Analysis of Revenue Improvements with Runtime Adaptation of Service Composition Based on Conditional Request Retries

Miroslav Živković[1] and Hans van den Berg[1,2]

[1] TNO, Brassersplein 2,
2612 CT Delft, The Netherlands
miroslav.zivkovic@tno.nl
[2] University of Twente, PO Box 217,
7500 AE Enschede, The Netherlands

Abstract. In this paper we consider the runtime service adaptation mechanism for service compositions that is based on conditional retries. A single retry may be issued while a concrete service within composition is executed. This retry could either invoke the same concrete service or a functionally equivalent service implementing the same task. We determine the optimal moments to terminate the current request and replicate it. The calculation of these moments for each task within the workflow is based on different QoS parameters from Service Level Agreements, like services' response–time distributions and cost–relating parameters. The calculations are performed taking into account the remaining actual time–to–deadline, and the benefit of conditional retry mechanism is illustrated by simulations. We further discuss the impact of costs and response–time distributions' parameters to the solution at hand.

Keywords: Service Oriented Architecture, Optimal Retry Policies, Watchdog Timer, Hazard Rate.

1 Introduction

Composite web services in a service oriented architecture (SOA) aggregate web services that may be deployed and executed within different administrative domains. In the orchestrated scenario composite web service provider acts as an orchestrator that invokes the aggregated services according to a pre–defined workflow. The workflow is based on an unambiguous functionality description of a service ("abstract service"), and several alternatives ("concrete services") may exist that match such a description [1]. With respect to functionality, all concrete services that match the same abstract service are identical.

For commercial success of the composite web service, it is important that the service provider is able to offer the service at attractive price-quality ratios. To this end, the composite service provider (CSP) negotiates Service Level Agreements (SLAs) with the client and third party domains. A service level agreement

F. De Paoli, E. Pimentel, and G. Zavattaro (Eds.): ESOCC 2012, LNCS 7592, pp. 169–183, 2012.

(SLA) is a legal contract that specifies the minimum expectations and obligations that exist between a service provider and a service consumer [2]. Due to the high variability of the service environment, the SLA violations could occur relatively often, leading to providers' losses and customer dissatisfaction.

One of the possible approaches to mitigate the problem of SLA violations is to optimize the running composition instances by adaptation of the composition itself at runtime. In general, the adaptations could be done by means of service rebinding/substitution, or via structural adaptation of the composition [3], [11]. When adaptation is done by service substitution, a service within the composition is exchanged by another one, where, in ideal case both services are functionally equivalent. On the other hand, an interesting possibility that may be applicable in order to satisfy the agreed SLA is to trigger the retry action hoping that the fault was transient [4]. The two basic issues that need to be addressed for any of the mentioned approaches are (1) *when* to perform the adaptation, and (2) how much does it *cost* (time, money, etc.)?

We analyse the runtime adaptation of the orchestrated service composition that is based on *conditional* retry mechanisms. For each task within the service workflow a concrete service has been selected based on some end–to–end optimality criteria, i.e. the service composition has been determined. Services that are not selected are "placed" in the pool of the functionally equivalent services. The concrete services' SLAs contain response–time probability density function, as well as the invocation costs, while the end–to–end SLA contains end–to–end deadline that CSP promises to its clients, as well as reward/penalty (for CSP) when the promised hard deadline is met/missed. We illustrate our scheme in Figure 1. When task i is executed by the concrete service that implements it ($\mathrm{CS}_i(1)$), the orchestrator starts the "watchdog timer" with the timeout count value that is set to θ_i for the execution of the selected service. When the counter expires and there is no response generated from the invoked service, the orchestrator terminates the original request, and initiates a new service invocation for the same task (i.e. makes a retry). The new invocation could be submitted, e.g. to the same concrete service as illustrated in Figure 1 for task 2, when timeout counter value θ_2 becomes zero. This may be the case when there is a single implementation of a given workflow task. In case there is more than one implementation of a given task i, the new invocation (retry) could be submitted to another concrete service (e.g. $\mathrm{CS}_i(2)$). In the latter case dynamic binding may be required, and once the response from alternative is generated, the execution proceeds with the execution of the next service from the initial composition. When the response from the concrete service is generated before the timeout expires, the orchestrator executes the next task within the workflow. The counter of the timer is set to the new value, e.g. θ_{i+1}, and so on, till all tasks within the workflow have been executed. Based on the fact whether the end–to–end deadline is then met or not, the CSP is rewarded or penalised.

