

# A Framework for Linking and Pricing No-Cure-No-Pay Services

K.M. van Hee<sup>1</sup>, H.M.W. Verbeek<sup>1</sup>, C. Stahl<sup>2,1,\*</sup>, and N. Sidorova<sup>1</sup>

<sup>1</sup> Department of Mathematics and Computer Science,  
Eindhoven University of Technology  
P.O. Box 513, 5600 MB Eindhoven, The Netherlands  
{k.m.v.hee,h.m.w.verbeek,n.sidorova}@tue.nl

<sup>2</sup> Humboldt-Universität zu Berlin, Institut für Informatik  
Unter den Linden 6, 10099 Berlin, Germany  
stahl@informatik.hu-berlin.de

**Abstract.** In this paper, we present a framework that allows us to orchestrate web services such that the web services involved in this orchestration interact properly. To achieve this, we predefine service interfaces and certain routing constructs.

Furthermore, we define a number of rules to incrementally compute the price of such a properly interacting orchestration (i.e. a web service) from the *price* of its web services. The fact that a web service gets only paid after its service is delivered (no-cure-no-pay) is reflected by considering a probability of success. To determine a safe price that includes the *risk* a web service takes, we consider the variance of costs.

## 1 Introduction

In the future, the world of web services will most likely behave in a way similar to business units in a real economy. Web services will sell services to their clients, that may be web services themselves. In order to produce these services, they may buy services from other web services. So each web service has two (inter)faces: a *sell-side* and a *buy-side*. In order to deliver one service, a tree of other web services forming a *supply chain* may be needed. Two services in the chain are linked if the buy-side of the one is connected to the sell-side of the other. When designing web services, one should take into account that the execution of a partner web service may be successful or it may fail. Examples of web services that might be used by other web services are public web services providing the stock market prices, weather forecasts, and news items. Other examples of web services that could be used in supply chains are computations, such as actuarial computations and risk analysis computations. Also services providing movies, television, and telephone services could be part of such a web service supply chain.

Web services will be paid for by a system of micro payments [1]. We assume a no-cure-no-pay system, which implies that a service is paid for if and only if

---

\* Funded by the DFG project “Substitutability of Services” (RE 834/16-1).

the service is successfully delivered. There are at least two important questions for such world of services: Firstly, how do we guarantee that two linked services work properly together? Secondly, what should be the price of service in order to be profitable?

For the first question we have to define what it means for two services to work properly together. Informally, this means that a service request is always followed by a response: Either the requested service is delivered, or a message is sent saying that the requested service cannot be delivered. We provide in this paper a framework where this property is guaranteed. In our framework, each web service contains a *sell-side process* that represents the protocol for selling a service. Internally, there is a process for the *orchestration* of the services. A service is built up from tasks that are either elementary or compound. An elementary task is either executed by the web service itself or it is outsourced to another web service. In the latter case the task contains a *buy-side process* that takes care of the purchase of a service. The sell-side of the service and buy-side processes of all the tasks of the orchestration together form the *choreography* of the world of web services. They are designed in such a way that two arbitrary services fit together properly.

To answer the second question, we devote the second part of the paper to the *pricing mechanism* for services. For the price of a service we have to take into consideration that a service may fail. Since we assume a no-cure-no-pay system, a web service may have to pay all or some of its service providers while it fails to deliver its service to its client. This can be due to stochastic phenomena, such as the availability of servers. That way, a web service takes *risks*. Therefore, it is a nontrivial problem to determine the price of a service. We develop a method to determine the expectation  $\mu$  and the variance  $\sigma^2$  of the cost of the service. Assuming a normal distribution for service cost  $C$ , we may say that the cost of service is less or equal to  $\mu + 1.65 \cdot \sigma$  with probability 95%. This assumption may be justified by the fact that a service is built up by several independent services and by the famous central limit theorem of statistics, the convolution of independent identical distributions can be approximated by a normal distribution. We also provide an exact probabilistic bound for the cost of a service.

The paper is structured as follows. Section 2 introduces Petri nets and some probability notions. The framework for linking web services is presented in Sect. 3. Here, web services are considered as components and the choreography and orchestration processes are modeled as Petri nets. Then, in Sect. 4, we present our pricing model for web services. We show how to compute the expected cost and the cost variance of a web service in an inductive way. Related work is presented in Sect. 5 and finally, Sect. 6 concludes the paper.

## 2 Preliminaries

**Petri nets and workflow nets** We assume the usual definition of a (place/transition) Petri net  $N = (P, T, F)$  (see [2, 3], for instance) and use the standard

notation to denote the preset and postset of a place or a transition:  $\bullet x = \{y \mid F(y, x) > 0\}$  and  $x^\bullet = \{y \mid F(x, y) > 0\}$ .

The state of a net is a marking  $m$  which is a distribution of (black) *tokens* over the set of places. A transition  $t$  is *enabled* if each place  $p$  of its preset holds at least  $F(p, t)$  tokens. An enabled transition  $t$  can *fire* in a marking  $m$  by consuming  $F(p, t)$  tokens from the every preset place  $p$  and producing  $F(t, q)$  tokens for every postset place  $q$ , yielding a marking  $m'$ . The firing of  $t$  is denoted by  $m \xrightarrow{t} m'$ , reachability of a marking  $m'$  from a marking  $m$  by a firing of a (possibly empty) sequence of transitions is denoted by  $m \xrightarrow{*} m'$ . The set of all reachable markings is denoted  $N[m]$ .

A *workflow net* (WF-net) [4] is a Petri net  $N$  that is specifically tailored towards modelling workflow processes. A workflow net  $N = (P, T, F)$  has exactly one input place  $i$ , i.e.  $\bullet i = \emptyset$ , and one output place  $o$ , i.e.  $o^\bullet = \emptyset$ , and every place and transition belongs to some path from  $i$  to  $o$ .

An important correctness property of workflows is *soundness* [4], which comprises the requirements that for every transition sequence that fires starting from the initial marking a marking can be reached where only  $o$  is marked, and no transition is dead in  $N$ . Soundness can be automatically checked by a number of Petri net tools, like the tool Woflan [5, 6].

**Some notions of probability** A service consists of tasks. Each task  $t$  has a random variable  $X_t$  with a Bernoulli distribution:  $X_t = 1$  indicates success whereas  $X_t = 0$  indicates failure. Furthermore, the probability of success for a task  $t$  is  $\mathbb{P}[X_t = 1] = s_t$ .

