Towards BPEL in the Cloud: Exploiting Different Delivery Models for the Execution of Business Processes

Tobias Anstett, Frank Leymann, Ralph Mietzner, and Steve Strauch

Institute of Architecture of Application Systems,
University of Stuttgart, Germany
{lastname}@iaas.uni-stuttgart.de

BIBTEX:

```
@inproceedings{AnstettLMS2009,
  author    = {Anstett, T. and Leymann, F. and Mietzner, R. and Strauch, S.},
  title     = {Towards BPEL in the Cloud: Exploiting Different Delivery Models
               for the Execution of Business Processes},
  booktitle = {Proceedings of the International Workshop on Cloud Services
               (IWCS’09) in conjunction with the 7th IEEE International
               Conference on Web Services (ICWS’09)},
  year      = {2009},
  pages     = {670--677},
  publisher = {IEEE Computer Society},
  doi       = {10.1109/SERVICES-I.2009.32}
}
```

Towards BPEL in the Cloud: Exploiting Different Delivery Models for the Execution of Business Processes

Tobias Anstett, Frank Leymann, Ralph Mietzner, Steve Strauch
Institute of Architecture of Application Systems, University of Stuttgart, Germany
{anstett, leymann, mietzner, strauch}@iaas.uni-stuttgart.de

Abstract

More and more companies are outsourcing parts of their business processes to third party providers to exploit the expertise and economies of scale of these third party providers. In the IT field, emerging delivery models for software such as Software as a Service and cloud computing offer the possibility to outsource applications and computing infrastructure and thus enable enterprises to focus on their core competences. In this paper we investigate how the new delivery models affect the outsourcing of business processes modeled in WS-BPEL. WS-BPEL is the standard to model and execute business processes in Web service-based IT landscapes. We describe how security and trust issues affect the execution of BPEL processes in the cloud and show the requirements on the middleware supporting the execution of BPEL processes.

1. Introduction

Enterprises today are faced with the challenge to rapidly react on ever changing market conditions. As enterprises grow, shrink, merge and transform, they constantly need to bring new products and services to market to stay competitive. As most operational procedures in an enterprise today are supported by IT systems, the IT systems must be flexible and adaptable enough to cope with these challenges. Often traditional monolithic application silos are considered to be of major hindrance for the flexibility of an enterprise. Big legacy applications hosted in an enterprises datacenter (*on-premise*) are often not build with integration and flexibility in mind. In the last years the service oriented architecture (SOA) has emerged as an architectural style allowing to build applications out of reusable components (services). These services are explicitly designed to be reused by other applications. Such applications are often called *composite applications*. One technology stack to realize an SOA is the Web service stack. Individual services are realized as Web services and expose an external interface described in WSDL[1]. The prominent standard to recursively compose Web services into higher-level Web services is the Web Service Business Process Execution Language (BPEL for short, [24]). BPEL allows to describe the control flow that is needed to *orchestrate* a set of services into a meaningful business process. Additionally BPEL allows via so-called *assign* activities to describe how data is passed between individual activities in the business process.

A business process orchestration language such as BPEL is widely regarded as beneficial for the flexibility of an application [18] as it allows to change the orchestration logic (described in BPEL) independently from the services. Additionally it fosters the modularization of application functionality into services which allows the reuse of these services in new applications.

In addition to reuse services from their own enterprises the encapsulation of functionality in services fosters application developers to make traditional "make or buy" decisions. As the invocation of external services becomes more or less transparent through the use of Web service technology enterprises can focus on a "best of breed" strategy by choosing the best service out of a variety of services in their own enterprise as well as at third party providers.

In addition to the outsourcing of services fostered by a SOA, new delivery models for software have been emerging. Enterprises more and more try to move away from traditional on-premise applications that are hosted and run in their own datacenters. As with traditional outsourcing scenarios in other fields such as manufacturing, companies try to focus on their core competences. As a consequence different outsourcing models have emerged, from outsourcing the infrastructure, over outsourcing of middleware components to outsourcing of whole applications.