In this paper we analyse the proposed conditional request retry mechanism when a *single* retry is made. This single retry for the executed service is made when, based on service's response–time distribution, it becomes "clear" that

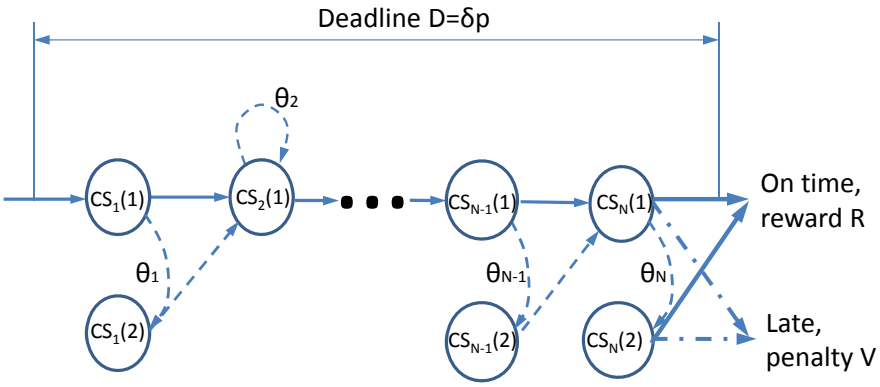

Fig. 1. Runtime service adaptation with conditional retries

the guarantees presented within SLA are jeopardized. In general case, the much faster and more expensive alternative is then executed, which makes it possible for CSP to claim the reward from it's clients. We analyse how the (optimal) values of timeout counter values θ_i could be determined, i.e. the procedure to calculate the time instances when retry should be attempted. We illustrate the impact of response–time distributions and invocation costs specified by services' SLAs to the solution at hand. We indicate which distributions may be considered when retry mechanism is to be applied, and the potential revenue improvements of our scheme. For a given example we determine the optimal position of the retry, i.e. we give an answer to the question whether it would be better to perform the retry the sooner or later during the workflow execution.

The paper is organized as follows: in the next Section we give details of the related work. In Section 3 we describe the system model and the assumptions taken. In Section 4 we explain how to determine the optimal timer values. Based on this analysis, we describe the simulation results for a couple of scenarios in Section 5 and conclude the paper with possible directions for the future work in Section 6.

2 Related Work

QoS–aware service composition within SOA is usually static process, i.e. it deals with determining the "best" available service for the abstract composition during the deployment, e.g. by maximizing some utility function [14] or by combining the local selection and global optimization [15]. The methods an approaches deal with the optimization in a static manner, i.e. the optimal compostion does not change at runtime. More recent work in this area focuses on dynamic, runtime composition solutions and adaptations [9, 10, 12]. For each task invocation, the orchestrator dynamically binds the task of the abstract composition to an actual implementation (i.e., concrete service), selecting it from the pool of service

providers that offer it. Due to the dynamic service composition it may happen that every composite service request is served by different composition. The service selection is driven by the solution of a suitable optimization problem, which is reduced to the linear optimization problem [9], or the optimization is based on evolutionary computation [10] or is based on the principles of dynamic programming [12]. However, none of [9, 10, 12] consider the possible applicability of retry mechanisms, i.e. the possibility of service adaptation *while* actual task is executed.

The retry mechanisms as self–healing solution for temporarily unavailable services, have been identified and classified, among others, in [3,4]. The performance of the retry mechanisms has been analysed in detail by van Moorsel, Wolter, et. al. in [5–7]. Their work has focused on optimal retry mechanisms for a single service in order to *minimize the completion time*. The number of retries could either be finite or infinite, and the completion time when restarting must be less than without restarting. Okamura et. al. in [8] analyse the optimal restart policies when deadline is given. First, they prove that, time–fixed restart time is the best policy even in non–stationary control setting under the assumption of unbounded restart opportunities. They also analyse the problem of optimal restart when a deadline is given and develop on–line adaptive algorithms for estimating the optimal restart time interval via reinforcement learning. The solutions mentioned focus on minimization of completion time. None of these solutions analyse the problem using the penalty or reward of any kind. The cost of the retries are defined as additional time to re–issue the service request. Besides, the retry mechanism is analysed from the single service point of view.

On the other hand, Yousefi et. al. in [13] describe a strategy for QoS aware service selection which takes advantage of the existing variability in QoS data to provide higher quality services with less cost compared to the conventional QoS aware service selection methods. In their method, *each request* is replicated over multiple independent services to achieve the required QoS. This strategy is clearly sub–optimal as it implies un–necessary request replications (and therefore higher costs) for all those requests that meet the required QoS without request replication. Our approach optimizes request replication from the point of increasing the profit of composite service provider. Therefore, we aim to issue request replication only when it is really meaningful.

3 Considered System Model

In this section we describe the model of the system that we will use for further analysis. We furhtermore adopt some assumptions for the considered system for the model illustrated in Figure 1.

