Towards simplifying and automating business process lifecycle management in hybrid clouds

Hua Liu, Yasmine Charif, Gueyoung Jung, Andres Quiroz, Frank Goetz, Naveen Sharma

Xerox Research Center at Webster
800 Phillips Road
Webster NY 14580


Abstract—Business Process Management (BPM) software provides visibility into business processes in organizations of all sizes and helps increase process efficiency continuously. However, the time and effort involved in modeling, deploying and executing a business process is tremendous and as a result, organizations struggle to agilely adapt business processes to dynamic business requirements. On the one hand, the growing popularity of cloud computing poses opportunities and challenges on how business processes can leverage resource outsourcing and elasticity. In light of the above, this paper presents a business process management platform that assists business analysts lacking necessary programming expertise by automating manual steps and providing guidance and recommendations to quickly and efficiently design, implement, deploy and execute business processes in a hybrid cloud environment.

I. INTRODUCTION

The high dynamics of markets demand that organizations be able to cope with constant business changes by quickly adapting their business processes [1]. Business Process Management (BPM) software aims at providing visibility into business processes in organizations of all sizes and increasing process efficiency continuously. However, the time and effort spent on modeling, implementing, deploying and executing business process is tremendous, and as a result, organizations struggle to stay agile in a cost effective way. The typical business process lifecycle includes modeling, execution and measurement [2]. When progressing from one phase to the next, manual work is often required and in some cases is very extensive. Business analysts model business processes at a very high level, usually only specifying their control flow. There are gaps and challenges in going from description of business processes to their implementation as software applications [5].

Meanwhile, with the growing adoption of clouds, today’s business processes are increasingly being deployed to clouds to leverage the resource outsourcing and elasticity they provide in order to accommodate dynamic workload in real-time without pre-investment in hardware and software. This introduces another level of complexity into business process instantiation and deployment. The same business process model, when required with different QoS, needs to be re-programmed to use different concrete services and/or have different deployment plans, a transformation that can take days or even months. In today’s fast changing environment, organizations need to stay agile to accommodate changes in an efficient manner and therefore keep looking for approaches to automate business process instantiation and deployment.

This paper presents a business process management platform that addresses the need for agility in business process design by providing guidance and recommendations to business analysts without requiring from them any minimum programming expertise. The platform automates manual steps by generating executable code and taking care of issues of distributed deployment across hybrid cloud environments. The rest of the paper is organized as follows. Section II describes the related work. Section III introduces our platform and the prototype implementation. Section IV concludes the paper.

II. RELATED WORK

Many research efforts have been devoted to BPM but they provide isolated solutions to support different lifecycle phases of business processes. However, the inter-dependence between lifecycle phases requires all the mechanisms to be integrated in an organic manner in order to create consistent and optimized BPM support. Integrated tools such as jBPM [8], Activiti [9] for defining, deploying and executing business processes exist but they require a significant amount of manual programming. Tools for human workflows enable programming-free development but they have limited usage since many business processes are hybrid with both human tasks and service tasks.

In the business process design phase, many research efforts have focused on managing the complexity of business process models [3], motivating users to actively participate in business process creation [4], and aligning business processes with business goals [5]. Due to the narrow focus on business process design, these solutions alone cannot enable agile end-to-end management. Integration with solutions for other phases of a business process lifecycle is problematic.

Many of the issues that are addressed in this paper for business process workflows have been similarly addressed in the scientific workflow domain. For example, [11] proposes a similar separation between abstract and concrete service interfaces (referring to them as logical and physical workflows), presenting formal mathematical abstractions for workflow compositions in terms of data flow. Our work essentially incorporates a data flow into the control flow oriented business process design and additionally deals with practical details such as user assistance for the specification of input/output
mappings and data type compatibility. Another issue that is dealt with in this paper, referred to as shimming in the scientific workflow literature, has been addressed for example in [12]. The model described in the cited paper, used for describing mappings of inputs/outputs of abstract and concrete services, service compositions, and data transformations, is similar to that described herein. In fact, the language framework described there could have been adopted in our implementation. However, we are also concerned in guiding users so that they can specify these mappings given multiple mapping options and so that the code necessary to realize them can be automatically generated.

