# Predicting Service Composition Costs With Complex Cost Behavior

Robson W. A. de Medeiros[1,2,3]
[1]*Federal Rural University of Pernambuco*
*Department of Statistics and Informatics*
*Recife, Brazil*
*rwam@cin.ufpe.br*

Nelson S. Rosa[2]
[2]*Federal University of Pernambuco*
*Centre of Informatics*
*Recife, Brazil*
*nsr@cin.ufpe.br*

Luís Ferreira Pires[3]
[3]*University of Twente*
*Enschede, The Netherlands*
*l.ferreirapires@utwente.nl*

***Abstract*— Nowadays, many companies expose their competencies as services on the Internet to facilitate the cooperation with their customers. This situation has created a new marketplace where services have been provided with similar functionality but different qualities such as cost, performance, and reliability. In this scenario, service composition providers have faced the challenge of choosing services that fulfill an expected quality without compromising a planned budget. This challenge is even more stringent when services have different cost behaviors. As a consequence, services can be less expensive than others in a scenario and be more expensive in others. Several approaches have been proposed to address cost analysis of service compositions. However, these approaches have not considered all classes of possible cost behaviors, e.g., fixed, variable, mixed and step cost. This paper addresses this limitation by proposing a solution to analyze costs of service compositions taking into account service reliability and all classes of cost behaviors. In order to evaluate the proposed solution, we carried out some experiments that show its effectiveness.**

***Keywords*-Service-Oriented Computing; Service Composition Management; Cost; Cost Management; Reliability.**

## I. Introduction

The service composition process allows multiple services to be aggregated in order to create more complex ones, called service compositions [1]. In this paper, we are adopting the term service composition to both the process of composing services and to the resulting composite service. Once service compositions are implemented by combining other services, their quality and cost depend on the quality and cost of each atomic service involved in the composition. For instance, a single unsecure or unreliable service can affect the security and reliability of the whole service composition. Moreover, the presence or absence of these qualities can also affect the cost of the service composition. In the case of security, for instance, more secure services tend to consume more computational resources (e.g., CPU, memory, and bandwidth) needed to perform security activities (e.g., encryption and decryption) [2]. In the case of reliability, the failure of a component may demand more resources for the execution of corrective behaviors in the service composition, leading to an increase of cost.

* Partially sponsored by CNPq (Brazilian Research Council)

Business Process Model and Notation (BPMN) [3] is a standard widely used to model business process and service compositions. An important advantage of using BPMN to model service composition is that all stakeholders can understand this notation when they cooperate to model the compositions. Another advantage of the BPMN is that the service designers can analyze functional and non-functional requirements before the service composition is implemented.

Nowadays, providers offer services with different pricing schemes, like prices that vary according to the number of invocations and the size of the exchanged messages. Furthermore, in order to stimulate the service consumption and remain competitive, providers offer discounts that vary according to the volume of use of their services. This is the case of Amazon Simple Email Service (Amazon SES) [4], which is a large-scale e-mail solution to send marketing and transactional messages. To predict the cost of using this service, it is necessary to take into account three attributes of its utilization: number of sent messages, amount of data transferred and length of message's attachments. Moreover, by reaching a certain level of utilization, the service cost is reduced to stimulate its use.

Therefore, the planning of service compositions should consider all cost behaviors (i.e., variable, fixed, mixed and step cost) and identify situations that affect the profitability of the composition. To address this issue, some approaches have been proposed to compute cost of business processes and service compositions [5]. However, these approaches do not take into account all classes of cost behaviors that a service can have. In this paper, we contribute to the state-of-the-art by proposing an analytical approach to predict the cost of service composition taking into account all aforementioned classes of cost behavior. Moreover, since the reliability of a service can affect the behavior and cost of the whole composition, reliability is also considered in the cost computation. In practice, we model cost behaviors of all atomic services using the metamodel introduced in [6], and annotate the description of the service composition with their reliability, cost behaviors, and cost drivers information. Typically, cost driver is the level or volume of activity that directly affects the cost.

In order to demonstrate and evaluate our solution, we

978-1-4673-7281-7/15 $31.00 © 2015 IEEE
DOI 10.1109/SCC.2015.64

implemented a service composition and its necessary atomic services. We then simulated the behavior of both the atomic services and service composition. As result, we compared the cost computed by our approach against the cost computed separately for each atomic service.

The rest of this paper is organized as follows. Section II introduces the conceptual background of cost and reliability. Section III describes our approach to predict the cost of service compositions. Section IV presents the evaluation of our proposal. Next, Section V discusses related work. Finally, Section VI gives our conclusions and some directions for future work.

## II. Basic Concepts

In this section, we introduce the two key concepts used throughout the paper, namely cost and reliability.

