Revenue Maximization with Quality Assurance for Composite Web Services

Daniël Worm, Miroslav Živković, Hans van den Berg
TNO
Delft, The Netherlands

Rob van der Mei
CWI
Amsterdam, The Netherlands

***Abstract*—Service composition is one of the major approaches in service oriented architecture (SOA) based systems. Due to the inherent stochastic nature of services execution environment the issue of composite service quality assurance within SOA is a very challenging one. Such heterogeneous environment requires dynamic, run–time composition of services. In this paper we show how to determine a policy that satisfies the quality assurance for the composite service provider with the aim of revenue maximization for this provider. The quality assurance is defined as the probability that end–to–end deadline will be met, while taking into account service availability, (composite) service response time and costs. The calculated policy is determined using the dynamic programming and allows fast decision making for run–time composition. Besides, we determine the end–to–end response–time distributions resulting from determined policies. We illustrate the proposed solution with a number of experiments.**

***Index Terms*—composite web service; quality assurance; revenue maximization; availability; response time;**

I. Introduction

Web service's QoS plays an essential role in web service selection and composition within service oriented architecture (SOA) systems. The heterogeneous and, in general, inherent stochastic nature of services' execution environment makes the issue of *composite service quality assurance* within SOA challenging, i.e. how to provide an accurate QoS value and maintain promised (agreed upon) value for the composite web service consumers. High service assurance is one of the crucial aspects of applications in different areas, e.g. medical, transport, military, as these typically require high availability, reliability and dependability. At the same time, these applications have to be cost–efficient and satisfy the promised end–to–end response–time deadlines. If the composite web service provider relies on a set of services being available and one of those services becomes unavailable (even transiently), it could have dire consequences on the whole composite service. In order for the services to be used (for which the service provider receives compensation), these services must be available when needed. Otherwise, the provider's reputation could be impacted, and the profit may diminish, especially if penalties have to be paid when the services are not available. High quality assurance is therefore one of the key elements for the commercial success of SOA.

This paper addresses the problem of high quality assurance of composite services within SOA, based on *runtime* service composition that maximizes revenues for the composite service provider. The quality assurance is defined as the probability that end–to–end deadline will be met. Motivated by practical applications, an important feature of our model is that is takes into account multiple QoS service parameters, namely availability, response time performance and service costs. We use a dynamic programming approach [9] in order to derive the selection policy that will be used at runtime. This policy is determined *before* actual deployment of composite service, and allows optimal service selection at runtime. Our solution therefore requires no lengthy (and time consuming) re–calculation of the service composition at runtime. At the same time, the calculated policy takes into account *actual* time to deadline, current *availability* of the services, execution *costs* and expected *revenue* of the composite service provider.

To illustrate our approach, we observe the orchestrated composite web service which sequential workflow is depicted at Fig. 1. The workflow is based on an unambiguous functionality description of a service ("abstract service"), and several functionally identical alternatives ("concrete services") may exist that match such a description [2]. The workflow in Fig. 1 consists of four abstract services, and each abstract service maps to a number of concrete services (alternatives), which a deployed by (independent) third–party service providers. When the client's request is served within the agreed end–to–end deadline, the composite service provider is rewarded by the client. Otherwise, when the deadline is not met or the service request has not been processed due to the un–availability of selected services the provider pays a penalty to the client. After the execution of a single task within the workflow, the orchestrator decides on the next concrete service to be executed, and composite service provider pays to the third party provider per single invocation. The decision points for given tasks are illustrated at Fig. 1 by A, B, C and D, respectively. The decision taken is based on taking into account (1) which of the considered services are available at the decision moment, (2) execution costs, and (3) the remaining time to meet the end–to–end deadline. Therefore, our approach is based on fully dynamic, run–time service selection and composition, taking into account the response–time commitments from service providers and availability of the services. The main goal of this run–time service selection and composition is the quality assurance with the aim of revenue maximization for the composite service provider.

The main contributions of our paper could be summarized

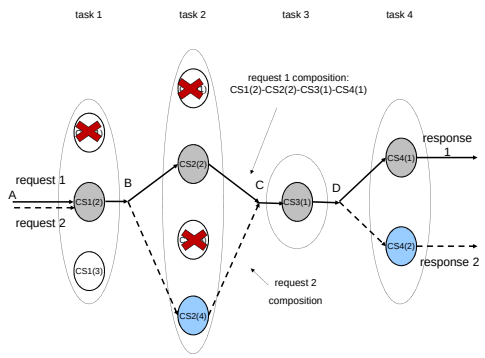

Fig. 1: Orchestrated composite web service depicted by a sequential workflow. Dynamic run–time service composition is based on pre–calculated policy. Decisions are taken at points A–D, and e.g. it is known to the orchestrator at B which services implementing task 2 are available at the decision instance.

as:

- We show how to determine the quality assurance for the composite service provider. We analyse the relation between the optimal revenue policy and quality assurance, and explain how the target assurance (e.g. assurance required by particular customer) for the composite service can be achieved. The revenue policy is optimal with respect to the profit maximization of composite service provider, and is a foundation for the dynamic, run–time service composition.
- We analyse the resulting end–to–end response–time of the composite service, and explain why our method results in a system which response–time has reduced variance with respect to the commitments of composite service provider.

The paper is organized as the following: after overview of relevant work in Section II, we briefly describe in Section III the model of the system under consideration. In Section IV we formulate the run–time service composition problem as a dynamic programming problem and provide the corresponding backward recursion solution method used to determine the optimal decision policy. Next, in Section V, we explain how to determine the end–to–end assurance and show strategies to increase it. In Section VI we give the formulae for calculation of the response time distributions taking the availability of the services into account as well. In Section VII, the results of extensive numerical experiments are presented and discussed. We conclude the paper and give directions for further research in Section VIII.

