Supporting Interoperability and Scalable Customization for Business-Process-as-a-Service

Majid Makki, Emad Heydari Beni, Dimitri Van Landuyt, Bert Lagaisse, Wouter Joosen

September 13, 2017

1 Introduction

Online software service providers, which include application service providers (ASP) and software as a service providers (SaaS), are evolving and extending their services towards a business process outsourcing (BPO) model. BPO comprises the delegation of many steps of a company's business processes to a specialized online software service provider.

Business processes as a service (BPaaS) can be defined as the offering of outsourced software services that encapsulate business processes. These software services are offered online on a cloud platform and typically adopt application-level multi-tenancy.

An Example. Many companies are outsourcing the business process of invoicing their customers to specialized providers (see Figure 1). In the traditional approach, on-premise workflow systems to manage the billing process are using the online web services of a document management provider to create, layout, send and receive invoices to their customers. This billing workflow will have fine-grained, synchronous interactions with this document service to create, send, update and resend the invoices when these are not paid.

In a BPO context, this whole billing process is outsourced to a billing provider. The company consuming the service of the billing provider now only has a long-term, coarse-grained interaction with the billing process: start the billing process of a customer and notify me when the customer has paid. While such a long-term interaction is running for weeks, the company might ask or receive intermediate updates about the billing process, such as *bill sent*, *bill resent*, or *bill paid*.

BPaaS applications and services build upon workflow engines and middleware such as jBPM or Windows Workflow Foundation to define and execute business processes. However, traditional workflow engines were often not suitable to handle the complex requirements and challenges of BPaaS and cloud-based deployments. In our research line on business process as a service, we have investigated and tackled the following key research challenges.

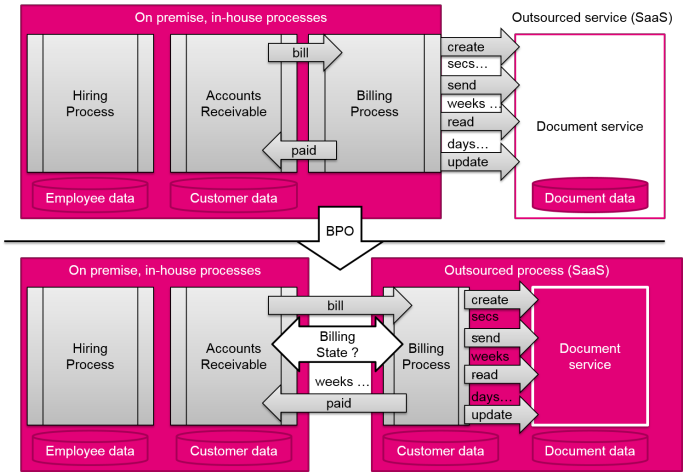


Figure 1: A BPaaS Example: billing as a service

The interoperability challenge. As illustrated in the previous example, business process outsourcing results in decentralized business process flows between companies and involves long term remote interactions between these work flows. Each of these companies typically has its own business process middleware (BPM) to support its automated business processes, such as jBPM[Red], Ruote[MKM+] or WWF[Mil09]. BPO thus results in remote workflow interactions between heterogeneous and federated workflow systems. In a lot of online BPO applications, a similar pattern emerges:

1. There are two federated parties (e.g. two companies) with heterogeneous workflow technologies (e.g. a jBPM workflow engine and a workflow engine for scientific simulations).

2. The first party starts a remote workflow on the second party's workflow system, and awaits its result. This is typically a long term process that can take weeks or months.

3. The first party wants to follow up on the progress now and then. This inspection interaction can be a pull request by the first party or a push notification by the second party. The progress inspection is typically decoupled from the internal task implementation of the remote workflow and expressed in a higher abstraction, e.g. *payment pending* or *simulation 65% completed*.

However, this similar pattern has to be implemented over and over again in each application, and for each type of workflow. Current workflow middleware only coordinates and supports short-term remote interactions such as synchronous or asynchronous calls to web services via SOAP or REST, on top of which the application-specific long-term interaction needs to be implemented.