Each task  $t$  (either elementary or compound) has a random variable  $C_t$  denoting the cost of the task. In case  $t$  is a compound task,  $C_t$  includes the costs of the tasks it contains. This random variable depends in general on  $X_t$  and has the following characteristics:

- $\mathbb{E}[C_t] = \mu_t$ , cost expectation,
- $\sigma^2(C_t) = \sigma_t^2$ , cost variance.

In Sect. 4 we will see that, to compute the cost variance, we need to compute the following auxiliary variable:

- $\mathbb{E}[C_t | X_t = 1] = \nu_t$ , cost expectation in case of success.

We use the well known inequality of Chebyshev [7] for a non-negative random variable  $Y$

$$\mathbb{P}[Y \geq c] \leq \frac{\mathbb{E}[\varphi(Y)]}{\varphi(c)}$$

for each non-decreasing function  $\varphi$  and any positive number  $c$ . In particular,

$$\mathbb{P}\left[\left|\frac{C_t - \mu_t}{\sigma_t}\right| \geq c\right] \leq \frac{\mathbb{E}\left[\left(\frac{C_t - \mu_t}{\sigma_t}\right)^2\right]}{c^2} = \frac{1}{c^2}$$

and therefore

$$\mathbb{P}[C_t \geq \mu_t + c \cdot \sigma_t] \leq \mathbb{P}\left[\left|\frac{C_t - \mu_t}{\sigma_t}\right| \geq c\right] \leq \frac{1}{c^2}.$$

**Running example** We will restrict ourselves to orchestration processes that are constructed using the four routing constructs sequence, parallel composition, choice, and iteration. Therefore, we can present an orchestration process in a tree-like fashion, where the leaf nodes in the tree correspond to elementary tasks and the non-leaf nodes correspond to compound tasks, where every compound task uses one of the four routing constructs.

This allows us to represent the orchestration processes in a process algebraic notation where the sequence operator is denoted by  $\cdot$ , parallel composition by  $\parallel$ , choice by  $+$ , and iteration by  $*$ . We will assume that  $*$  binds strongest, then  $\cdot$ , then  $+$ , and last  $\parallel$ . This notation will be very useful for the derivation of the cost formulas in Sect. 4.

As running example, consider an image editing process  $p$ . The customer uploads an image (task  $u$ ). Then, the following procedure is applied: The image is finished (task  $f$ ) and concurrently a thumbnail is created (task  $t$ ). Afterwards the results are evaluated (task  $e$ ). In case the evaluation is negative, the procedure is repeated. Finally, if the image is too big, it is stored temporarily and only a link is sent to the customer (task  $l$ ); otherwise the image is sent by e-mail (task  $m$ ). The process can be expressed in our process-algebraic notation as follows (note that we have to unfold the iteration  $(f||t) \cdot e$  once as the iteration is executed at least once):  $p = u \cdot (f||t) \cdot e \cdot ((f||t) \cdot e)^* \cdot (l + m)$ .

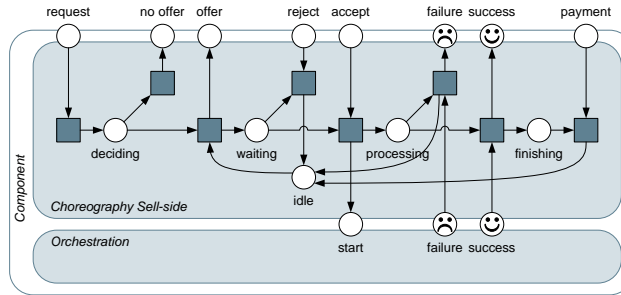
For the example, we assume that the following characteristics are given:

- In 90% of the cases, the upload succeeds.
- In 95% (80%) of the cases, finishing (creating the thumbnail) succeeds which costs the customer €20 (€10).
- In 99% of the cases the evaluation succeeds which costs another €5; in only 20% of the cases the loop has to be repeated.
- In 30% of the cases, the image is too big; the storing is charged with €10 whereas the sending of the images costs €1.

Note that we have two kinds of probabilities: the probability of a choice in the orchestration, here used for the iteration, and the probability of failure of a service. If some task of a service fails, the whole service fails. The question now is, what should the service provider charge the consumers to make any profit, and what are the risks involved? In the following, we present a mathematical framework to answer this question.

### 3 Web Services Framework

In the web services domain, Petri nets provide a good formalism to model web services that communicate with each other through an interface, which is typically modeled by *interface places* (see [8], for example). For this reason, we distinguish interface places from other places. An interface place can either be an input place or an output place. An input place has no (internal) input arcs, whereas an output place has no (internal) output arcs. As a result, a web service is not allowed to communicate with itself.



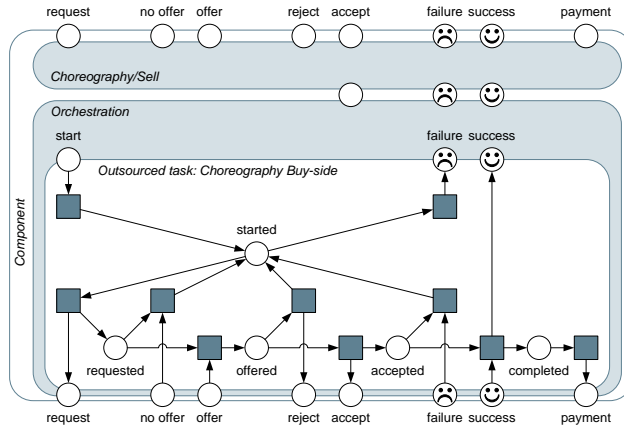
**Fig. 1.** The sell-side of a web service

A web service has a *sell-side process* and an *orchestration process*. The orchestration process has to be executed in order to deliver the service. The orchestration process consists of tasks. We distinguish *elementary* tasks, *outsourced* tasks, and *compound* tasks. An elementary task is modeled as a Petri net transition. An outsourced task is in fact a subnet that contains a *buy-side process*. A compound task is a subnet containing elementary or outsourced tasks. In fact, the whole orchestration process can be considered as one (compound) task. As mentioned in the introduction, we can distinguish between a sell-side and a buy-side of a web service. Typically, when some service wants to use another service, that is, if the former consumes a service that is being provided by the latter, the former first requests a quote from the latter. Based on the offer from the provider (which is optional, as the provider may decide not to offer the service), the client decides to either accept or reject the offer. If the offer is accepted, the provider actually provides the service, which might either succeed or fail. This result is communicated to the client, which pays the provider if the result was a success (no-cure-no-pay).