### A. Cost

Cost is the entire expenditure required to create and sell products and services [7], and it is characterized in terms of classes of cost behavior. In [6], we developed a cost metamodel to model these classes.

*1) Cost behaviors:* Cost can be classified according to its behavior, which is usually referred to variable, fixed, mixed, and step cost [8] [7] [9]. *Variable cost* is only computed if the service is used and its value is directly proportional to the level of use of the service, as show in Figure 1(a). The factor that causes change in this cost behavior is known as cost driver. A cost driver is a unit of an activity that changes the cost of the service as the activity is executed. In contrast, *fixed cost* is a behavior that remains constant independently of the use of the service within a relevant period, like day, month or year, as show in Figure 1(b).

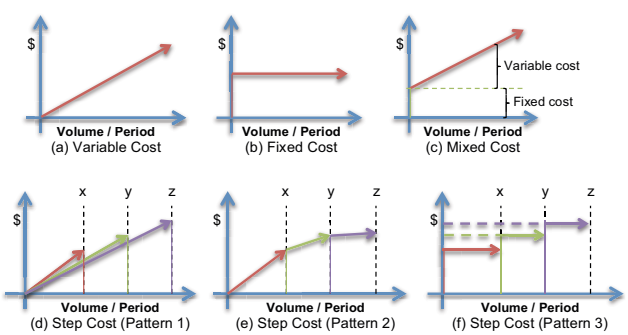


Figure 1. Classes of cost behavior.

*Mixed cost* behaviors combine variable and fixed costs. Mixed cost behavior starts with a fixed value that increases proportionally to the use of the service, as shown in Figure 1(c). *Step cost* is a class of cost behavior that can be divided into three patterns: (1) one in which the cost of an unit of a cost driver starts with a value and that is modified when the volume of the cost driver crosses some thresholds, as in Figure 1(d); (2) one in which a specific cost is defined for a volume unit of a cost driver in between each threshold, as in Figure 1(e); and (3) one in which the cost fixed within ranges of use of the service is defined and after each threshold the fixed cost is modified to a new value, see Figure 1(f).

*2) Cost metamodel:* In [6], we introduced a cost metamodel to model service cost behaviors in service compositions. *CostDriver*, *Rules*, *Choose*, and *Cost* are the main metaclasses of this metamodel. The metaclass *CostDriver* represents a cost driver that is directly associated to a *variable* and a *unit*. The *variable* represents a process attribute that should be used to compute the cost, for instance, '*message length*' and '*activity identification*'. The metaclass *unit* represents the unit of the variable, such as Megabyte, Minute, Hour, and so on.

The cost behaviors (i.e., fixed, variable, mixed and step cost) are defined in the metaclass *Rules*, which is composed of metaclasses *Cost* and *Choose*. Metaclass *Cost* defines the function to compute cost. Fixed cost behavior, for instance, is defined in metaclass *Cost*, by informing the monetary value charged for a certain amount of resource utilization, the currency of the value, the amount of resource utilization to which the value refers to, and the accounting period of the cost. Similarly, variable cost is defined by a value, a currency, an amount of resource utilization and an accounting period. Additionally, this cost behavior has also a cost driver that is necessary to compute the cost variation. Metaclass *Choose* defines the rules that should be applied to choose the appropriate *Cost*. Step cost and its thresholds are defined in the metaclass *Choose*. This metaclass contains one or more instances of a metaclass *When* and optionally one instance of a metaclass *Otherwise*. The metaclass *When* consists of a logical expression. If the logical condition holds, the costs defined in it have to be considered. Otherwise, a set of default costs defined in the Otherwise metaclass is used.

### B. Reliability

Reliability is often defined as the probability of a system to perform its intended function free of failures within a specified period of time [10]. We assume that unreliable services can harm the entire cost of a service composition [11] [12], since if a service fails, all services invoked previously have been already accounted for calculating the total cost.

The reliability of service compositions can be computed by recursively applying rules on workflow patterns of the composition behavior, such as the sequential, parallel, loop and conditional branching patterns [11] [13] [14]. These rules are intended to reduce the reliability of $n$ atomic services in one reliability, as if all services were reduced to a single one $s_r$. Figure 2 shows $n$ services ordered sequentially in a process, each one with reliability $R(s_i)$. In this case, the reliability of the reduced service $s_r$ can be computed

by multiplying the reliability of all services since failure probabilities are cumulative, as shown in Equation (1).