[1] Web Service Description Language

978-0-7695-3708-5/09 $25.00 © 2009 IEEE
DOI 10.1109/SERVICES-I.2009.32

In this paper we investigate how the two aspects (BPEL and new delivery models for software) combine. We therefore begin with background and related work on BPEL and outsourcing using BPEL. We then give a brief overview over different delivery models for software and hardware which can be used to run BPEL processes. Motivated by a running example (Section 3) we investigate the challenges for BPEL in the cloud in Section 4. We show how the different delivery models influence the design and execution of business process models as well as supporting runtime infrastructure (such as BPEL engines). We show how these challenges can be solved with existing engines or where existing engines need to be adapted or extended. We give a summary of the findings and the necessary modifications to BPEL engines in Section 5 and finish with a conclusion and outlook in Section 6.

2. Background and Related Work

In the last years more and more approaches to facilitate the modeling of business process have been introduced. From reference processes such as RosettaNet PIPs [14] or configurable EPCs [26], to configurable BPEL processes [16, 17] and configurable applications [22] and many more such as process fragments [20]. These approaches focus on the modeling aspect of business processes and support the modeler by offering pre-configured processes templates. Thus, the modeler does not need to reinvent the wheel and can reuse process model templates to create new processes. Other approaches deal with the splitting of BPEL process models across various engines [15]. However, supporting modelers in the creation of new processes out of predefined templates is only one aspect in facilitating business process modeling and execution. The second part is the advent of new delivery models for software. Several approaches and frameworks exist on how to offer applications as a service [6, 8] or how to automatically provision applications and related infrastructure [13, 19]. Salesforce.com[2] as a prominent example of a software as a service application allows users to customize business processes in the application, while coghead[3] allowed users to create their own business processes based on a BPEL engine. These approaches are geared toward the support of business processes in the cloud as they allow users to customize predefined applications or processes and deploy them on a provider's infrastructure. This notion is supported by other delivery models that have been introduced in the area of cloud computing in the last years. We identified the following three categories of delivery models namely infrastructure as a service (IaaS), platform as a service (PaaS) and software as a service (SaaS), but there are also other classifications available [4].

Infrastructure as a service provides the basic infrastructure to the customer. The customer requests and rents required hardware from the IaaS provider and has to take care for configuring the platform and application, i.e. installation of operation system and required software components, security configuration, etc. Amazon Web Services[4] especially Amazon Elastic Compute Cloud (Amazon EC2) as a prominent example of an infrastructure as a service application allow customers to hire required hardware components. The user employs already existing Amazon Machine Images (AMIs) or creates custom AMIs before deploying his Amazon EC2 instance.

The PaaS model offers both, the infrastructure as well as the platform to deploy applications. The user does not have to take care neither for reserving hardware resources nor for configuring the platform. Google's App Engine[5] is a well-established example of a platform as a service application.

Software as a service is a delivery model, which provides different customers the functionality of an application that is completely hosted in the cloud. The user does not have to worry about the required hardware resources, software components, deployment of the application, etc. SaaS allows different users the customization of the application, i.e. regarding the presentation, logical and database layer. This is done through customer profiles. One of the characteristics commonly required by SaaS applications is multi-tenancy. Multi-tenancy means that multiple customers (tenants) are served concurrently by one or more hosted application instance. Generally there are two multi-tenancy patterns: *multiple instance* and *native multi-tenancy* [8]. Multiple instance mean that for each tenant a separate instance is deployed. Native multi-tenancy means that all tenants are served by one native single possibly clustered instance of the application.

When talking about and comparing delivery models of cloud computing the customer's point of view, e.g. user interfaces, customizability, etc. as well as the provider's point of view, e.g. provision of hardware resources, virtualization, etc. should be taken into consideration [4,5]. Figure 1 shows an overview of the three delivery models described before.