The customization challenge. Workflows are defined as a composition and coordination of a set of steps and calls to web services (REST or SOAP). In a multi-tenant setting, customers often require customizations of these workflows to adapt to their company-specific data standards, their own hosted services, and additional activities or steps that need to be taken in a business process.

As a coordinated composition of steps and service calls, workflows can thus require customization at three levels:

1. Customization of the behaviour encapsulated in the implementation of one service. For example to adapt a storage service to customer-specific data formats.

2. Customization of the binding to a certain service. For example, to wire the correct mail service of a company to a workflow that is executing.

3. Customization of the control logic and steps in the workflow. For example to add additional audit steps or verification steps in a billing process.

Much research has been done on the customization of service implementations and service bindings. In the context of this paper we will therefore focus on the customization of the workflow definition itself.

DistriNet contributions In the remainder of this paper we discuss DistriNet's research solutions that have contributed to tackling these challenges for BPaaS.

The interoperability challenge, and the frequently occurring, long-term remote interactions between heterogenous workflow technologies requires (screams for) middleware support, on which workflow applications can leverage to reuse the long-term remote interaction pattern, including intermediate inspection via push or pull.

In Section 2 we describe the WF-interop solution. WF-Interop defines a set of interfaces that enable standardized communication between federated and heterogeneous workflow engines. The WF-Interop API focusses on deployment, activation and progress monitoring of workflows.

In Section 3 we present an in-depth analysis and comparison of different strategies for the run-time customization of workflow definitions, on a tenant-per-tenant basis. We also assess their impact on scalability and on the devops process.

We conclude with a discussion on the added value of these solutions for the SaaS market in Section 4.

2 Tackling the interoperability challenge with WF-Interop

The WF-interop interfaces define both a reflective and adaptive middleware for workflows.

- Depending on the workflow engine, and the actual workflow, the interface of the middleware adapts and publishes the supported operations at that moment. (e.g. the pause operation for a workflow is only exposed if the workflow engine supports it and if a workflow is actually running.)
- Running workflows can be inspected in terms of higher application-specific abstractions, depending on the type of workflow. WF-interop supports both a pull and a push model for state inspection.

Next to these advanced features, WF-interop also defines basic operations for workflow engines, such as:

- Workflow engines can be inspected for which types of workflows a specific engine supports.
- Workflow engines can be asked to create new workflow types (via the deployment interface).
- Workflow engines can be contacted to instantiate new workflow instances (via the activation interface).

The contribution of WF-interop is threefold. First, WF-interop introduces a standardized management interface for workflow engines to enable interoperability. Second, it introduces a reflective and adaptive approach to support discoverability, evolvability and adaptability of engine management interfaces. Third, to increase industry adoption, the WF-interop interfaces are currently supported in a REST-based architecture between workflow engines, but also leverage well-known principles such as *Hypermedia as the Engine of Application State* (HATEOAS) as a novel technique to support adaptive management interfaces in middleware.

2.1 WF-Interop Principles

WF-Interop aims to standardize the communication between federated and heterogeneous workflow engines with an adaptive and reflective approach to enable discoverability and evolvability of the management interfaces.

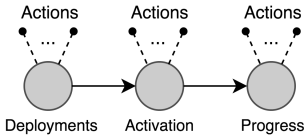


Figure 2: Sequence of resources and their actions.

WF-Interop promises *self-discoverability* of functionalities for consumers without the necessity of having static documentation. It assumes that workflow engines contain resources such as *deployments*, *activations* and progress *monitoring* as shown in figure 2. Various actions can be executed on these resources which may bring the execution flow as well as the internal states of the enactment engine to a different state. Upon execution of each action on a resource, WF-Interop guides consumers by proposing next possible moves in the form of resources and their actions based on the current application state. Consumers are able to read a short on-demand description for each of the proposed actions in order to figure out the protocol, input properties and a brief description of the semantics of the action. They may traverse the application state by invoking these proposed actions on the same or the other resources.