The assumptions and the main features of our model are:

– We observe the sequential workflow that consists of N tasks to be executed by the orchestrator. How to aggregate some of frequently used workflow patterns and transform the workflow into the sequential one is illustrated, e.g. in [12].

- The selection of candiate services for each task $i,\ i = 1, \ldots N$ has been performed, and there are at most $M_i = 2$ alternative (concrete) services to be considered, denoted by $\mathrm{CS}_i(j),\ j = 1, 2$. We call the initial service composition the static service composition (SSC).
- We adopt the convention that $\mathrm{CS}_i(1)$ is the service selected for static service composition.
- A watchdog timer with timeout value θ_i is associated to workflow task i. Once the timeout expires, and there is no response from the selected service, a retry attempt is made.
- There is only one retry attempt. When request replication is made, the timer is not used till response is obtained.
- When the response is obtained by the orchestrator before θ_i expires, the next task $(i + 1)$ in the workflow is executed, by service $\mathrm{CS}_{i+1}(1)$
- In case timer θ_i expires without response generated, the orchestrator invokes the functionally equivalent alternative service $\mathrm{CS}_i(2)$ (conditional request replication). In case there is only one service implementing the particular task, the orchestrator attempts a single retry using the same concrete service (i.e. $\mathrm{CS}_i(1)$).

In model illustrated at Figure 1 we see the second task is implemented by only one service, and therefore the retry takes place by this service. It is naturally possible this service is temporarily unavailable, or unavailable for a longer period of time. In the latter case multiple retries or some other mechanisms may be applicable, but we do not consider such problem in this paper.

Each concrete service $\mathrm{CS}_i(j),\ i = 1, \ldots, N,\ j = 1, 2$ has a response time represented by the random variable $D_{i,j} \geq 0$. We model the response-time of each concrete service as a black box, which means that $D_{i,j}$ is a random variable for which respective cummulative distribution function (CDF), or equivalent probability density function (PDF) is given. The CDFs and PDFs for concrete services are denoted by $F_{i,j}$ and $f_{i,j}$, respectively. For each concrete service $\mathrm{CS}_i(j),\ i = 1, \ldots, N,\ j = 1, \ldots, M_i$, there is an SLA agreed between the individual service provider (ISP) of that service and the composite service provider (CSP). This SLA contains the following elements:

- The response–time cummulative distribution function, $F_{i,j}$.
- The execution cost $c_{i,j}$ [money unit] per single invocation. From the ISP viewpoint, this value represents reward.

The composite service provider agrees the following SLA with its clients:

- The end-to-end response time penalty threshold δ_p [time unit].
- The fraction of response time realisations p_{e2e} that should be within the deadline δ_p.
- The reward R [money unit] that the CSP gets for executing a single request within penalty deadline δ_p.
- The penalty V [money unit] that the CSP pays to the end customer when the agreed end-to-end deadline is not met.

We therefore adopt a constant penalty function for the composite provider, i.e. a constant payment needs to be made if a given end–to–end response time threshold value is surpassed.

We assume that response times of concrete services are mutually independent, as the services are usually deployed by different service providers. Under this assumption of independence, the end–to–end response time distribution can be determined by taking the convolution of the respective concrete service distributions. Besides, the end–to–end response time distribution of the composite service is therefore calculated as

$$F_{\text{e2e}} = F_{1,1} \star F_{2,1} \star \cdots \star F_{N,1},$$

where $\star$ operator represents convolution. *For examples how to calculate the end–to–end response time distribution of some other frequently used workflow design patterns, see* [12].

In case of SSC, the execution costs for the composite service provider are defined as

$$C_{\text{e2e}} = c_{1,1} + c_{2,1} + \cdots + c_{N,1},$$

where $c_{i,1},\ i = 1, \ldots, N$ is the execution cost per individual composite service $\text{CS}_i(1),\ i = 1, \ldots, N$. We take here that $\text{CS}_i(1)$ is the service selected during service composition, as already explained.

In case that there is no conditional request replication, the party that owns the orchestrator, i.e. composite service provider has to perform the simple cost analysis for the given end–to–end deadline δ_p, parameters R, V and C_{e2e}. Representing the end–to–end reponse time by random variable D_{e2e}, whose response time distribution is F_{e2e}, the probability for a successful response within δ_p is defined by $p_{\text{e2e}} = \mathbb{P}\{D_{\text{e2e}} \leq \delta_p\} = F_{\text{e2e}}(\delta_p)$. The expected revenue per request for composite service provider in case of SSC could therefore be calculated as

$$\begin{aligned}\mathbb{E}[R_{\text{e2e}}] &= -C_{\text{e2e}} + p_{\text{e2e}} \cdot R - (1 - p_{\text{e2e}}) \cdot V = \\ &= -C_{\text{e2e}} - V + p_{\text{e2e}} \cdot (R + V).\end{aligned}$$

Our goal is to apply the runtime adaptation, i.e. dynamic service composition (DSC) by means of conditional request replication in order to increase the revenue of the composite service provider, CSP. In order to do that, we need to identify *the optimal values* $\theta_i^*,\ i = 1, \ldots, N$ of the timer(s) associated with the execution workflow. The optimality is represented as the profit merit for the composite service provider (CSP).