Implementing business process models into software applications has received much attention, resulting, for example, in the use of technologies such as WSBPEL [6]. Due to the complexity and limitations of WSBPEL, the recent trend has been toward the direct execution of business processes modeled in Business Process Modeling Notation (BPMN) [7] and XML Process Definition Language (XPDL) [10] on engines such as jBPM [8], Activiti [9] and XPDL [10]. These approaches typically require modeling business process in high level modeling languages and supplementing the models with software programs to make them executable. As a result, a significant amount of manual coding is required. One of the contributions of this paper is to automate the instantiation of business processes from high level models to executable application software.

Non-functional features such as cost, performance, availability, and data privacy have been recently researched to improve the performance of business process deployment and execution. [20] and [21] proposed techniques to shorten the length of a business process by reducing service component redundancy, increasing the parallel execution of service components, and pruning the search space. [18] and [19] optimized the end-to-end execution time (i.e., make-span) of business processes by smartly choosing service end-points (i.e., service providers) or computation end-points (i.e., clouds). However, their approaches did not consider cost optimization when deploying business processes. [16] and [17] integrated a cost function when scheduling business processes in cloud and grid environments. However, they did not address the tradeoff between the end-to-end execution time and the sum of service costs in a complex business processes that has a number of parallel sub-flows. By considering a critical path that represents the slow path in parallel sub-flows, our deployment planner can precisely address the tradeoff while deploying business processes in large distributed compute clouds.

III. Platform Overview

The platform has been prototyped and installed on Rackspace instances [22]. In this work, we define a solution template as a business process model and a solution as a solution template instantiation.

After log-in, a user can search for templates based on keywords, category, provider, creation date range and/or template identification. The user can select and modify an existing template or create a template from scratch in the web-based editor via dragging-and-dropping new services, deleting or modifying existing services in the middle canvas, shown in figure 1. After modification, the user completes the data flow of the template and verifies it by walking through every step and mapping inputs to outputs based on recommendations in a wizard interface. The wizard (described in Section III-A) assists non-programmers in designing business processes via making smart suggestions on data flow bindings.

The user can choose to save this template for future use or go ahead to instantiate this template into a solution and deploy it to the cloud environment, as shown in figure 2. The user can choose to further constraint steps to use private services, and configure cost and time parameters. The platform will then provide different instantiation and deployment plans with estimated overall cost and time (the technical details are covered in section III-C). The user then selects one solution from the list and deploys it to the cloud by clicking the “Deploy” button. The selected solution is decomposed into fragments and submitted to engines for decentralized execution. How a solution is decomposed is described in section III-D.

Up to this point, the user has successfully created, instantiated and deployed a solution to the cloud without writing a single line of code. For users who want to further customize the generated solution and deployment plan, APIs are provided for them to overwrite the automatically generated code.

A user can view and operate on the tasks assigned to him/her...
defined in solutions by going to the “View My Tasks” tab, shown in figure 3. Via clicking “Pass”, the user can pass the task to another user or group, which is the recommended by the platform based on the required skills for this task and workers’ skill sets. The section III-B provides the details of how to recommend workers for human tasks. The user can work on the task via clicking “Take”. After the task is complete, the task status becomes “Finished”. The engine will be notified and then start executing the steps following this human task in the solution.

A. Assisted business process design

The proposed platform assists non-programmer users in designing business processes. Users submit high-level descriptions of business processes in the form of sketches illustrating the orchestration of services (e.g., Figure 4). These business processes only contain abstract services and control flows but lack descriptions of data flows and data objects that are passed between services. The platform then helps specify data flows between services, but also detects missing edges and services, for which it provides suggestions for users to validate. The suggestions are computed and ranked using heuristics, and displayed through a wizard interface to the user. The output of the wizard is a completed and corrected business process described in BPMN 2.0 standard [7] (e.g., Figure 5), readily instantiable to be executed by existing business process engines (e.g., jBPM [8]).