This mechanism introduces API *evolvability* for workflow engine providers in terms of new functionalities on top of WF-Interop. WF-Interop can be adopted as a base interface. Later on, new resources along with their necessary actions can be defined and added to the appropriate application states. The newly added resources are discoverable by the navigational information sent back to the consumer upon each action invocation. Workflow engines come with different capabilities; some may not cover all minimum functionalities defined by WF-Interop. The aforementioned mechanism enables all providers with different levels of support to be *adaptive* to the current state of their production.

2.2 WF-Interop REST Architecture

In this section, we propose a RESTful interface, called *WF-Interop*, to standardize the communication between federated and heterogeneous workflow engines. In addition, we transform our proposed interface to an adaptive and reflective solution by using *Hypermedia as the Engine of Application State*, known as *HATEOAS*.

WF-Interop focuses on four fundamental aspects of workflow engine interactions: (1) deployment of workflow definitions, (2) activation and (3) progress monitoring of process instances within workflow systems including (4) observers.

According to our study based on runtime interfaces of several workflow engines and Workflow Management Coalition (WfMc) standards, the workflow definition deployment and its related functions are the first necessary requisite in our RESTful api proposal. In the first draft of the deployment interface as listed in table 1, one can deploy, undeploy, modify, delete, and fetch workflow definitions.

Table 1: Workflow Deployment Resources

Method	URI	
GET	`/deployments/`	Get all Workflow definitions
POST	`/deployments/`	Deploy a Workflow definition
GET	`/deployments/{id}`	Get a Workflow definition
PUT	`/deployments/{id}`	Update a Workflow definition
DELETE	`/deployments/{id}`	Undeploy a Workflow definition

The second interface provides activation functionalities for workflow process instances. The fundamental methods for these resources are listed in the table 2. It includes functionalities such as instantiation, aborting, pausing and resuming of process instances.

Table 2: Workflow Activation Resources

Method	URI	
GET	`/activations/`	Get all workflow process instances
POST	`/activations/`	Start a workflow process instance
GET	`/activations/{id}`	Get a workflow process instance
DELETE	`/activations/{id}`	Abort a workflow process instance
PUT	`/activations/{id}/pause`	Pause a workflow process instance
PUT	`/activations/{id}/resume`	Resume a workflow process instance

The last interface, specified in table 3, provides progress monitoring for workflow process instances. This interface standardizes the interactions between enactment engines to support progress reporting in a push and a pull model.

Table 3: Workflow Progress Resources

Method	URI	
GET	`/progress/{processInstanceID}`	Get current workflow state
GET	`/progress/{processInstanceID}/observers/`	Get the list of observers
POST	`/progress/{processInstanceID}/observers/`	Subscribe an observer
DELETE	`/progress/{processInstanceID}/observers/{observerID}`	Unsubscribe an observer

WF-Interop, as described in the previous paragraphs, aims at addressing interoperability issues of workflow engines in a RESTful architecture. The rationale behind REST architectures is described in Roy Fielding's dissertation [Fie00] and the fourth layer of the Richardson Maturity model [Fow10]. One of the principles in REST is to utilize *Hypermedia as the Engine of Application State* (HATEOAS). HATEOAS is described as a *constraint* of REST and supports the aforementioned architectural features proposed by WF-Interop such as *self-discoverability*, *evolvability* and *adaptability*.

As illustrated in figure 3, WF-Interop resources are dependent on each other and each resource has a set of actions. Relying on *Hypermedia*[Not10] based principles, one is able to start from one of these resources and explore the remainder of the dependency graph based on the current state