The flexibility for the customer increases from SaaS (lowest) to IaaS (highest), because the raised effort for setting up the platform as well as the applications by himself also induces the possibility of exerting influence on

[2] http://www.salesforce.com

[3] http://www.coghead.com/

[4] http://aws.amazon.com

[5] http://appengine.google.com

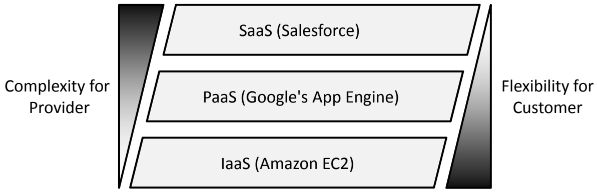

Figure 1. Overview of delivery models

selection of the middleware and application components. By contrast the complexity for the provider decreases from SaaS (highest) to IaaS (lowest). The comparison of the different delivery models will be discussed in more detail in Section 4.

Right now there is no approach enabling execution of business processes through cloud computing, because the delivery models are not geared specifically at offering BPM in the cloud and that is where our approach kicks in.

3. Running Example

We introduce the example of a fictional travel agency named Paradise Travel, which offers travel booking through a Web interface to potential customers. The offering of Paradise Travel comprises complete travel booking including transport to holiday resort, hotel booking as well as optional car rental service. The whole travel booking process is modeled as business process in BPEL. External partners involved in the business process, e.g. several car rental agencies located at the different available travel destinations as well as the credit investigation company for checking solvency of customer before credit card payment, are contacted through Web services.

Currently the infrastructure and the platform required for Paradise Travel, e.g. an application server including deployed Web services as well as the business process engine including deployed travel booking process, are hosted in an enterprise datacenter owned and maintained by the Paradise Travel company (*on-premise*).

4. BPEL in the cloud

In this section we will take a look on how to outsource several parts of Paradise Travel's business by applying the delivery models IaaS, PaaS and SaaS. Along the way we will consistently refer to the example described before.

4.1. IaaS

Although outsourcing the hosting and maintenance of the infrastructure required for the business of Paradise Travel to an IaaS provider reduces the costs for the travel booking company, the setup of the application server and business process engine as well as thedeployment of Web services and travel booking business process is still the task of Paradise Travel. So applying IaaS is the first step to concentrate on the core business.

4.1.1. Providing BPEL through IaaS. In terms of the requirements for a BPEL engine, IaaS (Figure 2) is very close to the traditional on-premise model. Customers have to make the same decisions regarding the installation of software such as *operating system*, *platform middleware* and *application*. Of course, this decisions must comprise security considerations such as blocking out attackers by locking ports, patching the operating system, running an anti-virus software, etc., as well as configuration and enforcement of access control policies.

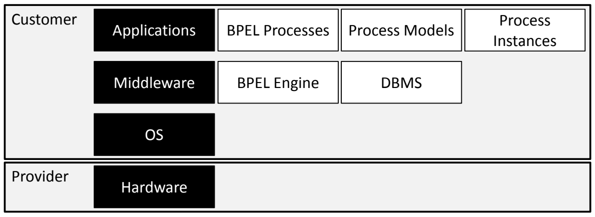

Figure 2. IaaS

In essence, IaaS realizes on-premise in the cloud by moving the responsibilities for hosting the infrastructure (e.g. *hardware*) from their own datacenters to an outsourcing provider. Thus, there are no special requirements or challenges that must be solved to be able to provide BPEL through IaaS, except the installation of a BPEL engine.

4.2. PaaS

Applying PaaS for outsourcing most of the tasks not part of the core business of Paradise Travel, e.g. hosting and maintenance of infrastructure as well as platform middleware, reduces the complexity for Paradise Travel compared to IaaS. On the one hand it is still the task of the travel booking company to provide the Web services to be deployed on the application server as well as the travel booking process to be deployed on the process engine. On the other hand this is not only a disadvantage but also the possibility to keep a part of the flexibility of the travel booking company by enabling the adaption of the business process, in case the needs of the customers concerning travel booking changed or in case the agency providing hotel information and hotel reservation is no longer available and another provider has to be found.

4.2.1. Providing BPEL through PaaS. In contradiction to the IaaS deployment model, PaaS providers host *hardware*, *operating system* and *platform middleware* (Figure 3) such as a BPEL engine and a database management system (DBMS).