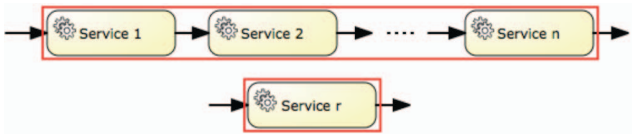


Figure 2. Sequential workflow reduction.

$$R(s_r) = \prod_{i=1}^{n} R(s_i) \tag{1}$$

When two or more services are executed in parallel, as shown in Figure 3, the sequence of execution cannot be guaranteed at design time. However, like the sequential pattern, all services must be executed correctly in the end of the parallel branches to enable the service composition to continue executing. Therefore, the reliability of this workflow pattern can also be computed by multiplying the individual probabilities of the $n$ parallel services by applying Equation (1).

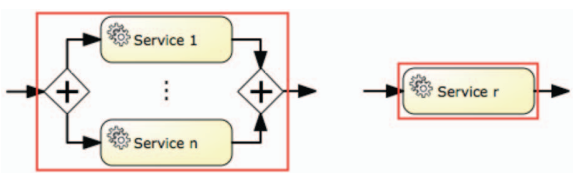


Figure 3. Parallel workflow reduction.

In the conditional branching pattern, in contrast, only one path is executed in a workflow execution. To analyze this pattern, we consider the probability of each path being executed. For instance, in Figure 4, each service has probability $w_i$ to be executed in the workflow and the sum of probabilities must be 100% since each service has different probabilitis to be executed and only one must be chosen among all options. Therefore, the reliability $R(s_r)$ of executing $n$ services in a conditional pattern can be computed as in Equation 2.

$$R(s_r) = \sum_{i=1}^{n} w_i R(s_i) \tag{2}$$

When a loop pattern has one or more services, these services can be invoked multiple times during the same process execution, depending on the condition defined in the service composition (see Figure 5).

To analyze this pattern, we consider either the number of iterations $n$ or the probability that the process moves either forward $w_j$ or backward $w_i$ to invoke again the services in the loop. Equation (3) shows the reliability of a service

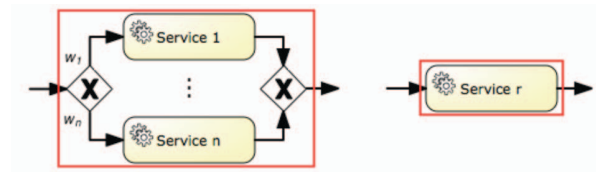


Figure 4. Conditional workflow reduction.

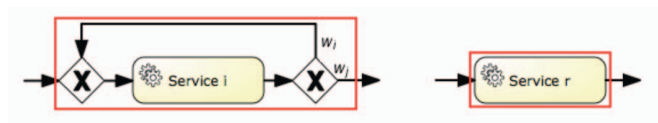


Figure 5. Loop workflow reduction.

$s_r$ can be computed by taking into account the number of iterations $n$. This equation can be adopted when the average number of iterations is known. Otherwise, Equation (4) can be adopted to compute the reliability taking into account the probability $w_i$ of a service $s_i$ getting into the loop.

$$R(s_r) = R(s_i)^n \tag{3}$$

$$R(s_r) = \frac{(1 - w_i) R(s_i)}{1 - w_i R(s_i)} \tag{4}$$

## III. Cost Prediction

Figure 6 shows the flow of tasks defined to predict the cost of service compositions. Firstly, we model the service composition in BPMN (Business Process Model and Notation) [3] (*Model Service Composition*) and the cost behavior of each individual atomic service of the composition (*Model Cost Behaviors*) using our metamodel (see Section II). We then annotate the service composition model with service reliability parameters (*Annotate Reliability*), the probability of each alternative path (*Annotate Conditional Statement Probabilities*) and cost information. The cost information consists of the cost behaviors (*Annotate Cost Behavior Identification*). In the case of variable cost, it is necessary to add the list of cost driver averages (*Annotate Cost Driver Values*).

With the service composition modeled and annotated with the aforementioned information, Algorithm 1 is executed (*Compute Cost Driver Values per Service Composition Execution*). Since the cost driver values annotated in each atomic service are independent from the service composition behavior, this algorithm computes the average cost driver values for each service composition execution. It takes into account the probability that the atomic service executes and terminates correctly, which depends on its reliability and the probability associated to the path that includes the invocation of the atomic service. After that, Algorithm 2 is executed to