To provide the service, the provider might have to consume third-party services in some order. Clearly, the provider needs to *orchestrate* these third party services on-the-fly to achieve its goal. In contrast, the negotiation between the provider and the customer belongs to the *choreography process*.

**Choreography** The choreography in the framework consists of the sell-side process of the web service and the buy-side processes of the outsourced tasks. Figure 1 visualizes the sell-side of a web service, whereas Fig. 2 visualizes the buy side of a task. As usual, circles represent places and squares represent transitions. For the ease of reference, sad smileys have been used for the failure places and happy smileys for the success places.

Underlying assumption for the sell-side is that it can handle a predefined number of requests simultaneously. This number corresponds to the number of tokens which are initially put into the place *idle*. Thus, if the maximum number of requests is being handled, then no offer can be made for the next request.



**Fig. 2.** A task in the buy-side of a web service

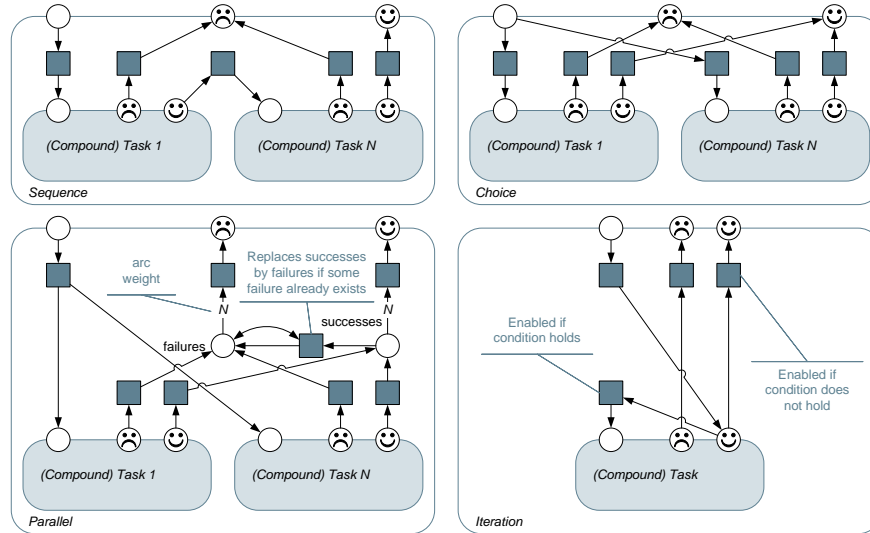
The buy-side of an outsourced task contains a buy-side process in which first a request for a proposal is sent to a potential supplier. The process is the “mirror” of the sell-side process except that the buy-side process handles requests in a sequential order.

**Orchestration** The web service can perform an *elementary task* like a computation, an *outsourced task* that requests a service from another web service like retrieving some information from an underlying database, or a *compound task* that needs to orchestrate a number of (sub)tasks. In principle, such an orchestration can be arbitrarily complex, but in this paper we consider four operations to construct compound tasks:

1. *sequence*, i.e. performing a number of tasks in a given order,
2. *parallel composition*, i.e. performing a number of tasks simultaneously,
3. *choice*, i.e. performing one task chosen from several tasks, based on some decision, and
4. *iteration*, i.e. performing a task as long as some condition holds.

Nevertheless, we would like to stress that the framework can be extended with additional operations if needed (as long as soundness is guaranteed, see further down in this section). Reason for restricting to this set of operations in this paper is that these four types are sufficient to explain the matters at hand, whereas additional ones might only distract the reader.

Figure 3 visualizes these four basic orchestration types. Again, we use the sad smileys for the failure places and the happy smileys for the success places, and we use grey boxes to visualize the orchestrated tasks. For the sake of simplicity, we used only two tasks for the sequence, parallel composition, and choice, but it is not hard to see that this scheme can be extended to any number.



**Fig. 3.** The basic orchestration operations

Figure 4 depicts the example service using the web service framework, containing one choreography component (the sell-side process of the service), nine orchestration components (which are compound tasks), and nine elementary tasks. Six of the elementary tasks are outsourced tasks, which form the buy-side processes of the service.

**Soundness** Clearly, any task should lead to either a success or a failure, i.e. if a task starts with one token in its start place, then for any reachable marking it should be possible to reach the marking with only one token either in place success or in place failure. We call this the *soundness of a task*. This soundness concept differs a little from the one given in Sect. 2, namely we have two final places (success and failure) instead of one. By the construction of Fig. 5 these notions can be unified. In order to analyze soundness of a task, we distinguish two parts: soundness of the choreography part and of the orchestration part.

For the choreography part we connect the buy-side of one task to the sell-side of another service as displayed in Fig. 6. We first neglect the three places *start*, *failure* and *success* at the bottom. Then, soundness is not difficult to verify, e.g. by brute force model checking since the set of reachable markings is finite. It is also easy to verify this property by observing that the upper (the buy-side) and lower part (sell-side) of the model are almost symmetrical state machine nets where each choice is made by only one of them. When considering places *start*, *failure* and *success* at the bottom, we need the assumption that if a token is put into *start*, eventually there will be a token in either *failure* or *success*. Under this assumption the system is sound. So we introduce here the concept of *conditional soundness*: the choreography part is sound if the invoked tasks are sound.