4 Analysis of the Retry Mechanism

Based on the model description given in Section 3, in this section we will perform analysis of our solution, i.e. the conditional request replication mechanism. We will first illustrate for which response–time distributions the considered mechanism could be considered. Then we perform the analysis of the request replication for the last task in the workflow, and subsequently, we analyse the request replication for other tasks in the workflow. We define the formulae that could be used to find the optimal timeout values.

4.1 Response–Time Distribution

As illustrated in [5–7] when θ_i is restart time, and random variables D and D_θ represent response times without and with retries, the retries could be considered only when expected response time with retry $\mathbb{E}[D_\theta]$ is smaller than response time without retries $\mathbb{E}[D]$, which is defined as

$$\mathbb{E}[D] < \mathbb{E}[D - \theta | D > \theta].$$

Based on this condition, it may be concluded that services with heavy–tailed response–time distributions could be considered for retries. The reason for this is that heavy–tailed distributions have considerable probability mass for relatively high values of response–times. The good indicator of the distributions' suitability for retries is hazard rate. If T is an absolutely continuous non-negative random variable (r.v.), its hazard rate function $h(t),\ t \geq 0$, is defined by

$$h(t) = \frac{f(t)}{1 - F(t)} = \frac{f(t)}{\bar{F}(t)},$$

where $f(t)$ is probability density function (PDF) of r.v. T, $F(t)$ is cummulative distribution function (CDF) of T, and $\bar{F}(t)$ is the so called survival function of r.v. T. For a *single service, and no costs involved*, under assumptions that

- the restart of a task terminates the previous attempt
- the successive trials are independent

hazard rate is indicative whether retry may be benefitial. The retries are benefitial for services with decreasing hazard rate; it does no harm to retry services with constant hazard rate, and retries should not be done for services with increasing hazard rate.

Therefore, the recommendations for the services with respect to response–time distributions are:

- Services with heavy–tailed response–time distributions could be used for request replication.
- When task is implemented by a single service that has no decreasing hazard rate, whether the request replication is benefitial should be determined taking into account the costs of execution and expected reward/penalty in such a case.

Another property that we consider for response–time distributions is so called, bimodal, or, in general case, multi–modal distribution. A bimodal distribution is a continuous probability distribution with two different modes, [16]. These appear as distinct peaks (local maxima) in the probability density function, as shown in Figure 2. It appears that number of services deployed today may have multi–modal or bimodal reponse–time distribution, see [17]. The example distribution at given figure indicates that majority of responses are generated within $\delta = 14$ seconds, and the probability this happens is 80%. When choice is to

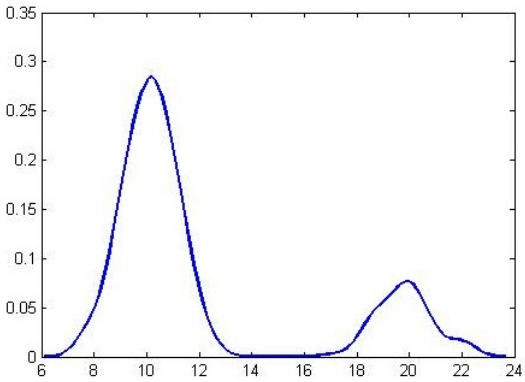

Fig. 2. A typical bimodal response–time distribution, with 80% of the values smaller than 14 seconds

be made between cheap alternative that has bimodal response–time distribution and very expensive service which indicates that response is generated within 5 seconds with, e.g. 95% probability at much higher execution cost, it seems to us reasonable to adopt the following strategy:

1. Use cheaper bimodal (or heavy–tailed) service as the first choice during service composition
2. Set the timeout value to the value that is related to the first maximum (i.e. slightly higher)
3. When the timeout value expires, terminate the current request, and then execute the very expensive alternative.

In case when there is a single implementation of the workflow task, the strategy may be:

1. Calculate the hazard rate of the response–time distribution
2. In case when response–time distribution is with decreasing hazard rate, calulate optimal moments for retries, and set the timeout value to one of the calculated thresholds.
3. In case when response–time distribution is with non–decreasing hazard rate, do not perform the retry mechanism.