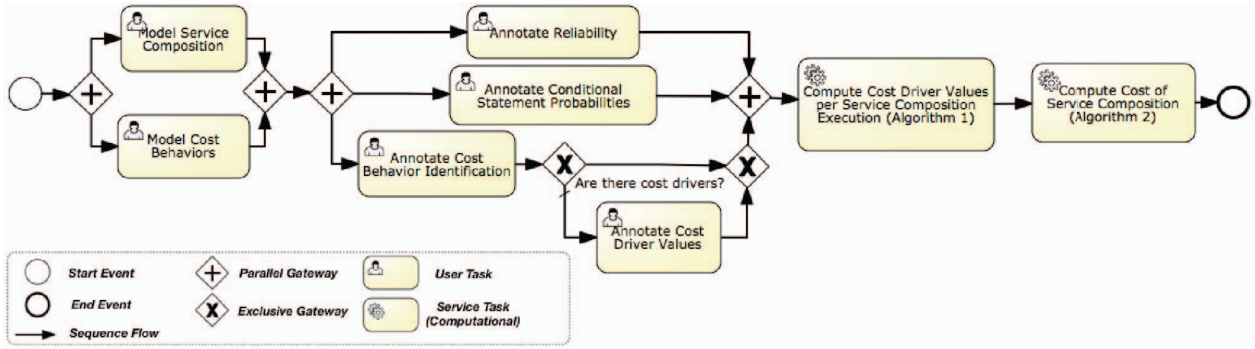


Figure 6. Process to compute cost of service composition.

compute the cost of the entire service composition (*Compute Cost of Service Composition*). This algorithm computes the service composition cost by taking into consideration the period of cost analysis, the cost behavior of each service, the average cost driver values associated to each service composition execution computed in the last task and the average number of the service composition execution for the period being considered.

**Algorithm 1** Compute average of cost driver per execution of service composition

1: **procedure** COMPUTECOSTDRIVERVALUE($BP$)
2: $CDV := empty$ ▷ A set of cost driver values per cost model
3: **for all** tasks $s_j$ in $BP$ **do**
4: $cm_k := s_j.getAnnotatedCostModel()$
5: $ACD_j := s_j.getAnnotatedCostDrivers()$
6: **for all** annotated cost drivers $acd_{ji}$ in $ACD_j$ **do**
7: $cdv_{ji} := PCE(s_j) * acd_{ji}.value$
8: $CDV.add(cm_k, acd_{ji}.name, cdv_{ji})$
9: **end for**
10: **end for**
11: Return $CDV$
12: **end procedure**

The input of Algorithm 1 is the annotated business process $BP$. Since there is no guarantee that a service is invoked when the composition is executed, the algorithm visits all tasks of the business process to compute the average of each cost driver value per execution of the service composition (Lines 3 - 10). For each task $s_j$ in the $BP$, the cost model ($cm_k$) and the annotated cost drivers ($ACD_j$) are obtained. Next, the cost driver average execution value is computed (Line 7) taking into account the probability that the service terminates correctly during the execution of the service composition ($PCE$) and the value of the cost driver ($acd_{ji}$) annotated in the business process. Consider task $s_j$ with reliability $R(s_j)$, $s_i$ a task executed before $s_j$ with reliability $R(s_i)$, and $w_{ij}$ the probability that $s_j$ is executed after $s_i$, the $PCE$ (Probability of Correct Execution) of task $s_j$ is computed as in Equation 5.

$$PCE\,(s_j) = PCE\,(s_i) \times w_{ij} \times R(s_j), \quad \text{where } PCE\,(s_i) \times w_{ij} = 1.0 \text{ if } s_j \text{ is the first task.} \tag{5}$$

When $s_j$ is preceded by a set of tasks in sequence (see Figure 2), in parallel (see Figure 3), conditionally (see Figure 4) or in a loop (see Figure 5), Equation (1), Equation (2), Equation (3), and Equation (4) can be applied to reduce the reliability of the set of tasks into a unique reliability in order to compute $PCE(s_j)$.

Since the same service can be adopted to perform more than one task, the average execution value of the cost driver is represented as a tuple $< key, value >$ (Line 8), where $key$ is defined in terms of the cost model ($cm_i$) and cost driver identifications ($cd_i$), and $value$ is computed by multiplying the value of the cost driver by the $PCE$ of the respective task. This tuple is stored in a set of cost driver values ($CDV$) whose $key$ must be unique. Therefore, when a new tuple needs to be stored and its $key$ already exists, its $value$ is added to the existing one. Otherwise, a new tuple is added to $CDV$.

Algorithm 2 uses the average of the execution values of the cost drivers ($CDV$) and the set of cost models ($CM$) annotated in the business process to compute the cost of the service composition. This computation considers the period of analysis ($periodOfAnalysis$) and the mean number of executions ($n$) in the period, e.g., 10 executions per day. $CM$ has no duplicate objects and this ensures that if one service is adopted to perform more than one task, it is considered only once, rather than multiple times. However, $CDV$ contains the sum of all the values of the cost drivers spent in each task that invoke the respective service, which ensures the correct cost calculation.