1) Preconditions: We assume that the platform is provided with a service registration mechanism and a data type hierarchy allowing the classification of services into service types and their instances. The purpose of service registration is to enable service interoperability and set up a common agreement on service semantics and usage. It is performed through an interface where service providers store and map their (concrete) services to abstract service types in a repository, and map their operation parameters to abstract data types. Figure 6 illustrates this concept. The registration process enables business process reusability by allowing the manipulation of abstract service types instead of service instances in the business process design.

The data type hierarchy supports input/output mapping in the service registration as well as service binding by assessing the compatibility between data types. For this purpose, the platform makes use of a predefined hierarchy of data types. Two data types are said to be compatible if they are on the same path within the hierarchy.

2) Wizard for completing business process data flow: The proposed platform assists users in specifying the data flow of the designed business process by means of a wizard. The wizard displays, for each service, suggestions for binding the service input parameters as illustrated in Figure 7. In order to go beyond data type information (e.g., as is the case in code completion features in IDEs such as NetBeans), those suggestions are ranked by means of a series of heuristics based on data type compatibility, data type transformations, and historical data to suggest the most likely option first.

3) Heuristics for ranking data flow binding suggestions: In order to complement data type compatibility, the heuristics are based on two general principles:
The wavefront outputs, e.g., from S1, that are sub-types of the input data type from S2.

The wavefront outputs, e.g., from S1, that are super-types of the input data type from S2, but can be linked to an output of class (C1), e.g., input transformed by S1.

The wavefront outputs, e.g., from S1, that are not compatible (neither sub- nor super-type) with the input data type from S2, but can be linked to an output of class (C1) or (C2), e.g., input to Si. This binding typically requires the insertion of a conversion service, e.g., Sx, to perform the transformation between S1-output and S2-input.

The wavefront outputs, e.g., from S1, that are not compatible (neither sub- nor super-type) with the input data type, but are the LHS of a possible transformation to that type; e.g., using Sx.

All other wavefront outputs.

1) A business process is often designed as an end-to-end transformation of data objects. Thus, identifying the transformations that each service performs on a data object as it moves through the business process will give an indication of the parameters that should be bound together.

2) Users will likely apply the same combinations of services in different business processes so that keeping track of the ways the services are bound in the past will be useful to guide future compositions.

In order to formalize the notion of the transformation of data objects in the context of business processes, we assume that each service can potentially hold a transformation function that associates its main input (the left-hand side, LHS) to its main output (the right-hand side, RHS, e.g., in the ImageEnhancement service the operation enhanceImage transforms the input rawImage to the output enhancedImage). The transformation function applies by default to inputs and outputs which: 1) have compatible data types, 2) have a small semantic distance. The service provider may then override it by linking the inputs and outputs involved in a transformation during service registration.

Applying the above general principles implies keeping a list of: 1) all business process inputs, i.e. external data objects made available by the user at the start of a business process, and 2) intermediate outputs, i.e. for a service, these are outputs from all services preceding or parallel to it in a business process. These data objects (represented by their corresponding data types) constitute the available outputs, and are referred to as a wavefront for that particular service. The wavefront originates from the business process inputs and grows as more outputs become available for each service. Then, for each service input, the available outputs need to be classified and ranked in the order that they are most likely to be connected to it in a data flow, and presented to the user accordingly.