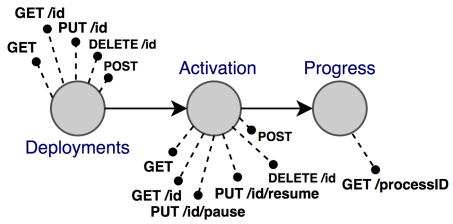


Figure 3: RESTful WF-Interop

of the application. In other words, consumers of the workflow engine are supposed to receive a list of possible actions after execution of an action. These proposed actions come with on-demand documentations enabling the clients to navigate the API's without any prior knowledge about interaction with workflow engines. For instance, one may execute a `GET` method on the `Activation` resource in order to fetch a long running business process. Depending on the current application state of the process and the policies of the enactment engine, WF-Interop may respond back some other actions in Hypermedia format such as links to methods of the progress monitoring resource for more comprehensive reporting or links to obtain intermediate results. All of the suggested links are based on each engine's capabilities which makes the RESTful WF-Interop *adaptive* to all engines.

This architecture is *change tolerant* in the sense that workflow engines are able to either implement new functionalities or even deprecate some actions by manipulation of hypermedia links propositions. For example, some engines can implement some extra methods on top of the Progress resource for additional capabilities and add those actions as links to the response bodies upon proper Activation method calls.

2.3 Implementation, Validation and Further Reading

In [BLJ15] we further described the implementation and architecture of WF-interop in a proof-of-concept middleware and illustrated its use with an accounting workflow that outsources the billing workflow between a jBPM and Ruote workflow engine. Both engines remained unchanged while all WF-Interop middleware support could be modularised in *reusable* software assets such as application independent sub-processes and workflow activities.

In [HBLZ+17] we extended the WF-interop standard and middleware to support advanced authorised delegation of workflow tasks. State-of-the-art workflow engines fall short of a distributed authorisation mechanism for the heterogeneous, federated BPO setting. In a cross-organisational context, the security requirements involve (i) delegation and verification of privileges in a confidential manner, (ii) secure asynchronous operations during the long-term workflows even when the users are logged-off, and (iii) controlling access to interfaces of the different workflow engines involved.

To address these challenges, we present a voucher-based authorisation architecture and middleware. We extended the WF-Interop middleware with a security module to support this authorisation architecture. We further validated our contributions by prototyping a billing workflow case study on top of the extended WF-Interop middleware and evaluated the performance overhead of the security extensions to the middleware.

3 Customization Challenge

This section deals with the challenge of customizing workflow definitions in a multi-tenant context. Section 3.1 provides an overview of all workflow customization strategies which are applicable in a multi-tenant BPaaS. Section 3.2 discusses the impacts of these strategies on the scalability of a multi-tenant system. Section 3.3 briefly elaborates on how DevOps activities are affected by the choice of a workflow customization strategy.

3.1 Overview of Strategies

There are essentially six strategies for customizing workflow definitions in a multi-tenant context [VDA11]. These strategies are summarized in Table 4. As indicated in the table, some

strategies require deploying the workflow definition for each tenant separately while some others allow global deployment of a single workflow definition for all tenants. In addition, some strategies share the required effort for designing workflows between the BPaaS provider and tenant administrators while some others do not require tenant administrators to participate in the workflow design.

Strategy	**Deployment Mode**	**Work Distribution**
Change	Per-Tenant	BPaaS Provider Tenant Admins
Decision Points	Global	BPaaS Provider
Deviation Rules	Global	BPaaS Provider
Underspecification		
Static Late Modeling (**SLM**)	Per-Tenant	BPaaS Provider Tenant Admins
Static Late Binding (**SLB**)	Per-Tenant	BPaaS Provider
Dynamic Late Binding (**DLB**)	Global	BPaaS Provider

Table 4: Summary of Workflow Customization Strategies

The `Change` strategy [VDA11] requires the BPaaS development team to provide a base workflow definition as well as a set of constraints for changing it. Then tenant administrators may *change* the base workflow definition, insofar as the constraints permit, and deploy a specific workflow definition for their own organization in the workflow engine which is shared between multiple tenants.