Each cost model ($cm_k$) has a set of cost functions ($c_i$) that define how to compute the service cost. Moreover, $c_i$ can be related to a cost driver. In this case, the value of the cost driver is calculated as shown in Line 8 of the Algorithm 2. In

**Algorithm 2** Compute cost of service composition

1: **procedure** COMPUTESERVICECOMPOSITIONCOST($BP$, $CDV$, $periodOfAnalysis$, $n$)
2: $CM := BP.getCostModels()$
3: $costValue := 0.0$
4: **for all** $cm_k$ in $CM$ **do**
5: **for all** cost $c_i$ in $cm_k$ **do**
6: $periodRate := \frac{periodOfAnalysis}{c_i.getPeriod()}$
7: **if** (there is cost driver in $c_i$) **then**
8: $costDriver := c_i.getCostDriver() * periodRate * n$
9: **end if**
10: **if** (($c_i$ is in a choose element) AND (condition is $true$)) OR ($c_i$ is NOT in a choose element) **then**
11: **if** $costDriver$ is NOT $NULL$ **then** ▷ Variable cost
12: $costValue := costValue + \frac{costDriver}{c_i.getAmount()} * c_i.getValuePerAmount()$
13: **else** ▷ Fixed cost
14: $costValue := costValue + c_i.getValuePerAmount() * periodRate$
15: **end if**
16: **end if**
17: **end for**
18: **end for**
19: **return** $costValue$
20: **end procedure**

addition, conditions can be used to define the proper function to be adopted. If that is the case, we must first evaluate the condition before using the cost function as shown in Line 10. When the cost function can be used, we verify if there is a cost driver associated to the cost function (Line 11). In the case of the $c_i$ has a cost driver, the cost function has a variable behavior (Line 12). Otherwise, it has a fixed behavior (Line 14). In the case of variable behavior, the cost is computed as shown in Equation 6.

$$VariableCost = \frac{costDriverValue}{amount} * valuePerAmount \quad (6)$$

where, $costDriverValue$ is the cost driver value accumulated in the period of analysis and the $valuePerAmount$ is the value that is charged to consume an amount of a cost driver ($amount$).

In the case of fixed behavior, the cost function only depends on the period of analysis ($periodRate$). Moreover, it has the same value ($ValuePerAmount$), which is independent of the use of the service, as shown in Equation (7).

$$FixedCost = valuePerAmount * periodRate \quad (7)$$

## IV. Experiment

We conducted an experiment in which some services are composed in order evaluate our approach. All atomic services and the service composition were fully implemented and simulated in this experiment. Moreover, we assigned values that we found reasonable to the costs of the atomic services.

In order to evaluate our approach, the cost of the composition was calculated in two different ways: by adding up the cost computed directly for each atomic service and by applying the proposed approach. Then, we compare both costs to evaluate the results obtained with our approach.

### A. Application

The application consists of a service composition that converts audio files from WAV to MP3 (see Figure 7). The process starts when the customer invokes the composition by informing her e-mail address and the WAV file to be converted. The e-mail address is validated by the first service (*Validate Email Address*) and, if it is valid, the WAV file is converted into the MP3 file format by another service (*Convert WAV to MP3 Audio Format*). In contrast, if the e-mail address is invalid, the service composition sends an e-mail to notify the composition administrator, informing that an error has happened (*Notify Administrator by Email*). If the file is converted correctly, the customer receives an e-mail with the file in the new format (*Send Email to Customer with the MP3 File*).

Each task of the service composition is performed by a web service with the cost behaviors shown in Table I.

### B. Simulation

All services were implemented in Java and deployed in a Tomcat server running on an instance of t2.micro virtual machine of Amazon EC2. Moreover, all service data were persisted in a MySQL database [15] installed in the same virtual machine.

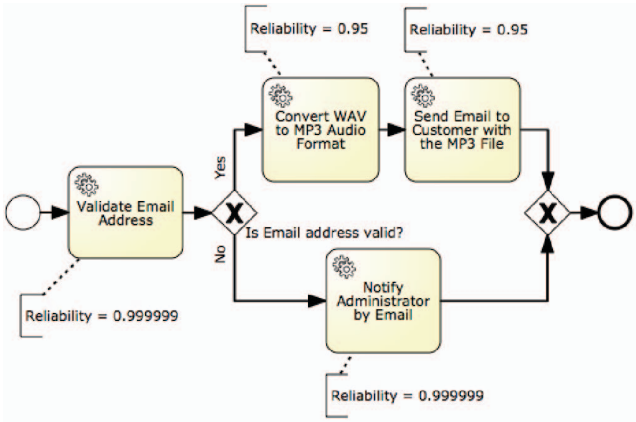


Figure 7. Test case business process with cost annotations.