Table I gives the initial classification of wavefront outputs (according to the level of compatibility between an input and an output). This classification produces groups of outputs in the wavefront. Within each class, for an unconnected service S2 input, outputs are ranked higher according to the following heuristics:

(a) If the output belongs to a service directly preceding the unconnected service S2 in a business process.
(b) If the output is produced by a service S1 such that S1 and S2 are bound by the same input/output pair in the glue code repository.
(c) If two outputs in the wavefront, e.g., from S0 and S1, meet the previous condition (b) they will be ranked according to the weight of the corresponding service pairs in the glue code repository, i.e. number of occurrences out of the total number of stored business processes.
(d) If the output is linked to a workflow/user input.
(e) If the output is not already bound to an input.

For classes (C3) and (C4), conversion services will be searched in the service type repository to perform the binding. A Conversion Service Sx has a transformation to convert the output of service S1 to the format required by the input of the service S2 in a business process. The following heuristics are used to rank conversion services:

(f) S1-Sx and Sx-S2 exist in the glue code repository, bound using the respective output/input of S1/S2.
(g) Remaining inputs of service Sx that can be bound with links of classes (C1) and (C2) is maximal.
(h) Remaining inputs of service S2 that can be bound with links of classes (C1) and (C2) with service Sx outputs is maximal.

Using this assisted business process specification wizard, the user will either accept the default suggestions or quickly find a suitable alternative within ranked output lists and conversion service suggestions. If the user is not able to completely specify the data flow with the help of the wizard, the platform will keep track of the suitable options so that the missing bindings can be established by complementary mechanisms, e.g., crowdsourcing and/or human tasks. Therefore, the platform provides both flexibility and as much as assistance as the user requires to complete the business process specification. After specification, the business process is ready to be instantiated as discussed in the next section.

The final step in the process design stage before the process is ready for instantiation is to generate an executable workflow description (i.e. BPMN script). The automatic generation process consists in first creating a global variable for each input/output bound when specifying the data flow. These global variables are then connected to the necessary inputs and outputs of the abstract services. This approach is beneficial because it facilitates breaking up the process in a distributed deployment (as described in Section III-D), using the global variables as pipes to link the different fragments of the workflow. After this, syntactic elements for the specific executable format (e.g. BPMN gates) are added. At this point, the generated script contains abstract services, but these are replaced by concrete service endpoints after the deployment planner described in Section III-C helps the user in selecting these services.
B. Support of Human Tasks

A typical business process contains automated services and human tasks. To instantiate a business process, services need to be identified to instantiate service tasks and workers will be selected to perform human tasks as discussed below. Selection of concrete services is addressed during the process deployment stage and is therefore discussed in the next section. This section describes the selection of workers with relevant skills for manual process tasks assignment.

Selecting the right workers with the right competency in the right position at the right time to perform human tasks is critical in order to achieve a high level of organizational productivity and effectiveness. It has been intensively researched in competency-based management. However, competency-based management is a top-down organizational process that cannot be exploited to support team building operations in informal working activities [14], for example crowdsourcing environments. In many organizations, competency-based management involves a lot of manual work, whereby jobs are assigned to workers by managers who normally do the mapping in spreadsheets, and recruitment and training processes are triggered at strategy planning meetings. This process suffers from the weaknesses of manual processes, such as slow adaptation and inaccuracy. A systematical approach is desired to automate the processes and reflect the changes in workers’ skills and levels in real-time. We provided a semi-automated approach that recommends workers for human tasks in business processes.

Step1: Using clustering techniques, workers are grouped based on their skills. The signature of group $x$ is defined as $sig_x$, which consists of a set of skills $\{s_i\}$. Suppose the group has $n_x$ workers, the number of workers with skill $s_i$ in this group is $n_{x,i}$, the total number of workers is $N$, and the total number of workers with skill $s_i$ is $N_i$. $s_i$ is included in $sig_x$, if $n_{x,i}/n_x > N_i/N$. As new workers join, they will be classified into different groups according to their skills and group signatures. The addition of new skills may trigger re-clustering and therefore result in re-calculation of signatures.

Step2: Skills $\{skill_a\}$ required for a human task (a) are automatically inferred using the mechanism discussed in section III-A and these skills ($\{skill_a\}$) are then verified by the business process designer.