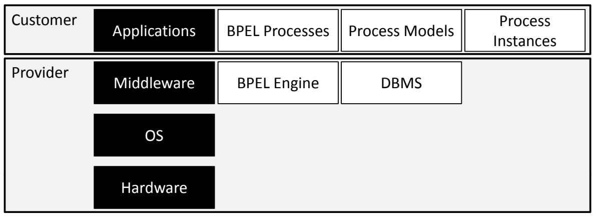

Figure 3. PaaS

Because everything except the business process model is hosted, the customer must trust his provider on the basis of external audits or security certificates, having expertise in protecting the system at hardware and operating system level.

Although a customer may trust the provider at the hardware or operating system level, one of the main show-stopper is the perceived lack of data confidentiality by means of data loss or even sellout of customer data. For example the on-demand cloud computing service FlexiScale[6] has been offline for several days because an employee accidentally deleted one of the main storage volumes [23].

Two main distinctions between the forms of data hosted at an outsourcing provider must be made: The first form is data, which describes the business process itself such as process models. The second form is the data that is processed by the business processes, such as customer information and order processing data. Because administrators can simply gain access to the business process models or the business process instance data in the underlying databases, it becomes easier to "steal" the assets of the enterprise.

To increase customers' confidence and trust in outsourcing and the provider itself, unauthorized disclosure must be impossible by design. Therefore we propose an architecture of a secure BPEL engine which ensures that the information about an asset of the enterprise such as process models and customer data is not inexpediently used. In the following section the security requirements to prevent unauthorized disclosure of data are analyzed and a possible realization in terms of requirements to a secure BPEL engine architecture is presented.

4.2.2. Requirements and Challenges. Business process management (BPM) is a evolutionary process that involves modeling, implementation, execution, monitoring, assessment and re-design of business processes. When outsourcing processes to a PaaS provider, the provider is responsible for the execution and partially monitoring of these processes. Before a business process can be executed, the process model has to be deployed to the provider's engine. The deployment process includes uploading of the process model and its final installation to the process engine. To ensure confidentiality and integrity of the enterprise assets that are implicitly reflected by the process model and process instance it is necessary to realize the following security requirements:

- It must not be possible to read the process model for somebody who gets hold of a process model description file such as a BPEL file.
- It must not be possible to alter the process model (and if yes, find out that it has been done)
- It must not be possible to deploy the process model on another (maybe corrupted) engine

Therefore a means to encrypt and sign parts of processes or complete processes is needed. These kinds of processes are further referred as *obfuscated processes*. Execution of an obfuscated process requires a modified process engine that implements additional features such as a public key infrastructure (PKI). Thus, each engine has a unique private key and only accepts obfuscated processes that are encrypted or signed using one of the provided public keys. Others are rejected. The private key of the engine is unknown even to the administrator.

Employing a PKI, the engine must provide the necessary interface to retrieve the public key that can then be used to encrypt the process, or publish it to a certification authority such as VeriSign[7], which can furthermore certify that the provider uses a secure BPEL engine. Although the PKI implicitly protects processes of being deployed to an unintended engine, it does not yet detect possible tampering with the engine through the provider. It must not be possible to corrupt the process or engine to log information in an undesired way. Therefore the compliant engine must be self-signed with its private key to detect modifications to its source.

Since BPEL is based on XML we propose to use the two W3C Recommendations XML-Encryption [30] and XML-Signature [31] to ensure secrecy, integrity and authenticity in BPEL processes. To apply these

[6] http://www.flexiscale.com/

[7] http://www.verisign.com/

requirements to a BPEL engine such as Apache ODE[8], the existing deployment component has to be changed. The deployment component must be able to decrypt obfuscated process models in order to validate them before saving them to database.

A general problem arising from the architecture of a BPEL engine is the underlying DBMS. BPEL engines use relational DBMS to store process model and process instance information and thus administrators can use the DBMS as back-door to access these assets. Therefore the following security requirements must be taken into account:

- It must not be possible to derive or reconstruct the process definition of a process by gaining access to the DBMS.
- It must not be possible to access process instance data such as credit card information of customers by gaining access to the DBMS.