In what follows, we will consider the case of expensive services with response–time modelled as lognormal distribution. The support of this distribution are non–negative real values, which overcomes well with the fact that the response–time cannot be negative. Also, the choice of parameters μ and σ of this distribution allow to easily model different response–time distributions.

The PDF $f(t)$ of the lognormal distribution is defined as

$$f(t) = \frac{1}{t\sqrt{2\pi\sigma^2}} \cdot e^{-\frac{\ln t - \mu}{2\sigma^2}},\ t \geq 0.$$

where μ and σ are so called location parameter and the scale parameter, respectively.

4.2 The Last Task Analysis

Let us suppose that for the example sequential workflow with N tasks, the first $N-1$ tasks have been executed with the elapsed time τ, which means the time–to–deadline for the last task is $d_n = \delta_p - \tau$. In order to simplify the notation, let us write $c_{N,i} = c_i,\ f_{N,i} = f_i,\ F_{N,i} = F_i,\ i = 1, 2$. Further, let us denote the execution costs of the tasks already executed by C_E (see Figure 3). The expected reward $\mathbb{E}_1$ in case that there is no replication mechanism (SSC) is

$$\begin{aligned}\mathbb{E}_1 &= -c_1 + R \cdot F_1(d_n) - V \cdot (1 - F_1(d_n)) - C_E \\ &= -C_E - c_1 - V + (R + V) \cdot F_1(d_n).\end{aligned}$$

The expected reward consists of costs incurred for the tasks executed (C_E), the cost of the last task execution (c_1), the reward R that is obtained when the task is executed within given deadline d_n with probability F_n, and the penalty V that is paid when deadline d_n is not met, with probability $1 - F_n$. Naturally, when $d_n <= 0$ we have that $F_n = 0$ – in other words, there is "no chance" the deadline would be met. When our approach is applied the expected reward denoted by $\mathbb{E}_{1\to2}$ is

$$\begin{aligned}\mathbb{E}_{1\to2} = -c_1 + R{\cdot}F_1(\theta_n) - C_E + (1 - F_1(\theta_n)){\cdot} \\ {\cdot}\{-c_2 + R{\cdot}F_2(d_n - \theta_n) - V \cdot (1 - F_2(d_n - \theta_n))\}.\end{aligned}$$

In this case we see that the reward is obtained either when the first service completes the execution before timeout value θ_n expires, which happens with probability $F_1(\theta_n)$. With probability $1 - F_1(\theta_n)$ we make a conditional retry. When the retry is made, the deadline for the second service is $d_n - \theta_n$ and this deadline is met with probability $F_2(d_n - \theta_n)$, which means that, when timeout expires after θ_n and retry is made, CSP obtains reward R with probability $(1 - F_1(\theta_n)) \cdot F_2(d_n - \theta_n)$. The similar reasoning could be made for the case when penalty is to be paid by CSP.

In order for our method to be applicable, there exists at least one θ such that $\mathbb{E}_1(\theta) \leq \mathbb{E}_{1\to2}(\theta),\ 0 < \theta < d_n$. The optimal value $\theta_n = \theta_n^*$ is the one for which $\mathbb{E}_{1\to2}$ reaches maximum at interval $(0, d_n)$. The value $\theta = \theta_n^*$ for which $\mathbb{E}_{1\to2}$ reaches maximum is determined by solving

$$\frac{\partial \mathbb{E}_{1\to2}}{\partial \theta}|_{\theta=\theta_n^*} = 0.$$

Elementary transformations give the following expression

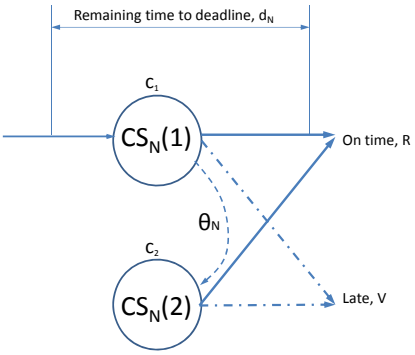

Fig. 3. The execution of the last task with the conditional request replication. Remaining time to deadline is d_n, and the timout value of the timer is θ_n.