The `Decision Point` strategy [GWVLJ13], on the contrary, does not require any effort from tenant administrators in designing the workflow definition. The BPaaS development team, instead, embeds a number of decision points in the workflow definition which diverts the control flow based on tenant preferences at runtime. Since applying tenant preferences takes place at runtime, there is only a single instance of the workflow definition deployed in the workflow engine. For instance, Figure 4 shows a BPMN 2.0 workflow fragment of a document processing application which distributes generated documents either via email or by regular post. The decision point marked by a cross branches the workflow based on tenant preferences about distribution type and the branches merge back into the main flow after the document distribution phase is done.

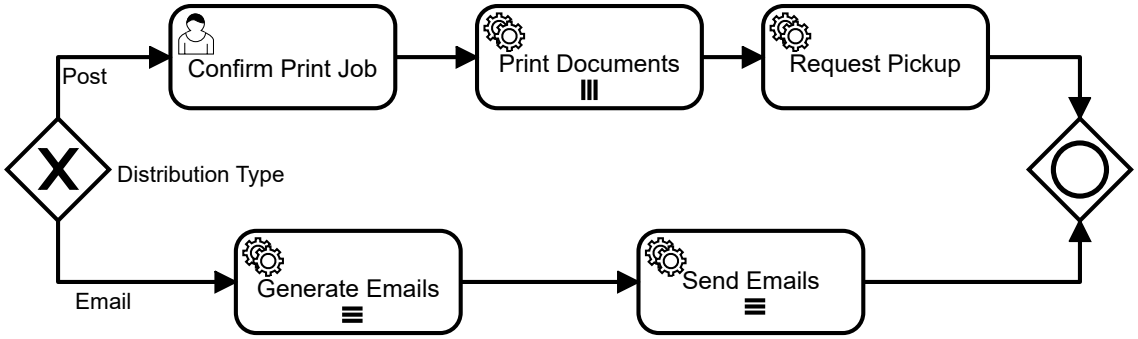


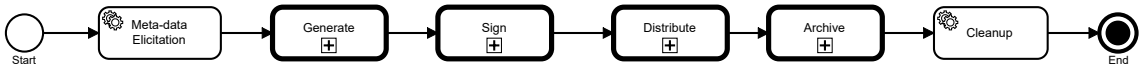
Figure 4: Decision Points for Choosing a Document Distribution Type - BPMN 2.0 Fragment

Applying tenant preferences in case of the `Deviation Rule` strategy, instead, is done from outside of the workflow definition by means of a set of rules expressing tenant preferences and enforcing deviation from the main flow at certain points. For instance, in the context of the above example, the five activities shown in rectangles will be placed sequentially one after the other but the skip rule shown in Listing 1 guarantees that documents are distributed via post which is the preference of a specific tenant. Similar to the case of the `Decision Points` strategy, this strategy requires only a single deployment of the workflow definition for all tenants and does not require any effort from tenant administrators in designing the workflow.

```
1 skip-activities=Generate Emails, Send Emails
```

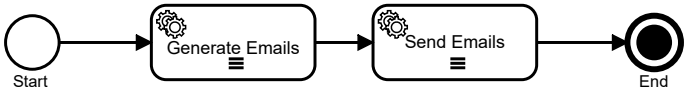
Listing 1: Partial View of Tenant Preferences for the Deviation Rules Strategy

The remaining strategies are grouped under the umbrella term "underspecification" because their master workflow is "underspecified", i.e. has missing parts. For instance, Figure 5a is an "underspecified" document processing workflow with 4 missing parts or placeholders indicated by a + sign. In case of the `Static Late Modeling (SLM)` strategy, these placeholders are filled by workflow definitions designed by tenant administrators at customization time and the updated master workflow will be deployed for each tenant separately. In case of the `Static Late Binding (SLB)` strategy [ML08], existing sub-workflows are automatically bound to the master workflow at customization time based and the automatically generated workflow will be deployed for each tenant separately. For instance, one of the workflows depicted in Figure 5b and Figure 5c) will be bound to the `Distribute` placeholder of the master workflow based on tenant preferences. In this case, tenant administrators do not design any workflow. In case of the `Dynamic Late Binding (DLB)` strategy [MVLWJ16], binding existing sub-workflows (such as those shown in Figure 5b and Figure 5c) to the master workflow takes place dynamically at runtime based on tenant preferences. This implies that each workflow is only once deployed globally for all tenants. Similarly, all workflow definitions are designed by the BPaaS development team.



