nodes, which divide an execution path into several simultaneously executable branches. The end-flow node can be used for joining parallel branches. Conditional behavior is represented in the intermediate representation by if-nodes. An if-node has two outgoing branches, one which is executed in case the condition of the if-node yields true, and the other executed otherwise.

Three types of edges are available: the control flows in the process are modeled by control edges. Data flows are modeled by using data edges. A data edge pointing from activity A1 to A2 means that A2 uses data, produced by activity A1. Communication edges pass control flow and data to other processes.

Fig. 5 is an example of a business process graphically represented in the intermediate representation. In the example, control edges are represented by solid lines, data edges are represented by dashed lines and communication edges are represented by dotted lines.

B. Transformation chain

Our transformation process consists of three transformations:

1) A transformation from a business process defined in an existing business process language, such as BPEL or BPMN to our intermediate graph representation. Data flow analysis is performed on the business process to discover the data dependencies that exists between activities in the process.
2) A transformation from a business process defined in the intermediate graph representation and a distribution list, in which the deployment locations of activities and data are defined, to a a new instance of the intermediate graph model, in which nodes are distributed according to the locations defined in the distribution list. Communication between processes is modeled by using communication nodes and communication edges.
3) A transformation from the transformed intermediate representation to an executable business process and a collaborative model in which communication between the newly created processes is described.

V. EXAMPLE

In this section we give an example of a process that is decomposed into collaborating processes with different distribution destinations. The example is meant for illustrating the goal and working of our transformation framework.

Consider a television broadcast organization that is searching for new television program formats. The organization uses an on-line system, in which users can submit their new program ideas. Users need to upload a short video, in which their idea is presented to the directors. Besides the video, users also need to upload their their personal information, so that the organization can reach the user, in case the directors are interested in the submitted idea. The uploaded videos are automatically converted to a specific format, so that directors can easily watch and compare uploaded video entries, without having to convert them manually. After the conversion, an analysis process is performed, which categorizes videos and collects general properties of the video, such as length, size and quality.

The activities in which the video is converted and analyzed are computation-intensive operations and, therefore the organization wants to outsource these activities to the cloud. The organization also decides to store the videos in the cloud, since many storage resources might be needed to store all the videos. The user information needs to be stored within the borders of the organization, in order to assure privacy of user data.

We omit here the transformation from original business process to the intermediate representation and back, and only discuss how the intermediate representation and distribution list are transformed into collaborating business processes.

Fig. 5 shows the intermediate representation of the described example. The distribution list is shown in Table I. Activities *act2* and *act3* have been marked to be performed in the cloud. Since these activities are sequentially executed, both activities can be moved as a sequence to a new process. In the original on-premise process, the nodes get replaced by a synchronous invocation node, which sends the video to the cloud process and waits until the video has been converted and analyzed. The cloud process sends the result

Table I
DISTRIBUTION LIST FOR MARKING DISTRIBUTION LOCATIONS AND SETTING DATA RESTRICTIONS.

Distribution list		
Activities:	*act1*	On-premise
	act2	Cloud
	act3	Cloud
	act4	On-premise
Data restrictions:	*personalInfo*	On-premise

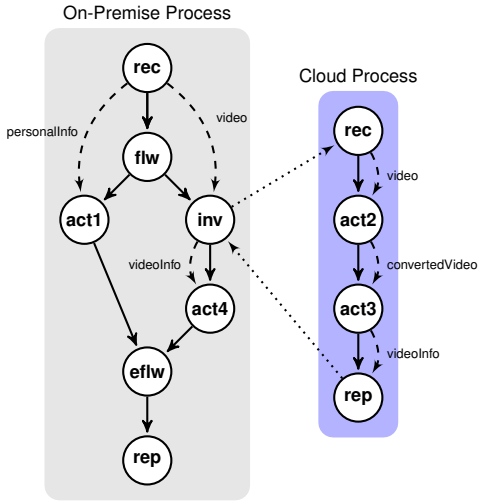

Figure 6. Intermediate representation of the transformed business process.

of the analysis back to the on-premise process, which in turn, sends the received data to activity *act4*. By performing a data verification algorithm after the transformation, we can ensure that the newly created process does not violate any data constraints that were imposed by the distribution list. The result obtained after the transformation is shown in Fig. 6. The cloud process created during the transformation is indicated in Fig. 6. The other nodes together form the on-premise process.