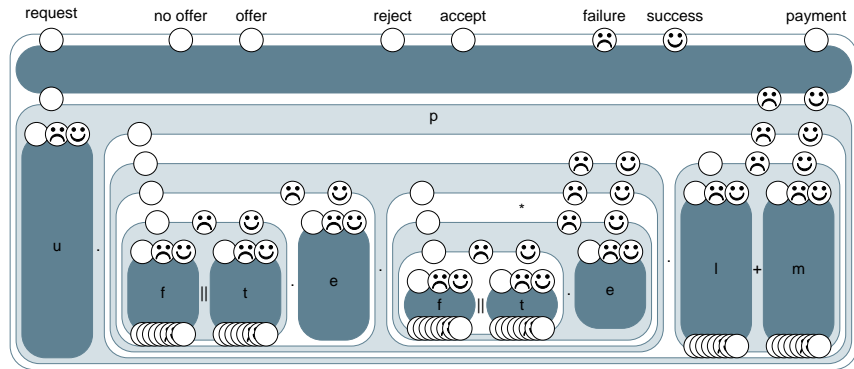


Fig. 4. The running service using the framework

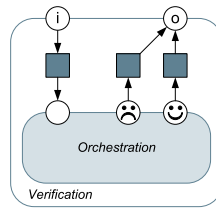
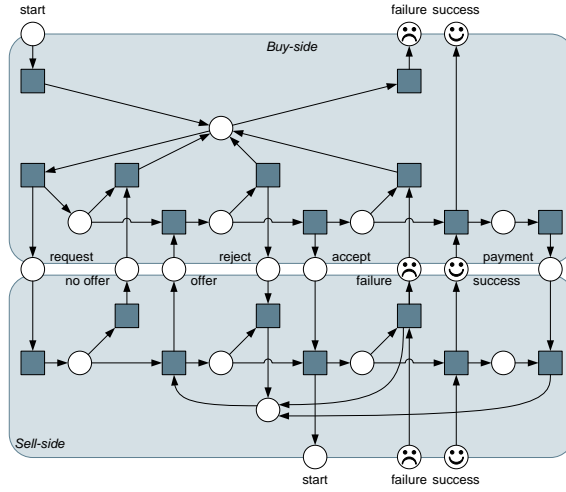


Fig. 5. Checking orchestration soundness

For the orchestration part we use the same conditional soundness. Any orchestration is sound if and only if its tasks are sound. This is straightforward to check using only Fig. 3. So we only have to check the soundness of elementary tasks, which are just transitions. In case the framework is extended by some new orchestration operations, we have to make sure that the orchestration is still sound if and only if the tasks are sound.

## 4 Pricing Model

The probability of success of a task  $t$  is  $s_t$ . The cost  $C_t$  of task  $t$  is a random variable, with expectation  $\mu_t$ , conditional expectation  $\nu_t$ , given the service does not fail and variance  $\sigma_t^2$ . With these four characteristics we will determine a safe price and the risk in case of failure. The price should compensate the loss in case the service fails. So a lower bound for the price is  $\mu_t/s_t$  which gives an expected reward of  $(\mu_t/s_t) \cdot s_t$  which equals to the expected cost. However, the service must have an earning capacity to cover losses, so we choose a price  $\mu_t/s_t + b$  where  $b$  is a parameter to be determined. The expected profit will be  $b \cdot s_t$  which is the difference between the expected reward and the expected cost. In order to determine  $b$ , we have to calculate the risk. There are several ways to define the risk.



**Fig. 6.** Buy-side and sell-side combined

An obvious definition of *risk* is the probability that the real cost exceed the expected reward. So  $\mathbb{P}[C_t \geq \mu_t + b \cdot s_t]$  should be small enough. If we accept a 5% risk we have to choose  $b$  such that

$$\mathbb{P}[C_t \geq \mu_t + b \cdot s_t] = \mathbb{P}\left[\frac{C_t - \mu_t}{\sigma_t} \geq \frac{b \cdot s_t}{\sigma_t}\right] \leq 0.05.$$

Assuming a normal distribution for  $C_t$ , we derive from the standard normal distribution, that  $b \cdot s_t = 1.65$  and so  $b = 1.65 \cdot \sigma_t / s_t$  and therefore the price of the service is  $(\mu_t + 1.65 \cdot \sigma_t) / s_t$ . The assumption of normality is justified by the fact that a service is built up by several independent services and so the central limit theorem [7] justifies a normal approximation. We also have an exact probabilistic bound based on the Chebyshev inequality (see Sect. 2)

$$\mathbb{P}\left[\frac{C_t - \mu_t}{\sigma_t} \geq \frac{b \cdot s_t}{\sigma_t}\right] \leq \mathbb{E}\left[\left(\frac{C_t - \mu_t}{\sigma_t}\right)^2 \geq \left(\frac{b \cdot s_t}{\sigma_t}\right)^2\right] = \frac{\sigma_t^2}{b^2 \cdot s_t^2} = 0.05$$

since  $\mathbb{E}[(C_t - \mu_t) / \sigma_t]^2 = 1$ . This gives  $b = (\sigma_t / s_t) / \sqrt{0.05} = 4.47 \cdot \sigma_t / s_t$  which is almost three times as large as the former bound.

A second way to define the risk is the expected cost, under the condition that the task fails. Since we will calculate  $\nu_t$  this can be expressed as  $\mu_t - \nu_t \cdot s_t$ . Note that this quantity is independent of the price. A third way is to define the risk is  $\mathbb{E}[\max(0, R_t - C_t)]$  where  $R_t$  is a random variable that represents the reward for a task  $t$ , so  $R_t = 0$  in case of a failure and  $R_t = \mu_t / s_t + b$  in case of success. This measure for risk, however, requires knowledge of the complete distribution of  $C_t$ . Therefore we will use the first interpretation of risk.

In the rest of this section we will calculate the four characteristics (success probability, cost expectation, cost variance and conditional cost expectation) in

an inductive way: For an elementary task  $t$  we assume they are given. This is reasonable, because such values can be estimated from log files, for instance. For a compound task we will derive the characteristics for the four orchestration constructs sequence, parallel, choice, and iteration. We use  $X_t$  as the random variable that indicates if the task  $t$  is a success ( $X_t = 1$ ) or a failure ( $X_t = 0$ ). So this random variable has a Bernoulli distribution. Note that  $C_t$  and  $X_t$  are dependent. The reason for this is in the sequence construct: the cost in case of failure might be less than the cost of success. In the rest of the paper we assume that each invocation of a task instance is (stochastically) independent of all other instances, in particular for  $a \neq b$  we have  $X_a$  and  $C_a$  are mutually independent of  $X_b$  and  $C_b$ . For a complete set of the proofs we refer to a technical report [9]. For the convenience of the reviewers, all proofs are listed in an appendix of this article which will be removed in a final version. To illustrate the techniques, we give the proofs for the sequence and the iteration.