(a) Master Workflow

(b) Distribution by Post



(c) Distribution by Email (Standard)

Figure 5: Underspecified Master Workflow along with Distribution Sub-workflows

3.2 Scalability Impacts

The choice of workflow customization strategy has an impact on scalability of a multi-tenant BPaaS. Scalability of a multi-tenant offering is directly linked to the question of how resource consumption increases when the number of tenants increases. As mentioned above and visible from Table 4, some workflow customization strategies require deploying a workflow definition separately for each tenant while a single global workflow deployment is sufficient when adopting other strategies. This has a direct impact on the memory usage when the number of tenants increases.

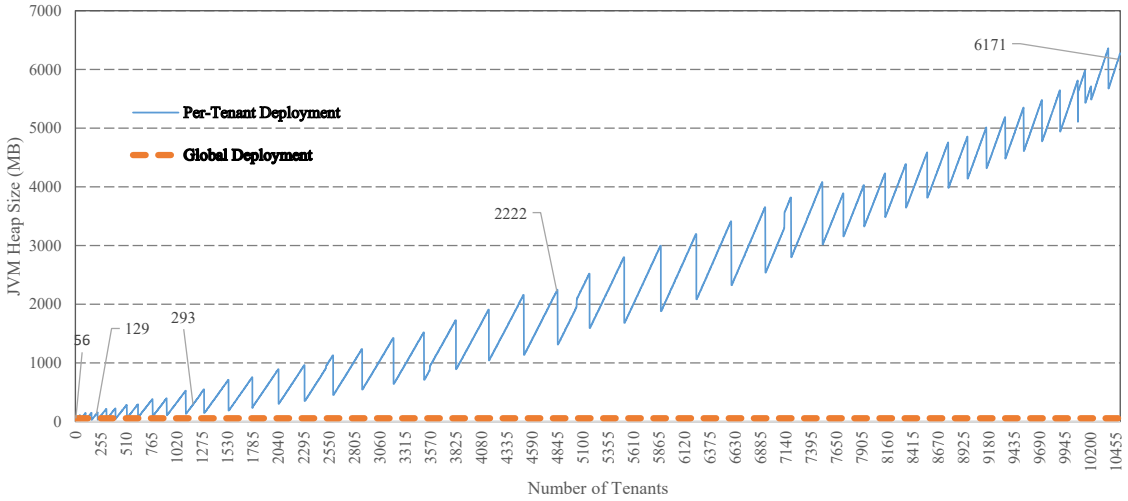


Figure 6: Impacts of Different Workflow Customization Strategies on Memory Usage

Figure 6 roughly compares the memory footprint of the groups of strategies. For the per-tenant deployment case, a single average document processing workflow definition is deployed repeatedly and the changes in JVM heap size are recorded as the memory footprint of this group of strategies. For the other group, a single document processing workflow is deployed and the change in JVM heap size is recorded (cf. [MVLWJ16]). Memory usage in the latter case is 30.5 megabytes, and as shown in Figure 6, does not change with increasing number of tenants while the memory used in the former increases drastically increases when taking an increasing number of tenants on board.[1] The memory footprint of the former group can increase even more when there is more than one workflow definition and more complex ones in a single application.

The huge memory footprint of per-tenant workflow deployment, which is different from memory usage of serving requests, indicates that adopting them requires grouping tenants and associating each group to different (set of) server nodes because these strategies put a considerably restrictive upper bound on the number of tenants that can be served using a single server node. This way, the precision of scaling, i.e. how precisely resources are allocated to tenants, will be reduced specially when the load on an application decreases significantly or when there is a large number of tenants with a very low service load on average. In other words, due to high memory overhead of per-tenant workflow deployment, it will be required to allocate multiple server nodes despite the low service load. As a consequence, BPaaS providers will lose the chance of saving on infrastructure cost if they implement one of these strategies.