Workflow engines distinguish between buildtime, runtime and audit databases [18]. The buildtime database defines the metamodel in other words the structure of process models. Runtime databases are used to create concrete instances of the process models stored in the buildtime database, for navigation and controlling the execution of the process instance and its accumulating data. Due to laws, regulations or just to enable monitoring, audit databases also referred as audit trails or event log databases, record the actual execution of process instances as set of traces.

The buildtime database contains process models in an easily reconstructable representation and thus provides a huge security hole for a possible theft. Assumed that the information contained in the buildtime database is protected, process models can be stolen either way. Process mining techniques [3, 28, 29] can be used to reconstruct the process model out of the process instance data and the execution traces contained in the runtime database or the audit database. This approach requires some basic understanding of process mining and maybe involves the transformation of data contained in the databases to another representation which is required for the algorithms. The problem of protecting process instance data such as customer information contained in the runtime database is related to the problem of protecting the process model in the buildtime database.

As before we propose a solution based on encryption reusing the unique private key of the engine. The general problem using encryption in databases is the possible restriction of expressiveness of client queries (e.g. relational operators such as JOINS), because encrypted values aren't allowed to be decrypted at the DBMS side. The selection of adequate encryption mechanisms which match the required query expressiveness in conjunction with the given database schema is crucial in terms of performance (and security). Customers must investigate which parts of their business processes are worth protecting and rather secure the relevant parts than the whole process.

At this point we would like to refer to existing research on database security [2, 21] or database as a service (DaaS) [1, 9, 10]. These research areas for example investigate how SQL queries over encrypted data can be realized using techniques such as deterministic encryption to support join queries or partitioning (bucketizing) [10, 11] techniques to support range queries. Employing dummy-noise such as falsification of timestamps, or unsharp queries to increase security, moves the scope of responsibility of traditional DBMS functionality into the client application e.g. the navigator component of a BPEL engine. Internal representation of encrypted nodes (attributes, elements, in particular activities) must be "new" objects in the internal representation of the BPEL process and must be treated differently by the navigator.

By identifying the requirements and challenges to securely execute BPEL in the PaaS delivery model, a fair amount of possible attacks to the enterprise assets can be reduced. However PaaS has more security issues compared to IaaS. Because read operations to the main memory are much faster than read operation to data located at the hard disk, applications including BPEL engines keep current execution data or frequently accessed data in main memory which is frail to mining algorithms. Therefore trust in business process outsourcing can only be established, if provider's platforms are certified to use secure coprocessors [27] in combination with the identified secure BPEL engine.

4.3. SaaS

Using SaaS for outsourcing of all tasks not part of the core business of Paradise Travel leads to the biggest reduction of complexity for Paradise Travel. Another effect is the nearly complete loss of flexibility concerning the business process. The SaaS provider provides Paradise Travel only a few customization options regarding the user interface, data management, etc. This leads to a loss of flexibility, as Paradise Travel cannot change the application in case of changing customer demand.

4.3.1. Providing BPEL through SaaS. In contradiction to IaaS and PaaS, the customer of a SaaS provider does not have to worry about *hardware*, *operating sys-*

[8] http://ode.apache.org/

tem, *platform middleware*, and even the *application* itself (Figure 4). At this point a paradigm shift can be observed. The process no longer represents an asset of the tenant's enterprise and is even not visible at all to the tenant.

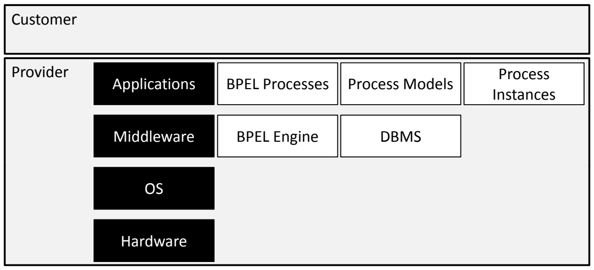

Figure 4. SaaS

By serving an application to multiple customers it has to be distinguished between single-tenant and multi-tenant architectures. When providing BPEL through SaaS, a single-tenant architecture implies the installation of one BPEL engine (and DBMS) not only for each tenant but also for each process model. By relying on a multi-tenant architecture, it is possible to serve multiple tenants with a single BPEL engine (and DBMS) hosting multiple business processes. In both scenarios a tenant request triggers the creation and execution of an instance of the requested business process. Thus the accumulating data of the respective tenants must be protected against unintended access by the SaaS provider as well as other tenants if using a multi-tenant architecture.