$$\frac{f_1(\theta_n^*)}{1-F_1(\theta_n^*)}+\frac{c_2}{R+V}\cdot\frac{f_1(\theta_n^*)}{1-F_1(\theta_n^*)}\cdot\frac{1}{1-F_2(d_n-\theta_n^*)}=$$
$$=\frac{f_2(d_n-\theta_n^*)}{1-F_2(d_n-\theta_n^*)}.$$

which could also be represented as

$$\frac{f_1(\theta_n^*)}{1-F_1(\theta_n^*)}\cdot\left\{1+\frac{c_2}{R+V}\cdot\frac{1}{1-F_2(d_n-\theta_n^*)}\right\}=$$
$$=\frac{f_2(d_n-\theta_n^*)}{1-F_2(d_n-\theta_n^*)},$$

or, equivalently

$$h_1(\theta_n^*)\cdot\left\{1+\frac{c_2}{R+V}\cdot\frac{1}{1-F_2(d_n-\theta_n^*)}\right\}=h_2(d_n-\theta_n^*),$$

where h_1 and h_2 are hazard–rate functions, represented by

$$h_1(t)=\frac{f_1(t)}{1-F_1(t)}$$
$$h_2(t)=\frac{f_2(t)}{1-F_2(t)}.$$

We see that, other than results from [5–8] cost structure plays important role in determining the optimal timout value θ_n^*. Besides the optimal value does not depend from the costs of the first attempt (c_1 in above example). It is trivial to determine the θ_n^* when the same service ($\mathrm{CS}_N(1)$) is considered for the reattempt.

4.3 Analysis of other Tasks in the Workflow

We turn our attention to other tasks in the workflow now. Due to the lack of the space, we would consider the case when there is a single retry within the workflow possible, and would like to determine whether it would be best to apply the given retry scheme either for a) the first task in the workflow, b) the last task N in the workflow or c) the task i in the workflow where $i \neq 1, i \neq N$.

In order to do the fair analysis, we would consider that all services $\mathrm{CS}_i(1)$ have the same execution cost c_1. The response–time distributions of the first service in case a), the last service in case b) and service i in case c) are identical and represented by the same bimodal distribution. In all three cases the remaining $N-1$ response–time distributions are identical (not necessarily bimodal). We want to determine the optimal position (from the revenue point of view) for the alternative service that has execution cost $c_2 > c_1$, and which response–time distribution is lognormal.

Let us analyse the case when retry is considered for the first task in the workflow. Since all response time distributions are known, it is easy to calculate the convolution distribution for the tasks $2-N$. This means that we have the following cases:

- A: The response from the first service is generated before the retry timeout value θ_1, and end–to–end deadline δ_p is met. The execution costs are $-\sum_{i=1}^{N} c_{i,1} = -N \cdot c_1$, and reward is R.
- B: The response from the first service is generated before the retry timeout value θ_1, and end–to–end deadline δ_p is not met. The execution costs are $-\sum_{i=1}^{N} c_{i,1} = -N \cdot c_1$, and penalty V is incurred.
- C: The response from the first service is not generated before the retry timeout value θ_1, so alternative service is invoked. The end–to–end deadline δ_p is met. The execution costs are $-c_2 - \sum_{i=1}^{N} c_{i,1} = -c_2 - N \cdot c_1$, and reward is obtained.
- D: The response from the first service is not generated before the retry timeout value θ_1, , so alternative service is invoked. The end–to–end deadline δ_p is not met. The execution costs are $-c_2 - \sum_{i=1}^{N} c_{i,1} = -c_2 - N \cdot c_1$, and penalty V is paid.

Similar analysis could be performed when the retry is applied at the last workflow task, or when retry is considered for workflow task i, $i \neq 1$, $i \neq N$. The detailed analysis will be omitted here, but, it is no surprise that the biggest benefit is *when retry mechanism is applied for the last workflow task*. This may be explained by the following reasoning: when executing the first task, it is possible to wait a little bit longer before the response is obtained, as the second task, with smaller time–to–deadline is more critical. Therefore it is better to replicate request for the latter task(s) then former. When request is replicated for

former task(s) the execution costs increase while the remaining time to deadline decreases significantly. In other words, any longer response times for the first task may be accounted with by the latter task(s). This holds in general for the problem at hand, as any outliers for first task(s) in the workflow may be of limited impact to the final outcome.

5 Experiments

Due to the limited space, we will show here just the very basic experimental results. These apply to the last task in the workflow, and as explained in Section 4 we consider cheap service with bimodal response–time distribution as the one selected during the initial service composition. The alternative is expensive service with *statistically* "superior" lognormal response–time distributions. The bimodal distribution is illustrated in Figure 2 and the two modes have mean values of 10 and 20, respectivelly. The mixture coefficient is 80%, which means that 80% values of response time have the mean of 10, while the remaining 20% values of response time belong to the mode with the mean value of 20. The mean value of lognormal distribution has been set to 0.25, while the variance of this distribution has been set to 4.

For the given deadline δ_p, and the last task in the workflow, the initial selection is cheap service. When timeout θ expires, the retry is made and expensive service is selected. We have varied the timeout value $0 \leq \theta \leq \delta_p$ and determined the expected revenue for given θ. The overview of the simulation parameters and their values used for the experiments are given in Table 1.