**Sequence** Assume that we have a sequence construct  $a \cdot b$  that contains (in the given order) tasks  $a$  and  $b$ . Thus,  $a \cdot b$  prescribes that first  $a$  has to be completed, after which  $b$  can be started. First, we compute the success probability ( $s_{a \cdot b}$ ) for  $a \cdot b$ ; second, we compute its cost expectation ( $\mu_{a \cdot b}$ ); third, we compute the cost variance ( $\sigma_{a \cdot b}^2$ ); and fourth, we compute the conditional cost expectation of  $a \cdot b$  in case of success ( $\nu_{a \cdot b}$ ). In all cases, we assume that these characteristics for  $a$  and  $b$  are known (by induction, if you will).

For the following proofs two properties are important:  $X_{a \cdot b} = X_a \cdot X_b$  and  $C_{a \cdot b} = C_a + X_a \cdot C_b$ .

**Theorem 1 (Success probability of  $a \cdot b$ ).**

$$s_{a \cdot b} = s_a \cdot s_b.$$

*Proof.*

$$\begin{aligned} s_{a \cdot b} &= \mathbb{P}[X_{a \cdot b} = 1] = \mathbb{P}[X_a \cdot X_b = 1] = \mathbb{P}[X_a = 1 \wedge X_b = 1] \\ &= \mathbb{P}[X_a = 1] \cdot \mathbb{P}[X_b = 1] = s_a \cdot s_b \end{aligned}$$

**Theorem 2 (Cost expectation of  $a \cdot b$ ).**

$$\mu_{a \cdot b} = \mu_a + s_a \cdot \mu_b$$

*Proof.*

$$\mu_{a \cdot b} = \mathbb{E}(C_{a \cdot b}) = \mathbb{E}(C_a + X_a \cdot C_b) = \mathbb{E}(C_a) + \mathbb{E}(X_a) \cdot \mathbb{E}(C_b) = \mu_a + s_a \cdot \mu_b$$

For the proof of the next theorem, we need a lemma (for a proof see [9]).

**Lemma 1.**

$$\mathbb{E}(C_{a \cdot b}^2) = \sigma_a^2 + \mu_a^2 + s_a \cdot (\sigma_b^2 + \mu_b^2) + 2 \cdot \mu_b \cdot \nu_a \cdot s_a$$

**Theorem 3 (Cost variance of  $a \cdot b$ ).**

$$\sigma_{a \cdot b}^2 = \sigma_a^2 + s_a \cdot \sigma_b^2 + (s_a - s_a^2) \cdot \mu_b^2 + 2 \cdot (\nu_a - \mu_a) \cdot s_a \cdot \mu_b$$

*Proof.*

$$\begin{aligned}
\sigma_{a \cdot b}^2 &= \mathbb{E}(C_{a \cdot b}^2) - (\mathbb{E}(C_{a \cdot b}))^2 \\
&= \sigma_a^2 + \mu_a^2 + s_a \cdot (\sigma_b^2 + \mu_b^2) + 2 \cdot \mu_b \cdot \nu_a \cdot s_a - (\mu_a + s_a \cdot \mu_b)^2 \\
&= \sigma_a^2 + \mu_a^2 + s_a \cdot (\sigma_b^2 + \mu_b^2) + 2 \cdot \mu_b \cdot \nu_a \cdot s_a - \mu_a^2 - s_a^2 \cdot \mu_b^2 - 2 \cdot \mu_a \cdot s_a \cdot \mu_b \\
&= \sigma_a^2 + s_a \cdot \sigma_b^2 + (s_a - s_a^2) \cdot \mu_b^2 + 2 \cdot (\nu_a - \mu_a) \cdot s_a \cdot \mu_b
\end{aligned}$$

**Theorem 4 (Conditional cost expectation of  $a \cdot b$  if success).**

$$\nu_{a \cdot b} = \nu_a + \nu_b$$

*Proof.*

$$\begin{aligned}
\nu_{a \cdot b} &= \mathbb{E}[C_{a \cdot b} | X_{a \cdot b} = 1] = \mathbb{E}[C_a + X_a \cdot C_b | X_a = X_b = 1] \\
&= \mathbb{E}[C_a | X_a = X_b = 1] + \mathbb{E}[X_a \cdot C_b | X_a = X_b = 1] \\
&= \mathbb{E}[C_a | X_a = 1] + \mathbb{E}[C_b | X_a = X_b = 1] = \mathbb{E}[C_a | X_a = 1] + \mathbb{E}[C_b | X_b = 1] \\
&= \nu_a + \nu_b
\end{aligned}$$

**Parallel composition** Assume that we have a parallel construct  $a||b$  that contains tasks  $a$  and  $b$ . Thus,  $a||b$  prescribes that both  $a$  and  $b$  have to be executed, and that they can be executed in parallel. Like with the sequence construct, the parallel construct  $a||b$  is successful if both  $a$  and  $b$  are successful. The following theorems use the properties:  $X_{a||b} = X_a \cdot X_b$  and  $C_{a||b} = C_a + C_b$ .

**Theorem 5 (Success probability of  $a||b$ ).**

$$s_{a||b} = s_a \cdot s_b.$$

**Theorem 6 (Cost expectation of  $a||b$ ).**

$$\mu_{a||b} = \mu_a + \mu_b$$

**Theorem 7 (Cost variance of  $a||b$ ).**

$$\sigma_{a||b}^2 = \sigma_a^2 + \sigma_b^2$$

**Theorem 8 (Conditional cost expectation of  $a||b$  if success).**

$$\nu_{a||b} = \nu_a + \nu_b$$

**Choice** Assume that we have a choice construct  $a + b$  that contains tasks  $a$  and  $b$ , and that the alternatives ( $a$  and  $b$ ) are chosen with an independent random variable  $A$ , with  $\mathbb{P}[A = 1] = \alpha = 1 - \mathbb{P}[A = 0]$ . If  $A = 1$ , then  $a$  will be chosen, else if  $A = 0$ , then  $b$  will be chosen. The following theorems use:  $X_{a+b} = A \cdot X_a + (1 - A) \cdot X_b$  and  $C_{a+b} = A \cdot C_a + (1 - A) \cdot C_b$ .