While the first security requirement can be easily solved by reusing the proposed modifications to BPEL engines as described in Section 4.2.2, the protection of information against other tenants sharing the same resources needs to be further analyzed.

4.3.2. Requirements and Challenges. Multi-tenant architectures [7] distinguish between three approaches to realize multi-tenancy at data level. These approaches can be further distinguished by means of the degree of data isolation reaching from separate databases to separate or shared schema in a common database. While data isolation is very important it can be observed that it correlates with the required system resources [12] in terms of memory-footprint, used disk space, CPU usage and socket allocation. Utilizing the separate databases approach the data of each tenant is *physical* isolated. The separate schemas approach ensures *logical* isolation of each tenant's data. Both approaches are applicable for using existing database access control mechanisms. Therefore the BPEL engine must be extended to establish a tenant context when a process instance is being created. A tenant context is defined analogous to an authentication context [25], encapsulating the identity of a tenant as well as a reference to the schema location. The engine must be able to map the tenant's context to the appropriate authenticators for database access. The navigator component of the BPEL engine has to be modified to use this context for any database operation.

However, this solution is not directly applicable if the tenants share the same schema. As the same tables are used for the data of multiple tenants, there is neither a logical nor physical separation. Assumed that a BPEL process includes a weak correlation mechanism it is possible that requests to process instances sharing the same correlation values get mixed up and responses containing confidential data are exposed to unintended tenants. A straight forward solution would be to add a column containing a tenant identifier [7] to each table. In the ideal case a DBMS is able to define access control at the level of rows by means of tenant identifiers. Because this kind of functionality would restrict the set of suitable DBMS too much, the concept of a tenant context as discussed above is needed. Furthermore the navigator component has to be extended to restrict the possible query results to the current tenant.

Another solution to the problem of serving multiple customers is to deviate from the multi-tenancy approach and dynamically redeploy a business process for every tenant. Thus a process instance and its corresponding process model are bound to one single tenant, still sharing the same resources at the DBMS level. Then there is no longer a risk to expose data to other tenants by mistake, because now each process has its own endpoint which serves as a unique identifier to distinguish the process instance information.

5. Analysis and First Evaluations with Existing Engines

Given the detailed analysis above of the different requirements for BPEL engines regarding the different delivery models we give an overview of the findings and a deeper analysis of what is required from future engines to support all kinds of delivery model. Today, open source and commercial BPEL engines such as Apache ODE, Active BPEL[9], IBM WebSphere Process Server or Oracle's BPEL engine and others are not explicitly developed with multi-tenancy in mind. They are more geared towards the on-premise market. We have made first experiments in combining the Apache ODE and ActiveBPEL engines and IaaS. Therefore we installed those engines on Amazon EC2[10]. We bundled respective Amazon EC2 images that allow us to quickly start up servers

[9] http://www.activebpel.org

[10] http://aws.amazon.com/ec2/

including the installed BPEL engines for testing and e-learning purposes. However, this approach requires that the user is familiar with the underlying IaaS platform (in this case Amazon EC2). Moreover in case several tenants require BPEL engines simultaneously, multiple servers must be started which means that the overhead of multiple operating systems and other middleware is still present.

Another approach to handle multiple tenants is to install multiple BPEL engines on one server. The advantage of this approach is that the operating system is only needed once. The disadvantage is that still multiple application servers are needed. Additionally the fact that multiple tenants can access a server requires that the engines are specifically secured so that one tenant cannot access the engine of another tenant. The IBM WebSphere Process Server includes the concept of so-called *profiles* that can be used to run multiple process servers on one machine. Access control can be configured on the level of profiles allowing to assign a particular profile to a particular tenant. However, running multiple engines on one server still has the disadvantage that a lot of overhead is caused by only running the engine and underlying application servers. The Apache ODE engine will allow to run multiple instances of the engine on one application server in future releases to minimize this burden.