Table 1. Overview of model parameters

Parameter	Definition	Value
f_1	Response–time distribution of $\mathrm{CS}_i(1)$	Bimodal
f_2	Response–time distribution of $\mathrm{CS}_i(2)$	Lognormal
c_1	Cost of invocation of $\mathrm{CS}_i(1)$	1
c_2	Cost of invocation of $\mathrm{CS}_i(2)$	10
δ_p	End–to–end deadline	
R	Reward per request within deadline δ_p	20
V	Penalty per request not completed within deadline	50
$\mathbb{E}$	Expected revenue without request replication	
$\mathbb{E}_{1\to 2}$	Expected revenue with request replication	
G	Gain of expected revenue	
θ_i	Timeout value for execution of the task	

The relative gain of expected revenues is calculated as $G = \frac{\mathbb{E}_{1\to 2} - \mathbb{E}}{\mathbb{E}}$. The value of the deadline δ_p has decreased from 18 down to 3. The simulation results are shown in the graphs presented in Figure 4 and these are also summarized in Table 2. The following observations and conclusions could be made:

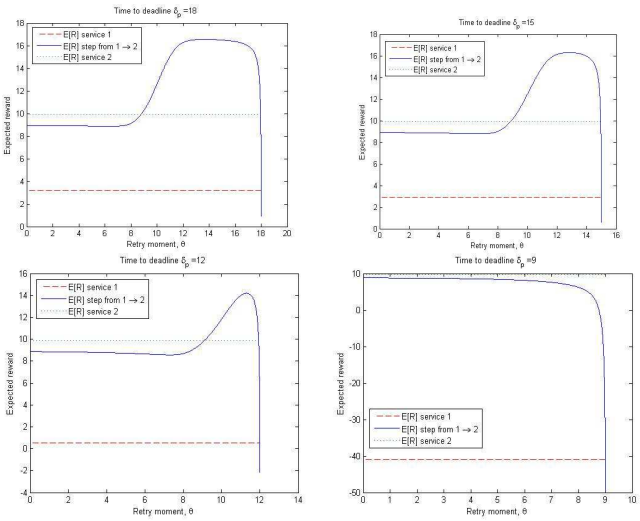

Fig. 4. Overview of the revenue gains for conditional request replication. In clockwise direction, starting from the top left corner, the deadline δ_p is 18, 15, 9 and 12, respectively.

- The scheme has its benefits for certain range of given deadlines, and when applicable, a "window of opportunity" for a retry. This interval becomes smaller as the remaining time to deadline becomes smaller.
- The gain increases as the remaining time to deadline increases. This is the consequence of the fact that it is easier to meet the deadline with the retry when there is more remaining time.
- The gains are possible with retry scheme even when the selected service is not the optimal one. For example, the expected reward for more expensive service (with lognormal response time distribution) is higher for all deadline values ≤ 20. Therefore, one may consider to select this expensive and fast service in such a case. However, we see that, when deadline is, e.g. 18, it is better to *first select* cheap and slow service, and, only when there is no response till e.g. 13 seconds, make a retry. By applying this scheme, much more revenue may be generated for the service provider. This is a consequence to the fact, that a lot of requests would be served by slow service (for given example well over 50%) and the execution costs differ 10 times.
- There is no gain of the proposed scheme when $\delta_p \leq 9$. In such a case the initial service selection should be the fast (and expensive) service. This is noticable from the graph given for δ_p - the expected reward for the whole range of retry moments with initial choice of expensive service is bigger than expected reward of retry scheme, which in turn is bigger than the expected reward when initial choice is cheap (and slow) service.

Table 2. Summary of experimental results

Deadline (δ_p)	Retry moment θ^*	Revenue: with retry	without retry	Revenue gain(%)
18	13.5	16.8	9.94	69.1
15	13.4	16.5	9.91	66.4
12	11.3	14.5	9.88	46.7

6 Summary and Future Work

In this paper we considered the runtime service adaptation mechanism for service compositions that is based on conditional retries. A single retry to the same or alternative service may be issued while task within composition is executed. We have analysed the impact of different QoS parameters, namely response–time distributions and cost parameters to the applicability of the scheme, and the potential revenue gain for the composite service provider.

The analysis has been performed for a relatively simple sequential workflow, under assumption that response–time distributions are accurate and time–invariant. In practise, however, these distributions change over time, e.g. due temporary overload of the service, and need to be estimated. The estimation is based on response–time measurements over a finite time interval, and therefore may change over time. This needs to be addressed by methods that would recalculate the timeout values, with the main issue of optimal number of recalculations. Next to it, we plan to investigate applicability of the retry mechanism for different workflow patterns and more complex workflows.

Yet another possibility to extend the research is to find the optimal retry mechanisms when penalty function is linearly increasing, with or without the cap. In such a case the minimization of the response time, even when penalty deadline is missed may be the optimal retry scheme.