**Theorem 9 (Success probability of  $a + b$ ).**

$$s_{a+b} = \alpha \cdot s_a + (1 - \alpha) \cdot s_b.$$

**Theorem 10 (Cost expectation of  $a + b$ ).**

$$\mu_{a+b} = \alpha \cdot \mu_a + (1 - \alpha) \cdot \mu_b$$

**Theorem 11 (Cost variance of  $a + b$ ).**

$$\sigma_{a+b}^2 = \alpha \cdot \sigma_a^2 + (1 - \alpha) \cdot \sigma_b^2 + \alpha \cdot (1 - \alpha) \cdot (\mu_a - \mu_b)^2$$

**Theorem 12 (Conditional cost expectation of  $a + b$  if success).**

$$\nu_{a+b} = \frac{\nu_a \cdot \alpha \cdot s_a + \nu_b \cdot (1 - \alpha) \cdot s_b}{\alpha \cdot s_a + (1 - \alpha) \cdot s_b}$$

**Iteration** Assume that we have an iteration construct  $a^*$ , which contains the task  $a$ , and that the  $i$ -th iteration is chosen with an independent random variable  $Y_i$ , with  $\mathbb{P}[Y_i = 1] = \alpha = 1 - \mathbb{P}[Y_i = 0]$ . To compute iteration characteristics we can simply unfold the iteration once using an alternative, a sequence, and a skip action  $\tau$ , which has the following characteristics:  $s_\tau = 1$ ,  $\mu_\tau = 0$ ,  $\sigma_\tau^2 = 0$ , and  $\nu_\tau = 0$ . Thus,  $a^* = \tau + a \cdot a^*$ , where  $\tau$  has probability  $1 - \alpha$  and  $a \cdot a^*$  has probability  $\alpha$ . In terms of random variables we have the recursive equation:  $P = Y \cdot T + (1 - Y) \cdot A \cdot P$  where  $P$  stands for the process  $a^*$ ,  $T$  stands for task  $\tau$  and  $A$  for task  $a$ . We get the following characteristics:

**Theorem 13 (Success probability of  $a^*$ ).**

$$s_{a^*} = \frac{1-\alpha}{1-\alpha \cdot s_a}.$$

*Proof.*

$$\begin{aligned} s_{a^*} &= s_{\tau+a \cdot a^*} = (1-\alpha) \cdot s_\tau + \alpha \cdot s_{a \cdot a^*} = 1 - \alpha + \alpha \cdot s_{a \cdot a^*} \\ &= 1 - \alpha + \alpha \cdot s_a \cdot s_{a^*} = \frac{1-\alpha}{1-\alpha \cdot s_a} \end{aligned}$$

**Theorem 14 (Cost expectation of  $a^*$ ).**

$$\mu_{a^*} = \frac{\alpha \cdot \mu_a}{1-\alpha \cdot s_a}$$

*Proof.*

$$\begin{aligned} \mu_{a^*} &= \mu_{\tau+a \cdot a^*} = (1-\alpha) \cdot \mu_\tau + \alpha \cdot \mu_{a \cdot a^*} = \alpha \cdot \mu_{a \cdot a^*} = \alpha \cdot (\mu_a + s_a \cdot \mu_{a^*}) \\ &= \frac{\alpha \cdot \mu_a}{1-\alpha \cdot s_a} \end{aligned}$$

**Theorem 15 (Cost variance of  $a^*$ ).**

$$\sigma_{a^*}^2 = \frac{\alpha \cdot (\sigma_a^2 + (s_a - s_a^2) \cdot \mu_{a^*}^2 + 2 \cdot (\nu_a - \mu_a) \cdot s_a \cdot \mu_{a^*}) + (1-\alpha) \cdot \alpha \cdot \mu_{a \cdot a^*}^2}{1-\alpha \cdot s_a}$$

*Proof.*

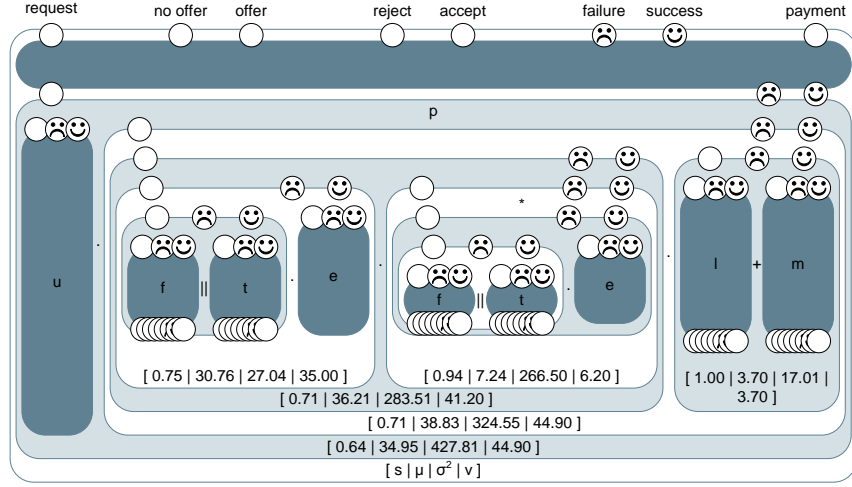
$$\begin{aligned} \sigma_{a^*}^2 &= \sigma_{\tau+a \cdot a^*}^2 = (1-\alpha) \cdot \sigma_\tau^2 + \alpha \cdot \sigma_{a \cdot a^*}^2 + (1-\alpha) \cdot \alpha \cdot (\mu_\tau - \mu_{a \cdot a^*})^2 \\ &= \alpha \cdot \sigma_{a \cdot a^*}^2 + (1-\alpha) \cdot \alpha \cdot \mu_{a \cdot a^*}^2 \\ &= \alpha \cdot (\sigma_a^2 + s_a \cdot \sigma_{a^*}^2 + (s_a - s_a^2) \cdot \mu_{a^*}^2 + 2 \cdot (\nu_a - \mu_a) \cdot s_a \cdot \mu_{a^*}) + \\ &\quad (1-\alpha) \cdot \alpha \cdot \mu_{a \cdot a^*}^2 \\ &= \alpha \cdot s_a \cdot \sigma_{a^*}^2 + \alpha \cdot (\sigma_a^2 + (s_a - s_a^2) \cdot \mu_{a^*}^2 + 2 \cdot (\nu_a - \mu_a) \cdot s_a \cdot \mu_{a^*}) + \\ &\quad (1-\alpha) \cdot \alpha \cdot \mu_{a \cdot a^*}^2 \\ &= \frac{\alpha \cdot (\sigma_a^2 + (s_a - s_a^2) \cdot \mu_{a^*}^2 + 2 \cdot (\nu_a - \mu_a) \cdot s_a \cdot \mu_{a^*}) + (1-\alpha) \cdot \alpha \cdot \mu_{a \cdot a^*}^2}{1-\alpha \cdot s_a} \end{aligned}$$