The commercial BPEL engines such as IBM's Process Server allow access control on a per-model level. For example different administrators (that can stop, resume and modify process models) can be assigned to different process models. In general this makes these engines ready for a PaaS scenario. However, they lack sophisticated features such as the storage of different process models in different database tables to ensure isolation of data. However, in case this is not required and shared databases are of no concern such engines can be used in a PaaS scenario.

In SaaS scenarios access control and isolation on a process instance level is needed. This is not supported by current engines. For example it is not possible to assign different administrators to different instances depending under which tenant context they have been started. We are currently extending the Apache ODE engine with an access control concept that allows to define access control for different tenants on a per-instance level. Furthermore we incorporate techniques from the database field such as encryption to ensure that tenant-specific data can only be read by the correct tenant.

In conclusion, the requirements for a BPEL engine regarding multi-tenancy support rise with complexity of the delivery model. While in the on-premise and IaaS model the mechanisms of the underlying operating system can be used, the PaaS and SaaS model require multi-tenancy on the process model and even process instance level.

6. Conclusion and Future Work

In this paper we investigated the execution of BPEL processes in different delivery models, from IaaS to SaaS. We described the characteristics of the different models and showed their requirements on the underlying middleware. We showed how in particular security and trust issues affect the execution of BPEL processes in different delivery models. Whereas BPEL processes in the IaaS model can be fairly easily executed given today's BPEL engines we showed that other delivery models (in particular PaaS and SaaS) require isolation on the process model and process instance level. Additionally we used several available BPEL engines to provide BPEL in an IaaS model and investigated whether the advanced authentication mechanisms of commercial-grade BPEL engines today are sufficient to provide BPEL engines as platform as a service. As all required features are not available in today's BPEL engines, we described the modifications that are needed to be able to execute BPEL processes in the cloud in the future. We started to modify an open-source BPEL engine (Apache ODE) to include these features. In future work we will also investigate how delivering BPEL in the different delivery models affects the performance of BPEL execution as well as the treatment of data in a BPEL process. Other future work includes aspects such as the user interfaces for both managing the processes as well as for the participation of humans in processes.

Acknowledgments

The work published in this article has partially received funding from the European Community's 7th Framework Programme Information Society Technologies Objective under the MASTER project[11] contract no. FP7-216917 and the COMPAS project[12] contract no. FP7-215175.

References

[1] Providing database as a service. In *ICDE '02: Proceedings of the 18th International Conference on Data Engineering*, page 29, Washington, DC, USA, 2002. IEEE Computer Society.

[11] http://www.master-fp7.eu

[12] http://www.compas-ict.eu

[2] Database security-concepts, approaches, and challenges. *IEEE Trans. Dependable Secur. Comput.*, 2(1):2–19, 2005. Fellow-Bertino,, Elisa and Fellow-Sandhu,, Ravi.

[3] R. Agrawal, D. Gunopulos, and F. Leymann. Mining process models from workflow logs. In *In Sixth International Conference on Extending Database Technology*, pages 469–483, 1998.

[4] F. Aymerich, G. Fenu, and S. Surcis. An approach to a cloud computing network. *In: Proceedings of the First International Conference on the Applications of Digital Information and Web Technologies (ICADIWT 2008)*, pages 113–118, Aug. 2008.

[5] R. Buyya, C. S. Yeo, and S. Venugopal. Market-oriented cloud computing: Vision, hype, and reality for delivering it services as computing utilities. *Proceedings of HPCC '08.*, 2008.

[6] F. Chong and G. Carraro. *Building Distributed Applications Architecture Strategies for Catching the Long Tail*. MSDN architecture center, http://msdn2.microsoft.com/en-us/library/aa479069.aspx, 2006.

[7] F. Chong, G. Carraro, and R. Wolter. *Multi-Tenant Data Architecture*. MSDN architecture center, http://msdn.microsoft.com/en-us/library/aa479086.aspx, 2006.

[8] C. Guo, W. Sun, Y. Huang, Z. Wang, B. Gao, and B. IBM. A framework for native multi-tenancy application development and management. In *In Proceedings of CEC/EEE 2007*, 2007.