Table I
COST BEHAVIORS

| Tasks | Service | Cost Model |
|---|---|---|
| *Validate Email Address* | **ValidateEmail** | $0,05 per invocation. |
| *Notify Administrator by Email* | **SimpleEmail** | $2,00 per month. |
| *Convert WAV to MP3 Audio Format* | **WAVtoMP3** | $10.00 per month; and, $0,03 per 1 MB of file. |
| *Send Email to Customer with the MP3,File* | **CompleteEmail** | $0.10 per 100 emails / month; and, Attachments: First 10 GB/month: $0.12/MB, Greater than 10 GB/month: $0.09/MB |

The service composition was modeled and executed using Activiti [16]. The *SimpleEmail* service was used to perform task *Notify Administrator by Email*. Task *Validate Email Address* was performed by web service *ValidateEmail* with a variable cost that depends on the number of invocations. In order to simulate different scenarios, we forced that 95% of the e-mails sent to this service were valid and 5% of them were invalid. Service $WAVtoMP3$ realized the task *Convert WAV to MP3 Audio Format* and has a mixed cost with a fixed cost of $20,00 per month plus a variable cost driver depending of the file size. In relation to its reliability, this service converts 95% of files correctly and 5% incorrectly. Moreover, the size of the WAV files sent to this service followed an uniform distribution between 1 and 5 MB. For the MP3 files sent back to the customer by the task *Send Email to Customer with the MP3 File*, we assumed that they had a compression ratio following a normal distribution with mean equal to 0.20622 and standard deviation equal to 0.04755 with respect to the corresponding WAV file. These parameters were obtained by observing some WAV to MP3 conversion rates. Finally, we assumed that 5% of the e-mail sending attempts failed.

In order to simulate the number of daily service composition executions, we generated a random number in between 20 and 50 with a continuous uniform distribution function. As result of this function, we obtained the value 33, which means that the service composition in our experiment has an average of 33 daily executions. Therefore, for each simulated day, the number of the service composition executions was determined by a normal distribution function with mean 33 obtained previously and standard deviation 1, which we assumed to compute the distribution.

### C. Predicting cost

By using the approach presented in Section III, we first modeled the service composition in BPMN (*Model Service Composition*) and the cost behavior of all services using our cost metamodel (*Model Cost Behaviors*). Next, we annotated the business process with the cost behaviors (*Annotate Cost Behavior Identification*), cost driver values (*Annotate Cost Driver Values*), task reliability (*Annotate Reliability*) and probabilities to all alternative paths of the business process (*Annotate Conditional Statement Probabilities*). After the business process was annotated, we executed the Algorithm 1 to compute the probability of correct execution ($PCE$) of each service and the average cost driver values per service composition execution (*Compute Cost Driver Values per Service Composition Execution*). These cost driver values are obtained by multiplying $PCE$, the average value per invocation of each cost driver of the respective service and the number of invocations in the period as shown in Table II. Finally, Algorithm 2 was executed to compute the cost of each service according to its respective cost behavior (*Compute Cost of Service Composition*).

Table II
COST PREDICTION OF SERVICE COMPOSITION

| Service | PCE | Cost Driver Values | Cost |
|---|---|---|---|
| *ValidateEmail* | 1.0000 | Invocations: 990 | $49.50 |
| *SimpleEmail* | 0,0500 | ———————— | $2.00 |
| *WAVtoMP3* | 0.9025 | Files: 2,680Mb | $90.40 |
| *CompleteEmail* | 0.8574 | Email: 849<br>Attachments: 525.25Mb | $63.88 |
| | | **Total Cost** | **$205.78** |

### D. Hypothesis Testing

In our experiment, we simulated the cost of 30 days of service composition execution in order to test the hypothesis test: 'the mean cost calculated by service providers invoked by the composition ($\eta_1$) is significantly different from the one predicted by our approach'. For this purpose, we generated 20 examples of the cost computed by services providers and compared them with the cost computed by our approach. The parameters of our hypothesis testing are the following:

- Mean cost calculated by service providers invoked by the composition: ($\eta$)
- Cost predicted by our approach: $205.78

In order to identify which hypothesis test should be applied in our experiment, we tested if the examples collected fitted a normal distribution. Since the P-value we obtained was 0.228, we can not reject the hypothesis that the data follows a normal distribution with 95% of confidence level, mean 205.4 and standard deviation equal to 2.503, as shown in Figure 8.