3.3 DevOps Impacts

In addition to scalability, DevOPs activities are also affected by the choice of workflow customization strategy. These impacts, which are discussed below, comprise four principal groups namely *development*, *testing*, *release* and *audit & monitoring*.

Development. The `Decision Point` and `Deviation Rules` strategies yield the most difficult workflow definitions to maintain over time as they confuse two different concerns namely implementation of the business logic and handling of tenant preferences [GWVLJ13, MVLWJ16]. The `Static Late Binding (SLB)` and the `Dynamic Late Binding (DLB)` strategies require more effort from the BPaaS development team compared to the `Change` and `Static Late Modeling (SLM)` especially the latter which postpones a great deal of the work to the customization time and delegates it to tenant administrators (cf. Table 4).

Testing. It is easier to test new features for the BPaaS providers when adopting the `SLB` and `DLB` because these two strategies break down the workflow into modules which can be independently tested. On the contrary, the `Change` and the `SLM` strategies incurs the most burden vis-à-vis testing. This is because they require tenant administrators participate in workflow design and consequently necessitate online testing tools (cf. [VLWJ15]) such that tenant administrators can test workflows in the customization dashboard of the BPaaS offering (e.g. [MVLJ16b]). This may require gathering data for automating the testing process which itself incurs some overhead [MVLJ16a].

Release. In a multi-tenant context, new features should be released seamlessly, i.e. without interrupting the service. This is because uninterested tenants should not be disturbed for releasing a new feature which is of interest only for a subset of tenants. For the same modularity reason mentioned above, releasing new features as sub-workflows are the most straightforward release tasks when adopting the `SLB` and the `DLB` strategies. Releasing new features is not a major issue when adopting the `Change` and the `SLM` strategies because the updated workflow definition is not deployed globally. In other words, other tenants will not be disturbed. However, in case of adopting the `Decision Points` or `Deviation Rules` strategies, seamless release is not possible unless two different versions of a workflow definition coexist in the workflow engine. Such a version coexistence requirement is not widely supported by workflow engines. For instance, in the realm of BPMN 2.0 engines, `Activiti` [Act] is the only widely used engine which supports version coexistence. For the engines which do not support this, some workarounds will be required.

[1] The bounces in Figure 6 are the effect of the Java garbage collector (GC) which releases memory of unused objects. But even Java GC cannot stop the increasing memory usage.

Audit & Monitoring. By adopting workflow customization strategies which deploy workflow definitions on a per-tenant basis, the audit & monitoring tools of existing workflow engines can be used without any problem [ML08]. This is the case for the `Change`, the `SLM` and `SLB` strategies. However, an additional middleware is required on top of existing workflow audit & monitoring tools when adopting the other three strategies. This middleware is responsible to give a single-tenant view of a multi-tenant global workflow definition in case of monitoring and to filter out other tenants data in the audit process.

4 Added value and Conclusion

SaaS-oriented software companies can add value to their offering by growing into the space of business process outsourcing. Business process outsourcing refers to the systematic and controlled delegation of many of the steps of a company's business process, typically a process that is enabled by ICT means. In fact the service provider will thus administer and manage the business process steps according to a service level agreement. In summary, BPaaS is an important extension to SaaS, as it allows the provider to add more value in the online application services; and as it enables the outsourcer to obtain more cost efficiency.

However this cost efficiency should not reduce the customer-intimacy between provider and consumer. Customer-specific customizations and extensions are crucial for the Flemish software industry. Another challenge is that business process outsourcing results in decentralized, federated business process flows that cross the borders of companies.

In this overview paper, we presented solutions to enable service providers in the cloud to increase added value in their offerings by supporting complex business processes and workflows in an interoperable and adaptable way. Specifically, we focussed on 1) solutions for secure interoperable collaboration between such business processes in a federated, inter-organizational setting, and 2) cloud-enabled scalable execution for multi-tenant workflows with customer-specific customizations.