VI. RELATED WORK

A workflow system that scales workflow tasks on the Amazon EC2 cloud has been presented in [13]. The paper describes a workflow system in which activities can be divided among different nodes. A load balancer is used for letting the system scale up and down according to the load of the workflow system. The paper does not take into account security measures for data protection, but mainly focuses on the scalability of the system and the distribution of activities.

In a centralized orchestration, a process is coordinated by a single orchestrator. Decentralized orchestrations are distributed among several orchestrators. In centralized orchestrations, all the communication between services is performed by the orchestrator. By distributing parts of a process over separate orchestrators, the message overhead may be reduced, which leads to better response time and throughput [14]. Several research groups have investigated the possibility of decentralizing orchestrations. In [8, 14, 15], new orchestrations are created for each service that is used within the business process, hereby creating direct communication between services, instead of being coordinated by one single orchestrator. Business processes are defined in BPEL. In addition to the partitioning of business processes, also issues with synchronization and process completion with error handling are described. Data analysis is performed to identify the data dependencies between several nodes. The control dependencies and data dependencies are modeled in a Program Dependency Graph (PDG) [16]. The transformations are performed on PDGs and the newly created graphs are transformed back into BPEL. The partitioning approach is based on the observation that each service in the process corresponds to a fixed node and for each fixed node a partition is generated. In our approach we want to create processes in which multiple services can be used. This partitioning algorithm is therefore not suitable to our approach.

In [9] a transformation of a business process into several new business processes for each participant within the process has been described. The input of the described transformation is an extension to BPEL, called BPEL-D, in which datalinks between nodes are explicitly defined. In later work [17], transformation is suggested by using traditional BPEL as input and additional data dependency information, which is collected by performing data analysis on the original process [18]. These papers devote most effort to how to deal with DPE (see Section IV), when transforming a business process into multiple individual entities. The transformation described in [9] is performed manually. The rules defined in the papers are not directly applicable to our approach. However, in the final step in which the intermediate representation is transformed to an actual business process language, the rules defined in [9] can be used as guidelines for implementing processes in which faults can be propagated along the control links of the process, to ensure completion of processes during execution.

Decentralization of BPEL processes is addressed in [10]. A graph structure is used as intermediate model to describe BPEL processes. Graphs are used to model not only control flow, but also data flow. The transformation has been defined in terms of graph transformations. The paper introduces a type graph for defining graph transformations, but does not define the transformation rules. A type graph is the meta model graph, which is used to restrict transformations. The type graph presented in the paper can be used for defining our own transformation rules, since it contains the same concepts as our intermediate representation.

The authors of [19] state that [8, 9] focus too much on the BPEL language to justify their approach in which decentralization is performed at an abstract level. By performing

data analysis on business processes, they generate data dependency tables, which in turn are used for transforming these business processes.

VII. Conclusions

In this paper we propose an approach to decompose a business process into multiple processes that are deployed on-premise and in the cloud, depending on performance and sensitivity requirements expressed as annotations on activities and data. We concluded that the automation of this approach requires business process decomposition techniques, and we investigated the theoretical and technical support already available for this sort of transformation. We are currently implementing our transformation, which is expected to be a component in a comprehensive (forthcoming) tool kit that should support business process annotation, automated transformation, deployment and monitoring.

References

[1] M. Weske, *Business Process Management: Concepts, Languages, Architectures*. Springer, 2007.

[2] M. P. Papazoglou, *Web Services - Principles and Technology*. Prentice Hall, 2008.

[3] A. Alves, A. Arkin, S. Askary, B. Bloch, F. Curbera, Y. Goland, N. Kartha, Sterling, D. König, V. Mehta, S. Thatte, D. van der Rijn, P. Yendluri, and A. Yiu, "Web Services Business Process Execution Language Version 2.0," OASIS Committee, 2007.