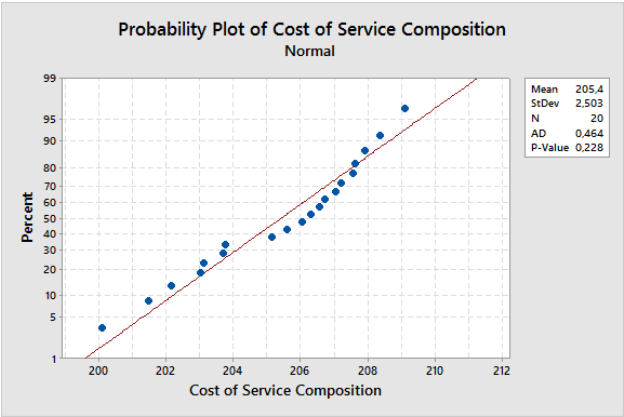


Figure 8. Testing the normalization of the examples.

Therefore, we applied the Z-test with 95% of confidence level and standard deviation 2.503 to test the hypotheses:

- Null Hypothesis ($H_0$): $\eta = 205.78$
- Alternative Hypothesis ($H_1$): $\eta \neq 205.78$

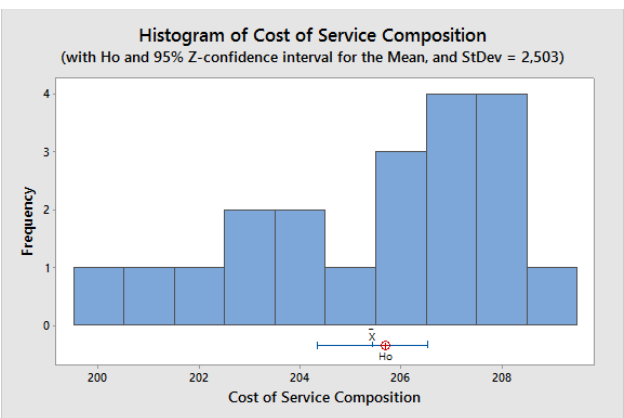


Figure 9. Histogram of the cost examples for 33 invocations per day.

According to the result of the hypothesis test (see Figure 9), with P-value equal to 0.645, we cannot reject the null hypothesis that the cost computed directly by the service providers is equal to 205.78, which is the cost computed by our approach.

## V. Related Work

BPMN [3] is a standard that allows organizations to graphically model their business processes as service compositions and analyze these processes before they are implemented. Magnani et al. [17] proposed an approach to analyze the cost of business processes by adding cost properties to BPMN elements. Similarly to our solution, they also add probabilities to all alternative branches in the process. However, only the average cost is annotated in each element of the BPMN, and neither reliability nor all cost behaviors are considered.

Saeedi at al. [12] proposes a BPMN extension to predict cost. Unlike [17], Saeedi's approach is also able to predict performance and reliability of service composition. However, different from our approach, they do not take into account all cost behaviors and the reliability and performance of the services are not used to compute the cost of the service composition.

Sampathkumaran [11] also proposes a BPMN extension to predict the cost of business processes. Additionally, this approach takes into account the reliability of each task when computing the cost of the service compositions. Like other approaches, only the average cost is annotated in the BPMN task, and the cost of the services is computed by only considering the number of invocations. In our approach, instead of annotating the average cost, we annotate the cost behavior of each task.

The work of Wynn at al. [18] presents a framework for reporting and predicting the cost of business processes with cost model annotations. Moreover, their approach uses ProM [19] as part of their framework for accounting business process cost. To compute the process cost, ProM receives cost-annotated event logs enriched with detailed cost information and a business process cost model. In the absence of a log, the framework generates logs that can be used to predict cost. In their work, they mention that the framework computes variable, fixed and mixed cost. However, only variable costs are supported since they treat costs per invocation as fixed costs. In our approach, we do not use process mining nor logs to compute cost of service compositions. However, we consider that business execution logs should not be annotated with explicit cost values, since in step cost behavior the cost can vary during the accounting period, for example, due to discounts and execution failures.

Table III shows a comparison of all the works mentioned in this section and shows that our contribution advances by considering all classes of cost behavior to compute cost of service compositions.

## VI. Conclusion

In this paper, we proposed an analytical approach to predict the cost of service compositions. Unlike related work, our approach takes into consideration all classes of cost behaviors, i.e, variable, fixed, mixed and step cost. Moreover, as reliability can affect the cost of the entire composition, we also take into account reliability in the cost computation.

Table III
COMPARISON OF COST PREDICTION APPROACHES.

| Approach | Cost Behaviors | | | | Non-Functional Properties |
|---|---|---|---|---|---|
| | Variable | Fixed | Mixed | Step | Reliability |
| **[17]** | Yes | No | No | No | No |
| **[12]** | Yes | No | No | No | No |
| **[11]** | Yes | No | No | No | Yes |
| **[18]** | Yes | No | No | No | Yes (implicit) |
| **Our** | Yes | Yes | Yes | Yes | Yes |

In our approach service compositions are not annotated with simple cost information. Instead, we prescribe the annotation of the cost behavior of each service, as well as the average values of the cost drivers involved in service invocations. Since many costs can have their behavior modified over time, our approach calculates the cost of the composition by taking into account the average number of service composition executions within an accounting period.