**Theorem 16 (Conditional cost expectation of  $a^*$  if success).**

$$\nu_{a^*} = \frac{\nu_a \cdot \alpha \cdot s_a}{1-\alpha \cdot s_a}$$

*Proof.*

$$\begin{aligned} \nu_{a^*} &= \nu_{\tau+a \cdot a^*} = \frac{\nu_\tau \cdot (1-\alpha) \cdot s_\tau + \nu_{a \cdot a^*} \cdot \alpha \cdot s_{a \cdot a^*}}{(1-\alpha) \cdot s_\tau + \alpha \cdot s_{a \cdot a^*}} = \frac{\nu_{a \cdot a^*} \cdot \alpha \cdot s_{a \cdot a^*}}{(1-\alpha) + \alpha \cdot s_{a \cdot a^*}} \\ &= \frac{(\nu_a + \nu_{a^*}) \cdot \alpha \cdot s_{a \cdot a^*}}{(1-\alpha) + \alpha \cdot s_{a \cdot a^*}} = \frac{\nu_a \cdot \alpha \cdot s_{a \cdot a^*} + \nu_{a^*} \cdot \alpha \cdot s_{a \cdot a^*}}{(1-\alpha) + \alpha \cdot s_{a \cdot a^*}} = \frac{\nu_a \cdot \alpha \cdot s_{a \cdot a^*} + \nu_{a^*} \cdot \alpha \cdot s_{a \cdot a^*}}{(1-\alpha) + \alpha \cdot s_{a \cdot a^*}} \\ &= \frac{\nu_a \cdot \alpha \cdot s_{a \cdot a^*}}{(1-\alpha) + \alpha \cdot s_{a \cdot a^*}} + \frac{\nu_{a^*} \cdot \alpha \cdot s_{a \cdot a^*}}{(1-\alpha) + \alpha \cdot s_{a \cdot a^*}} = \frac{\nu_a \cdot \alpha \cdot s_{a \cdot a^*}}{(1-\alpha) + \alpha \cdot s_{a \cdot a^*}} + \nu_{a^*} \cdot \frac{\alpha \cdot s_{a \cdot a^*}}{(1-\alpha) + \alpha \cdot s_{a \cdot a^*}} \\ &= \frac{\nu_a \cdot \alpha \cdot s_{a \cdot a^*}}{(1-\alpha) + \alpha \cdot s_{a \cdot a^*}} / \left(1 - \frac{\alpha \cdot s_{a \cdot a^*}}{(1-\alpha) + \alpha \cdot s_{a \cdot a^*}}\right) = \frac{\nu_a \cdot \alpha \cdot s_{a \cdot a^*}}{1-\alpha} = \frac{\nu_a \cdot \alpha \cdot s_a \cdot \frac{1-\alpha}{1-\alpha \cdot s_a}}{1-\alpha} \\ &= \frac{\nu_a \cdot \alpha \cdot s_a}{1-\alpha \cdot s_a} \end{aligned}$$



**Fig. 7.** The characteristics of the running service

Applying our proposed cost model for the example process  $p$  we can (incrementally) compute the following values for the four characteristics (see Fig. 7):

- $s_p = 0.64$ ; i.e. the service succeeds in about 64% of the cases.
- $\mu_p = 34.95$ , i.e. the provider has expected costs of about €34.95.
- $\sigma_p^2 = 427.81$  is the cost variance
- $\nu_p = 44.90$  are the conditional cost in case of success

Hence  $\sigma_p/s_p = 32.31$  and  $\mu_p/s_p = 54.60$ . We assume the provider wants to accept a risk of 5%. If we apply the normal approximation, we obtain  $b = 53.31$  and so the price of the service  $\mu_p/s_p + b$  should be €107.91. The expected profit  $b \cdot s_p$  equals €34.11. If we apply the exact Chebyshev inequality we obtain  $b = 142.16$  and then the price becomes €199.02. Then the expected profit equals €90.98 which is obviously larger. In both cases is the expected cost, given the service fails,  $\mu_p - \nu_p \cdot s_p$ , equals €6.21. It is not difficult to repeat this exercise for other choices of the risk.

## 5 Related work

In our previous work [10], we have developed an SOA-based architecture framework which is similar to the service component architecture (SCA) [11]. In this article, we extended this work by a web service framework which allows to check the soundness property compositionally. The goals of this part of our work are close to the ones of Milanovic [12], where the author seeks for correctness proofs for compositions of web services. The framework there is based of abstract state

machines, the same four basic operations are considered (the iteration is however left out of consideration in the correctness part), and proof obligations are generated to show the correctness of the composition.

In [13], Diaz et al. represent a translation from WS-CDL to timed automata and from timed automata to WS-BPEL to generate web services. The correctness is checked by model checking properties of the obtained timed automata.

Furthermore, we have presented a method to compute the cost of a web service. There is a long list of publications dealing with cost of services. However, to the best of our knowledge none of them takes the risk (i.e. the variance of cost) into consideration.

Magnani and Montesi [14] present rules to calculate the cost of BPMN models. These rules cover operations like sequence, parallel, choice, and loop. However, there are no rules given for calculating the probability of success of these operations.

Cardoso et al. [15] present a QoS model for time, reliability, and cost of workflows. Each task has a QoS attribute. Based on these attributes, the cost of the overall workflow can be computed using the METEOR workflow system. In [16], Zeng et al. present a framework for QoS-aware service selection. Price is one of the nonfunctional properties which are taken into account. Paoli et al. [17] address the problem of designing a composed system that has to guarantee certain quality criteria such as security, completion time, and also cost. It is shown how these criteria can be computed on the structure of a service. To this end, quality evaluation rules (similar those in [15]) for sequence, parallel, switch, and loop are proposed. To summarize, all three approaches cover a broader spectrum of QoS criteria than cost. However, probabilities for successful termination and for the price calculation of activities and services are not considered.