[4] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. H. Katz, A. Konwinski, G. Lee, D. A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," EECS Department, University of California, Berkeley, Tech. Rep. UCB/EECS-2009-28, Feb 2009.

[5] P. Mell and T. Grance, "The NIST Definition of Cloud Computing," *National Institute of Standards and Technology*, vol. 53, no. 6, p. 50, 2009.

[6] Y.-B. Han, J.-Y. Sun, G.-L. Wang, and H.-F. Li, "A cloud-based bpm architecture with user-end distribution of non-compute-intensive activities and sensitive data," *J. Comput. Sci. Technol.*, vol. 25, no. 6, pp. 1157–1167, 2010.

[7] T. Anstett, F. Leymann, R. Mietzner, and S. Strauch, "Towards bpel in the cloud: Exploiting different delivery models for the execution of business processes," in *Proceedings of the 2009 Congress on Services - I*. Washington, DC, USA: IEEE Computer Society, 2009, pp. 670–677.

[8] M. G. Nanda and N. Karnik, "Synchronization analysis for decentralizing composite web services," in *Proceedings of the 2003 ACM symposium on Applied computing*, ser. SAC '03. New York, NY, USA: ACM, 2003, pp. 407–414.

[9] R. Khalaf and F. Leymann, "E role-based decomposition of business processes using bpel," in *Proceedings of the IEEE International Conference on Web Services*, ser. ICWS '06. Washington, DC, USA: IEEE Computer Society, 2006, pp. 770–780.

[10] L. Baresi, A. Maurino, and S. Modafferi, "Towards distributed bpel orchestrations," *ECEASST*, vol. 3, 2006.

[11] L. Grunske, L. Geiger, A. Zndorf, N. V. Eetvelde, P. V. Gorp, and D. Varr, "Using graph transformation for practical model driven software engineering," in *Model-Driven Software Development - Volume II of Research and Practice in Software Engineering*. Springer, 2005.

[12] S. Kent, "Model driven engineering," in *Proceedings of the Third International Conference on Integrated Formal Methods*, ser. IFM '02. London, UK, UK: Springer-Verlag, 2002, pp. 286–298.

[13] T. Dornemann, E. Juhnke, and B. Freisleben, "On-demand resource provisioning for bpel workflows using amazon's elastic compute cloud," in *Proceedings of the 2009 9th IEEE/ACM International Symposium on Cluster Computing and the Grid*, ser. CCGRID '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 140–147.

[14] M. G. Nanda, S. Chandra, and V. Sarkar, "Decentralizing execution of composite web services," in *Proceedings of the 19th Annual ACM SIGPLAN Conference on Object-Oriented Programming, Systems, Languages, and Applications, OOPSLA 2004*, J. M. Vlissides and D. C. Schmidt, Eds. Vancouver, BC, Canada: ACM, 2004, pp. 170–187.

[15] G. B. Chafle, S. Chandra, V. Mann, and M. G. Nanda, "Decentralized orchestration of composite web services," in *Proceedings of the 13th international World Wide Web conference on Alternate track papers & posters*, ser. WWW Alt. '04. New York, NY, USA: ACM, 2004, pp. 134–143.

[16] J. Ferrante, K. J. Ottenstein, and J. D. Warren, "The program dependence graph and its use in optimization," *ACM Trans. Program. Lang. Syst.*, vol. 9, no. 3, pp. 319–349, Jul. 1987.

[17] R. Khalaf, O. Kopp, and F. Leymann, "Maintaining data dependencies across bpel process fragments," in *Proceedings of the 5th international conference on Service-Oriented Computing*, ser. ICSOC '07. Berlin, Heidelberg: Springer-Verlag, 2007, pp. 207–219.

[18] O. Kopp, R. Khalaf, and F. Leymann, "Deriving explicit data links in ws-bpel processes," in *Proceedings of the 2008 IEEE International Conference on Services Computing - Volume 2*, ser. SCC '08. Washington, DC, USA: IEEE Computer Society, 2008, pp. 367–376.

[19] W. Fdhila, U. Yildiz, and C. Godart, "A flexible approach for automatic process decentralization using dependency tables," in *ICWS*, 2009, pp. 847–855.