Acknowledgment. Part of this work has been carried out in the context of the IOP GenCom project Service Optimization and Quality (SeQual), which is supported by the Dutch Ministry of Economic Affairs, Agriculture and Innovation via its agency Agentschap NL.

References

1. Preist, C.: A Conceptual Architecture for Semantic Web Services. In: McIlraith, S.A., Plexousakis, D., van Harmelen, F. (eds.) ISWC 2004. LNCS, vol. 3298, pp. 395–409. Springer, Heidelberg (2004)
2. Ward, C., Buco, M.J., Chang, R.N., Luan, L.Z.: A Generic SLA Semantic Model for the Execution Management of E-business Outsourcing Contracts. In: Bauknecht, K., Tjoa, A.M., Quirchmayr, G. (eds.) EC-Web 2002. LNCS, vol. 2455, pp. 363–376. Springer, Heidelberg (2002)
3. Ardagna, D., Comuzzi, M., Mussi, E., Pernici, B., Plebani, P.: PAWS: A Framework for Executing Adaptive Web-Service Processes. IEEE Software (24), 39–46 (2007)

4. Baresi, L., Ghezzi, C., Guinea, S., Krämer, H.: Towards Self-healing Composition of Services. In: Krämer, B.J., Halang, W.A. (eds.) Contributions to Ubiquitous Computing. SCI, vol. 42, pp. 27–46. Springer, Heidelberg (2007)
5. van Moorsel, A., Wolter, K.: Analysis of Restart Mechanisms in Software Systems. IEEE Trans. on Software Engineering (32), 547–558 (2006)
6. van Moorsel, A., Wolter, K.: Optimal restart times for moments of completion time. IEEE Proc. of Software Engineering 151(5), 219–223 (2004)
7. Wolter, K.: Stochastic Models for Restart, Rejuvenation and Checkpointing. Habilitation thesis, Humboldt-University, Berlin, Germany (2008)
8. Okamura, H., Dohi, T., Trivedi, K.S.: On-Line Adaptive Algorithms in Autonomic Restart Control. In: Xie, B., Branke, J., Sadjadi, S.M., Zhang, D., Zhou, X. (eds.) ATC 2010. LNCS, vol. 6407, pp. 32–46. Springer, Heidelberg (2010)
9. Cardellini, V., Casalicchio, E., Grassi, V., Lo Presti, F.: Adaptive Management of Composite Services under Percentile-Based Service Level Agreements. In: Maglio, P.P., Weske, M., Yang, J., Fantinato, M. (eds.) ICSOC 2010. LNCS, vol. 6470, pp. 381–395. Springer, Heidelberg (2010)
10. Leitner, P., Hummer, W., Dustdar, S.: Cost–Based Optimization of Service Compositions. Journal Trans. on Services Computing (TSC) (to appear)
11. Leitner, P., Hummer, W., Satzger, B., Dustdar, S.: Stepwise and Asynchronous Runtime Optimization of Web Service Compositions. In: Bouguettaya, A., Hauswirth, M., Liu, L. (eds.) WISE 2011. LNCS, vol. 6997, pp. 290–297. Springer, Heidelberg (2011)
12. Živković, M., Bosman, J.W., van den Berg, H., van der Mei, R., Meeuwissen, H.B., Núñez–Queija, R.: Run-time Revenue Maximization for Composite Web Services with Response Time Commitments. In: 26th IEEE Conference on Advanced Information Networking and Applications, AINA (2012)
13. Yousefi, A., Down, D.G.: Request Replication: An Alternative to QoS Aware Service Selection. In: Proceedings of the 2011 IEEE Conference of Service Oriented Computing and Applications (SOCA 2011), pp. 1–4 (2011)
14. Zeng, L., Benatallah, B., Ngu, A.H.H., Dumas, M., Kalagnanam, J., Chang, H.: QoS–aware middleware for web services composition. IEEE Transactions on Software Engineering 30(5), 311–327 (2004)
15. Yang, Y., Tang, S., Xu, Y., Zhang, W., Fang, L.: An Approach to QoS-Aware Service Selection in Dynamic Web Service Composition. In: 3rd IEEE Int. Conf. on Networking and Services (ICNS 2007), pp. 18–23 (2007)
16. Wikipedia: Bimodal distribution, `http://en.wikipedia.org/wiki/Bimodal_distribution`
17. Chen, L., Yang, J., Zhang, L.: Time Based QoS Modeling and Prediction for Web Services. In: Kappel, G., Maamar, Z., Motahari-Nezhad, H.R. (eds.) ICSOC 2011. LNCS, vol. 7084, pp. 532–540. Springer, Heidelberg (2011)