If $\text{skill}_a \subseteq \bigcup sig_x$, meaning the required skill set is the combination of these group signatures, matching workers will be searched from the overlapped set of these groups. If $\text{skill}_a \subseteq \bigcap sig_x$, meaning the required skill set is the subset of these group signatures, matching workers will be searched from these groups.

If a business process contains more than one human task and/or one task needs more than one worker, the social relationships among these workers, such as trust [14], roles and temperaments [15], are further considered to build an efficient and strong team fabric. Techniques such as collaborative filtering and case-based reasoning can therefore be used [13].

In some cases, it is difficult to find an exact match for workers with all the required skills at the time of need. As a result, training existing workers and/or hiring new workers for missing or highly demanded skills is needed. From a cost perspective, an ideal candidate for training purpose should have the most sufficient background knowledge and strong learning capabilities, which can be reflected by the percentage of relevant skills in his/her skill set. Workers possessing more of the relevant skills have higher chances to acquire other missing relevant skills. Therefore, workers with the largest number of relevant skills compared to the missing or highly demanded skills are identified as training candidates. Skill relevance can be calculated using the formula:

If $n_{i,j}/n_i > N_j/N$ and $n_{j,i}/n_j > N_i/N$: \[ \text{Relevance}(s_i, s_j) = 1 \]

Otherwise: \[ \text{Relevance}(s_i, s_j) = 0 \]

$s_i$ is skill $i$, $s_j$ is skill $j$, $n_{i,j}$ and $n_{j,i}$ represent number of workers with both skill $i$ and $j$, $N_i$ is the number of workers with skill $i$, and $N_j$ is the number of workers with skill $j$. Intuitively speaking, skills $s_i$ and $s_j$ are relevant to each other, if the majority of workers with skill $s_i$ also has skill $s_j$ and vice versa; skills $s_1, s_2, ... s_n$ are relevant if every pair of them is relevant.

C. Optimized instantiation and deployment

A deployment planner has been developed and integrated into our platform to generate concrete business processes in the form of executable workflows, and to optimize those concrete business processes in terms of the end-to-end execution time and cost in multiple clouds. Figure 8 outlines the business process deployment principle. At this point, the planner is provided with an abstract business process (given from III-A) and a list of concrete services candidates for each service of the business process. The planner first generates a list of concrete business processes (marked 1 in Figure 8) by: 1) replacing each service of the given abstract business process with one of the corresponding concrete services, 2) considering different combinations of concrete services. It then optimizes the deployment of these concrete business processes by mapping concrete services to available data centers in private and public clouds (marked 2 in Figure 8), in a way that minimizes both the end-to-end execution time and cost. Our platform takes into account user preference to estimate the optimal deployment by allowing the user to provide weight values for the optimization objectives (i.e., time and cost). Based on the user preference, the planner computes a utility value by normalizing time and cost estimates for each concrete business process. Formally, for given cost upperbound, time upperbound, cost weight, and time weight, it computes utility:

$$ u = \frac{w_c}{c_u} + \frac{w_t}{t_u} $$

Here, $w_c$ represents cost weight, $w_t$ time weight, $c_u$ cost upperbound, and $t_u$ time upperbound. Cost $c$ and time $t$ are accumulated while exploring services of business process.

The planner returns concrete business processes and estimates with minimal utility and displays them to the user sorted
Fig. 8. The overview of business process deployment into a hybrid cloud

The optimization of the deployment of concrete business processes faces three specific challenges. The first is the explosion of available cloud infrastructures that offer different pay-as-you-go schemes and resource capacities. This fact makes the selection of appropriate data centers in clouds for running/hosting different services a non-trivial problem. The second is the significant potential tradeoff between cost, task computation time, and data transfer delay between geographically distributed data centers in clouds, which makes finding an optimal balance difficult. The third is that the parallel sub-paths induced by branch steps in a business process add complexity, given the time slack caused by the different velocities in which these sub-paths reach a merging step.