In [18], Brocke and Lindner present a general framework for the evaluation of the financial consequences of outsourcing, while in [19] Günter et al. investigate pricing mechanisms appropriate for web services. This is, however, far beyond the scope of this paper. In [20], Ding proposes a method of value-based pricing where the price is a function reflecting the expected value of the product. This method can be used, for example, to derive the price of each elementary task.

## 6 Conclusion

In this paper, we have presented a web services framework to design service orchestrations that are sound by design. To this end, service interfaces and routing constructs are designed in such a way that two arbitrary services always interact properly.

In order to provide a certain functionality, web services often have to buy some functionality from other web services. In such a setting calculating cost of a web service is an important issue, because a web service can only survive on the market if it is profitable. To this end, we have provided an approach to compute the expected cost of a sound service orchestration (i.e. web service). Since we consider no-cure-no-pay services, that is, a customer only pays for a delivered

service, the proposed approach takes probabilities for successful execution of a web service into account. We also consider the risk involved by calculating the cost variance. The approach is compositional in the sense that the cost of the whole web service is computed from the cost of all tasks. Therefore we developed rules to compute the cost, the probability of success, and the cost variance for the constructs sequence, parallel, choice, and iteration.

So far the proposed approach is subject to some restrictions. With sequence, parallel, choice, and iteration we only consider a restricted set of operations to construct compound tasks. Concepts such as cancelation and compensation of tasks are also not considered. Furthermore, it might be interesting to see to what extent processes other than no-cure-no-pay processes can be supported in a similar way with our approach.

In ongoing research we plan to extend our web service framework with more types of choreography protocols, in particular protocols that will ease cancelation of (parts of) services and allow for compensation mechanisms. An other extension we consider is to relax the assumption that all tasks in an iteration are independent of each other. This is more realistic, however, it requires more knowledge (data) about the performance of tasks.

## References

1. Micali, S., Rivest, R.L.: Micropayments revisited. In Preneel, B., ed.: Topics in Cryptology - CT-RSA 2002, The Cryptographer's Track at the RSA Conference, 2002, San Jose, CA, USA, February 18-22, 2002, Proceedings. Volume 2271 of Lecture Notes in Computer Science., Springer (2002) 149–163
2. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE **77**(4) (1989) 541–580
3. Reisig, W.: Petri Nets: An Introduction. Volume 4 of EATCS Monographs on Theoretical Computer Science. Springer, Berlin, Germany (1985)
4. Aalst, W.M.P.v.d.: The application of Petri nets to workflow management. The Journal of Circuits, Systems and Computers **8**(1) (1998) 21–66
5. Verbeek, H.M.W., Basten, T., Aalst, W.M.P.v.d.: Diagnosing workflow processes using Woflan. The Computer Journal **44**(4) (2001) 246–279
6. Verbeek, H.M.W., Aalst, W.M.P.v.d.: Woflan 2.0: A Petri-net-based workflow diagnosis tool. In Nielsen, M., Simpson, D., eds.: Application and Theory of Petri Nets 2000. Volume 1825 of Lecture Notes in Computer Science., Springer, Berlin, Germany (2000) 475–484
7. Ross, S.: Introduction to probability models. Academic Press (2007)
8. Massuthe, P., Reisig, W., Schmidt, K.: An Operating Guideline Approach to the SOA. Annals of Mathematics, Computing & Teleinformatics **1**(3) (2005) 35–43
9. Hee, K.M.v., Verbeek, H.M.W., Stahl, C., Sidorova, N.: A Framework for Linking and Pricing No-Cure-No-Pay Services. Computer Science Report 08/19, Technische Universiteit Eindhoven, The Netherlands (2008)
10. Aalst, W.M.P.v.d., Beisiegel, M., Hee, K.M.v., König, D., Stahl, C.: An SOA-based architecture framework. International Journal of Business Process Integration and Management (IJBPIIM) **2**(2) (2007) 91–101
11. Beisiegel et al., M.: Service Component Architecture – Assembly Model Specification. SCA Version 1.00, March 15 2007, IBM, SAP et al. (2007)

12. Milanovic, N.: Contract-based web service composition framework with correctness guarantees. In Malek, M., Nett, E., Suri, N., eds.: ISAS. Volume 3694 of Lecture Notes in Computer Science., Springer (2005) 52–67
13. Díaz, G., Cambronero, M.E., Pardo, J.J., Valero, V., Cuartero, F.: Automatic generation of correct web services choreographies and orchestrations with model checking techniques. In: AICT/ICIW. (2006)
14. Magnani, M., Montesi, D.: BPMN: How Much Does It Cost? An Incremental Approach. In Alonso, G., Dadam, P., Rosemann, M., eds.: Business Process Management, 5th International Conference, BPM 2007, Brisbane, Australia, September 24–28, 2007, Proceedings. Volume 4714 of Lecture Notes in Computer Science., Springer (2007) 80–87
15. Cardoso, J., Sheth, A.P., Miller, J.A., Arnold, J., Kochut, K.: Quality of service for workflows and web service processes. *J. Web Sem.* **1**(3) (2004) 281–308
16. Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., Chang, H.: Qos-aware middleware for web services composition. *IEEE Trans. Software Eng.* **30**(5) (2004) 311–327
17. Paoli, F.D., Lulli, G., Maurino, A.: Design of quality-based composite web services. In Dan, A., Lamersdorf, W., eds.: ICSOC 2006. Volume 4294 of Lecture Notes in Computer Science., Springer (2006) 153–164
18. Brocke, J.v., Lindner, M.A.: Service portfolio measurement: a framework for evaluating the financial consequences of out-tasking decisions. In Aiello, M., Aoyama, M., Curbera, F., Papazoglou, M.P., eds.: ICSOC 2004, ACM (2004) 203–211
19. Günther, O., Tamm, G., Leymann, F.: Pricing web services. *International Journal of Business Process Integration and Management (IJBPIIM)* **2**(2) (2007) 132–140
20. Ding, W.: Services pricing through business value modeling and analysis. In: 2007 IEEE International Conference on Services Computing (SCC 2007), 9–13 July 2007, Salt Lake City, Utah, USA, IEEE Computer Society (2007) 380–386