In order to evaluate our approach, we carried out experiments to simulate the cost of service composition executions. In the experiment, we tested the hypothesis that the mean cost calculated by adding up the cost of each atomic service was significantly different from the one predicted by our approach. As result, we asserted that the cost computed by our approach is statistically similar to the cost computed directly by the atomic service providers, which showed us the accuracy of our approach.

In future work, we intend to apply our approach in a real world case study. Moreover, we intend to work on simulation techniques to predict the cost of service compositions by taking into account all classes of cost behaviors. We believe that simulation can improve even more the accuracy of the cost prediction in this case. Furthermore, we also intend to investigate algorithms to select optimal sets of services in a service composition taking into account these classes of cost behavior.

## REFERENCES


[1] M. P. Papazoglou and D. Georgakopoulos, *Service-Oriented Computing*. The MIT Press, Nov. 2008.

[2] E. Spyropoulou, T. Levin, and C. Irvine, "Calculating costs for quality of security service," in *Computer Security Applications, 2000. ACSAC '00. 16th Annual Conference*, Dec. 2000, pp. 334–343.

[3] "Business process model and notation (BPMN)." [Online]. Available: http://www.omg.org/spec/BPMN/

[4] "AWS | amazon simple email service (SES) | pricing." [Online]. Available: //aws.amazon.com/ses/pricing/

[5] R. Albuquerque de Medeiros, N. Souto Rosa, G. Medeiros Campos, and L. Ferreira Pires, "A survey of cost accounting in service-oriented computing," in *2014 IEEE World Congress on Services (SERVICES)*, Jun. 2014, pp. 77–84.

[6] R. W. A. Medeiros, N. S. Rosa, and L. F. Pires, "A metamodel for modeling cost behavior in service composition." Doha, Qatar: IEEE, Oct. 2014.

[7] S. M. Bragg, *Cost Accounting Fundamentals: Essential Concepts and Examples*. Centennial, Colo.: AccountingTools, Jan. 2012.

[8] C. T. Horngren, S. M. Datar, and M. V. Rajan, *Cost Accounting: A Managerial Emphasis*, 14th ed. Upper Saddle River, N.J: Prentice Hall, Jan. 2011.

[9] D. R. Hansen and M. M. Mowen, *Cornerstones of Cost Management*, 2nd ed. Mason, OH: Cengage Learning, Jun. 2012.

[10] M. Xie, K.-L. Poh, and Y.-S. Dai, *Computing System Reliability: Models and Analysis*, 2004th ed. New York: Springer, Apr. 2004.

[11] P. B. Sampathkumaran, "Computing the cost of business processes," Ph.D. dissertation, Mnchen, Ludwig-Maximilians-Universitt, Diss., 2013, 2013. [Online]. Available: http://d-nb.info/103707632X/34

[12] K. Saeedi, L. Zhao, and P. Sampaio, "Extending BPMN for supporting customer-facing service quality requirements," in *2010 IEEE International Conference on Web Services (ICWS)*, Jul. 2010, pp. 616–623.

[13] J. Cardoso, J. Miller, A. Sheth, and J. Arnold, "Quality of service for workflows and web service processes," *Journal of Web Semantics*, vol. 1, pp. 281–308, 2004.

[14] C. Xie and J. Ren, "A dynamical reliability prediction algorithm for composite service." [Online]. Available: http://downloads.hindawi.com/journals/mpe/aip/917903.pdf

[15] Oracle, "MySQL." [Online]. Available: http://www.mysql.com

[16] T. Rademakers, *Activiti in Action: Executable business processes in BPMN 2.0*, 1st ed. Shelter Island, NY: Manning Publications, Jul. 2012.

[17] M. Magnani and D. Montesi, "BPMN: How much does it cost? an incremental approach," in *Business Process Management*, ser. Lecture Notes in Computer Science, G. Alonso, P. Dadam, and M. Rosemann, Eds. Springer Berlin Heidelberg, Jan. 2007, no. 4714, pp. 80–87.

[18] M. T. Wynn, W. Z. Low, A. H. M. ter Hofstede, and W. Nauta, "A framework for cost-aware process management : cost reporting and cost prediction," *Journal of Universal Computer Science*, Aug. 2013. [Online]. Available: http://eprints.qut.edu.au/63036/

[19] W. M. P. v. d. Aalst, *Process Mining: Discovery, Conformance and Enhancement of Business Processes*, 2011th ed. New York: Springer, Apr. 2011.