Accounting for these challenges, we cast this optimality problem into a shortest path search problem in a directed acyclic graph. In a sequential sub-path in the graph, a node represents a single candidate data center selected for corresponding concrete service of a given concrete business process. In parallel sub-paths, a node represents a set of parallel data centers, each of which is selected from each sub-path. Nodes have positive values (i.e., time and cost to execute concrete services in selected data centers), and edges have positive weights (i.e., data transfer delay between consecutive data centers). Then, for given source and sink nodes, the goal is to find the shortest path from source to sink while expanding nodes. We adopt the A* graph search algorithm [23] to speed up the search in potentially large graphs. When searching in a sequential sub-path, our algorithm expands a current node, which has a data center for a current service, to the next candidate nodes, each of which has a data center for the next service in the business process. Then, it chooses one of candidate nodes as the next start point in the search. Meanwhile, when searching in parallel sub-paths induced by a branch node, our algorithm visits the next data centers in parallel from a current node. In this case, it expands the current node by generating different nodes, each of which has a different combination of parallel data centers. Moreover, our algorithm considers the temporal difference (i.e., time slack in Figure 9) between parallel steps. Because the algorithm should minimize the time slack, we consider only cost as a factor within the time slack when the algorithm chooses a data center in the faster sub-path. For example, it chooses a slower but cheaper data center to deploy a service of this sub-path. It repeats this parallel search until it reaches a merging step.

Figure 9 shows a concrete business process that consists of 6 services including Start and End. It has two parallel sub-paths with a branch service. The first sub-path has Service1 and Service2 while the second sub-path only has Service3. Our algorithm works as follows for a given concrete business process:

- **Step 1:** determines available data centers for each concrete service.
- **Step 2:** computes heuristic value for each service by choosing the best data center for each service (among candidate data centers) in the context of normalized computation time and cost (note that the data transfer delay is excluded at this step). Then, the heuristic value of a service is a cumulated value from the service to End, called “admissible heuristic estimate” in A* algorithm.
- **Step 3:** in sequential sub-path (e.g., from Start to BranchService in Figure 9), visits each data center and computes utility value using cumulative cost, computation time, data transfer delay, and heuristic value of the service, and then stores the data center with the utility value in a new node. In parallel paths, makes different combinations of data centers, each of which is selected from each path, and then, computes utility value for each combination. Note that the cost is the sum of that of all data centers in the combination, but the longest time among data centers is chosen for the time value. Finally, stores the combination with the utility value in a new node. Records generated nodes into the node list.
The algorithm repeats the search until reaching End. Now the concrete business process is deployed to different data centers in multiple clouds. This multi-cloud deployment, which requires special encoding of the process into an executable language such as BPMN, is the subject of the next section.

D. Decentralized execution

Currently, BPMN engines execute BPMN scripts that accommodate services in heterogeneous and geographically separated data centers by orchestrating the invocation or execution of the services in a hub and spoke manner, marshaling and linking inputs and outputs to and from each service. This is a centralized approach, since a single BPMN engine participates in the orchestration, even though the different services are executed in a distributed manner. However, it may be desirable for multiple BPMN engines to interact in the distributed orchestration for reasons such as privacy and communication efficiency (efficient local interaction with services and efficient direct communication between data centers).

The BPMN 2.0 specification [7] describes collaborating business processes orchestrated by multiple engines, but each of these business processes must be defined manually as separate BPMN scripts. This section describes a method for partitioning a BPMN process definition and a platform architecture for enabling the communication of multiple BPMN engines for their distributed execution and meta-orchestration, so that the orchestrated execution produces the same result as the original script.

We consider an input process to be a complete and correct BPMN process definition describing a single sequence or data flow. The output of the algorithm will consist of one or more process fragments, each of which is associated to a single service group. A group is defined as a context for the execution of one or more business process activities/services, such as an execution engine, data center, or other set of shared resources. In other words, services that are grouped together will be executed in the same context.