The industry demand for these research solutions came from both service providers and technology providers. By engaging in BPaaS, the SaaS providers will add value to their business. Moreover, building up of the necessary capabilities will be a key success factor in maintaining and improving market leadership. The technical outcome of DistriNet's research contributions resulted in robust customization and secure outsourcing of whole business processes, and thus the strengthening of existing SaaS offerings towards better services and competitive advantages.

References

[Act] Activiti User Guide. https://www.activiti.org/userguide/. Accessed: 2017-05-24.

[BLJ15] Emad Heydari Beni, Bert Lagaisse, and Wouter Joosen. Wf-interop: Adaptive and reflective rest interfaces for interoperability between workflow engines. In *Proceedings of the 14th International Workshop on Adaptive and Reflective Middleware*, ARM 2015, pages 1:1–1:6, New York, NY, USA, 2015. ACM.

[Fie00] Roy Thomas Fielding. *Architectural Styles and the Design of Network-based Software Architectures*. PhD thesis, 2000. AAI9980887.

[Fow10] Martin Fowler. Richardson maturity model: Steps toward the glory of rest. 2010.

[GWVLJ13] Fatih Gey, Stefan Walraven, Dimitri Van Landuyt, and Wouter Joosen. Building a customizable business-process-as-a-service application with current state-of-practice. In *International Conference on Software Composition*, pages 113–127. Springer, 2013.

[HBLZ+17] Emad Heydari Beni, Bert Lagaisse, Ren Zhang, Danny De Cock, and Wouter Joosen. A voucher-based security middleware for secure business process outsourcing. In *9th International Symposium, ESSoS 2017*, volume 10379, pages 19–35. Springer, Cham, June 2017.

[Mil09] Matt Milner. A developer's introduction to windows workflow foundation (wf) in. net 4. *Retrieved from: on Oct*, 11(2010):47, 2009.

[MKM+] J Mettraux, K Kalmer, R Meyers, HC de Mik, A Kohlbecker, M Barnaba, G Neskovic, N Stults, O Pudeyev, M Gfeller, et al. Ruote-a ruby workflow engine.

[ML08] Ralph Mietzner and Frank Leymann. Generation of bpel customization processes for saas applications from variability descriptors. In *Services Computing, 2008. SCC'08. IEEE International Conference on*, volume 2, pages 359–366. IEEE, 2008.

[MVLJ16a] Majid Makki, Dimitri Van Landuyt, and Wouter Joosen. Automated regression testing of bpmn 2.0 processes: a capture and replay framework for continuous delivery. In *Proceedings of the 2016 ACM SIGPLAN International Conference on Generative Programming: Concepts and Experiences*, pages 178–189. ACM, 2016.

[MVLJ16b] Majid Makki, Dimitri Van Landuyt, and Wouter Joosen. Automated workflow regression testing for multi-tenant saas: integrated support in self-service configuration dashboard. In *Proceedings of the 7th International Workshop on Automating Test Case Design, Selection, and Evaluation*, pages 70–73. ACM, 2016.

[MVLWJ16] Majid Makki, Dimitri Van Landuyt, Stefan Walraven, and Wouter Joosen. Scalable and manageable customization of workflows in multi-tenant saas offerings. In *Proceedings of the 31st annual acm symposium on applied computing*, pages 432–439. ACM, 2016.

[Not10] M Nottingham. Rfc5988: Web linking. *Internet Engineering Task Force (IETF) Request for Comments*, 2010.

[Red] RedHat. jbpm business process management suite.

[VDA11] Wil MP Van Der Aalst. Business process configuration in the cloud: How to support and analyze multi-tenant processes? In *Web Services (ECOWS), 2011 Ninth IEEE European Conference on*, pages 3–10. IEEE, 2011.

[VLWJ15] Dimitri Van Landuyt, Stefan Walraven, and Wouter Joosen. Variability middleware for multi-tenant saas applications. In *Proceedings of the 19th International Software Product Line Conference*, pages 211–215, 2015.