[9] H. Hacigms, S. Mehrotra, and B. Iyer. Providing database as a service. *Data Engineering, International Conference on*, 0:0029, 2002.

[10] H. Hacigümüş, B. Iyer, C. Li, and S. Mehrotra. Executing sql over encrypted data in the database-service-provider model. In *SIGMOD '02: Proceedings of the 2002 ACM SIGMOD international conference on Management of data*, pages 216–227, New York, NY, USA, 2002. ACM.

[11] B. Hore, S. Mehrotra, and G. Tsudik. A privacy-preserving index for range queries. In *VLDB '04: Proceedings of the Thirtieth international conference on Very large data bases*, pages 720–731. VLDB Endowment, 2004.

[12] D. Jacobs and S. Aulbach. Ruminations on multi-tenant databases. In *BTW*, pages 514–521, 2007.

[13] A. Keller and R. Badonnel. Automating the Provisioning of Application Services with the BPEL4WS Workflow Language. In *In Proceedings of DSOM 2004*. Springer, 2004.

[14] R. Khalaf. From RosettaNet PIPs to BPEL processes: A three level approach for business protocols. *Data & Knowledge Engineering*, 61(1):23–38, 2007.

[15] R. Khalaf and F. Leymann. A Role-based Decomposition of Business Processes using BPEL. In *Proceedings of ICWS'06.*, 2006.

[16] M. Koning, C. Sun, M. Sinnema, and P. Avgeriou. VxBPEL: Supporting variability for Web services in BPEL. *Information and Software Technology*, 51(2):258–269, 2009.

[17] A. Lazovik and H. Ludwig. Managing Process Customizability and Customization: Model, Language and Process. In *In Proceedings of WISE 2007*. Springer, 2007.

[18] F. Leymann and D. Roller. *Production Workflow – Concepts and Techniques*. Prentice Hall PTR, 2000.

[19] H. Ludwig. Web services QoS: external SLAs and internal policies or: how do we deliver what we promise? In *WISE Workshops, 2003.*, 2003.

[20] Z. Ma and F. Leymann. A Lifecycle Model for Using Process Fragment in Business Process Modeling. In *Proceedings of BPDMS 2008 Workshop at CAiSE'08*, Montpellier, June 2008. Online.

[21] U. Maurer. The role of cryptography in database security. In *SIGMOD '04: Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, pages 5–10, New York, NY, USA, 2004. ACM.

[22] R. Mietzner and F. Leymann. Generation of BPEL Customization Processes for SaaS Applications from Variability Descriptors. In *Proceedings of SCC'08*, 2008.

[23] R. Miller. *Serious Cloud Storage Stumble for FlexiScale*. datacenterknowledge.com, http://www.datacenterknowledge.com/archives/2008/08/28/serious-cloud-storage-stumble-for-flexiscale/, 2008.

[24] OASIS. *Web Services Business Process Execution Language Version 2.0 – OASIS Standard*, 2007.

[25] M. Papazoglou. *Web Services: Principles and Technology*. Prentice Hall, 2007.

[26] J. Recker, M. Rosemann, W. van der Aalst, and J. Mendling. On the Syntax of Reference Model Configuration–Transforming the C-EPC into Lawful EPC Models. *Business Process Reference Models*, page 60.

[27] S. W. Smith and D. Safford. Practical server privacy with secure coprocessors. *IBM Syst. J.*, 40(3):683–695, 2001.

[28] W. van der Aalst, T. Weijters, and L. Maruster. Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering*, 16(9):1128–1142, 2004.

[29] W. M. P. van der Aalst, B. F. van Dongen, J. Herbst, L. Maruster, G. Schimm, and A. J. M. M. Weijters. Workflow mining: A survey of issues and approaches. *Data Knowl. Eng.*, 47(2):237–267, 2003.

[30] W3C. *XML Encryption Syntax and Processing – W3C Recommendation*, 2002.

[31] W3C. *XML Signature Syntax and Processing (Second Edition) – W3C Recommendation*, 2008.