We define three constructs for use in the division algorithm. These constructs are sub-diagrams that perform a specific function and, as their name implies, are used to construct the output fragments. An End join will be used as a sink for all paths created for an output process fragment. A Branch indicator will signal other processes that a particular branch in the process graph was taken. Finally, a Conditional start triggers the execution of a fragment depending on an event received, so that the fragment will either start or terminate given the event message.

In addition to the BPMN constructs and the message types defined by the input process, our division algorithm defines specialized message types that will be used by the execution engine for sequence orchestration and data flow. An Activity call message represents an activity that does not belong to the group of the current fragment and signals the execution of that activity in another fragment. An Indicator message represents a choice resulting from an exclusive conditional branch gateway, and signals the execution/termination of fragments that depend on that choice. Finally, a Merge message is like an activity call message, but containing the merge identifier and all outputs of the previous activity or flow.

Two global structures are maintained for the algorithm: a set of search contexts, each of which represents a depth-first search from a different starting point in the input graph, and an initially empty merge table. One search context is initialized for each start event in the input graph, and, for each active search context, a depth-first search is conducted, by default copying nodes from the input graph into an output fragment. Different actions are carried out depending on the category of each node encountered in the search. The main actions are:

- **For exclusive branch gateways**: Store the branch in the current search context and create a new branch indicator in the output fragment for each outgoing path from the branch.
- **For merge or join gateways**: Update the merge table with a reference to the gateway, and create the corresponding gateway in the output fragment, or retrieve the gateway from the table and create a merge message send and receive events from the current output fragment to the stored gateway node.
- **For activity nodes**: Copy the activity to the output fragment and continue depth-first search if within the current search context group, or create an activity call message to a new search context created with a conditional start, taking into account branches in the search context.

Once a depth-first search for each context finishes, the algorithm either starts with a new search context in the context set, or terminates if the context set is empty. For each new search context started, a new output fragment is initialized.

Figures 10 and 11 show an example of input and output diagrams of a process that is divided using mechanism described in this section. Figure 10 is the original BPMN graph (input), with 3 group bindings, and figure 11 is one of the output fragments generated for group B, showing activity call and merge messages, and branch indicator constructs.

Meta-orchestration between execution engines is implied by the messages that will be generated due to the division into fragments. When a message throw event happens, the meta-orchestrator will use a destination group attribute to
pass the included payload to the corresponding group. The receiving group’s orchestrator will introduce the message to the process instance through a message catch event. Note that the advantage of this approach is that the meta-orchestrator does not have to manage BPMN elements or constructs and is thus independent of the BPMN engines used. At the same time, BPMN engines can be reused without modification and wrapped by the meta-orchestrator module to catch and inject messages.

IV. Conclusion

This paper presented a business process management platform that assists business analysts to quickly and efficiently design, implement, deploy and execute business processes in a hybrid cloud environment. Business process design and implementation are enhanced by a data flow specification wizard that provides guidance and recommendations to the user for input/output bindings, for missing edges and services, and that automatically generates an executable (e.g., BPMN script). This assistance alleviates the burden from users to be aware of programming details, such as data type compatibility and necessary glue code and/or services, as well as syntactical notation details. They only need to provide a high-level abstract control flow and follow the steps of the wizard to connect the services and obtain a final executable process.

The platform also facilitates the optimal instantiation of the business processes by selecting concrete service implementations to maximize users’ utility, considering multiple implementations of abstract services and multiple possible deployments in hybrid cloud environments. Final deployment and execution are also automated, which is made possible by dividing the business process description between multiple distributed execution engines. The platform helps reduce cost and time in business process development, increase reusability of business process models and leverage cloud elasticity in business process deployment and execution.

The platform has been used to develop paper-to-digital document workflows with a small number of services. As the next phase, the platform will be piloted by a group of selected users. User feedback will subsequently be used to evaluate and improve the platform.

References