II. Related work

A number of solutions have been proposed for the problem of dynamic, run–time QoS–aware service selection and composition within SOA, [3]–[6]. These (proactive) solutions aim to adapt the service composition dynamically at run time. In [3] the authors analyse a problem of dynamic web service composition for different composition patterns. QoS parameters considered include reliability, availability, as well as cost and *expected value* of the response time of individual web services. The authors propose a solution based on Markov Decision Processes to minimize the expected response time, taking into account the availability and reliability of the respective services and the invocation costs. However, the authors do not consider the stochastic nature of response time, but the expected value of it. Besides, they do not consider the cost structure, revenue and penalty model as given in this paper. Doshi et al. in [4] presents a policy–based approach for dynamically choreographing web services. The goal of the decision policy is to minimize the execution cost, while the impact of the reward and penalty on per request base are not analysed. Besides, the impact of the service availability on the revenue of composite service provider is not addressed in this paper. Cardellini et al. in [5] consider dynamic service composition in the context of admission control with various service classes. The (different) composite service configurations for the service classes can be dynamically adapted according to variations in the operating environment due to the admission or departure of users generating requests for the composite service. The authors derive an optimal admission and re–composition policy (by formulating the problem as a linear optimization problem) that maximizes the profit while guaranteeing QoS for the admitted users. The paper [6] considers the problem of dynamic run–time service composition with the revenue maximization as the goal. It uses the dynamic programming approach in order to determine the policy to maximize the revenue of the composite service provider.

Despite the fact that each of these papers provide useful results on improving the response time performance, neither [5] nor [6] take into account the availability of services. This observation is the main motivation for the present paper, as we take into account both availability and response time performance aspects. The service (un)availability may have significant impact for the end–to–end service composition and response–time commitments made by the composite service providers, as illustrated further in the paper.

III. Sequential workflow decision model

In this section we briefly describe our model based on a composite web service represented by a sequential workflow. Our solution is applicable to any workflow that could be aggregated and mapped into a sequential one. Some basic rules for aggregation of a non–sequential into the sequential workflow have been illustrated in, e.g. [6], [7], [10]. However, the aggregation leads to coarser control, since decisions could not be taken for a single service within the aggregated workflow, but rather for the aggregated workflow patterns themselves.

Per single composite service request, the orchestrator executes tasks one–by–one as indicated by the sequential workflow. There are in total N tasks in the workflow and the task position within workflow is indexed by $i,\ i = 1, 2, \ldots, N$. Each task i maps onto $M_i \geq 1$ concrete services (alternatives),

where service j implementing task i is denoted by $\text{CS}_i(j)$, see also Fig. 1.

We denote by $c_{i,j}$ the cost that the composite provider pays for the single invokation of service $\text{CS}_i(j)$. To stochastic nature of response–time for service $\text{CS}_i(j)$ is modelled by probability density function (PDF) or respective cummulative distribution function (CDF). The PDF and CDF for service $\text{CS}_i(j)$ are denoted by $f_{i,j}(t)$ and $F_{i,j}(t)$, respectively.

We define the composite service ***assurance*** T as the probability that a single request will be executed within end–to–end penalty deadline δ_p. Composite service provider is rewarded by R when single request is completed within the deadline, otherwise, composite service provider pays penalty V to the end customer when this deadline is not met. This penalty is also enforced when none of the concrete services is available for task i and therefore the composite service provider cannot generate the response at all.

We assume that the orchestrator knows (due to constant monitoring of the services) which of the service alternatives are available for task i, as some of the alternatives may not be available at the decision moment. The availability of services is expressed by the probability $p_{i,j}$ that the concrete service $\text{CS}_i(j)$ is operational and accessible when required for use, [1]. The orchestrator does *not* know at service selection moment for task i which of the concrete services (for tasks $i+1$ and onwards) are available, but it does know the availability probabilities of these. In absence of constant service availability monitoring, the orchestrator could, due to the lack of knowledge, submit the request to the service that is temporarily unavailable. This issue may be overcome by establishing the response–time thresholds that indicate when certain service may be considered unavailable. This approach would further require an implementation of "watchdog timer" policy, i.e. determination of the optimal moments to terminate the current request, and perform the retry, using the same or a functionally equivalent service implementing the same task, [8]. This is beyond the scope of this paper, and will not be further considered here.

IV. Algorithm for optimising expected CSP revenue

In this section we describe how to optimise expected revenue in the setting described in the previous section, by formulating the dynamic service selection as a dynamic programming (DP) problem, [9]. Although formulae deduced here may appear "similar" to those developed in [6], the inclusion of the services' availability within these formulae is a challenging task. The given formulae result in different (and more accurate) service composition policies, which, consequently, leads to *significant* revenue improvements, as illustrated in Section VII. This also stresses the need to at jointly investigate impact of availability and response–time parameters for the service composition.

Due to the fact that we take into account the actual availability of services, different from "classical" DP approach, we store a permutation (ordering) of concrete services implementing task i. This permutation is denoted by $j_1, \ldots, j_{M_i}$. When the orchestrator makes service selection decision, it will use these orderings to choose the 'best' concrete services among the ones that are available. Besides, if the deadline expires during the execution of the composite service request, with some tasks still not executed, the decision rule is to use the cheapest services available for each of the remainin tasks.

The decision policy is determined by the current position within the sequential workflow i, the available services at this position, and the remaining time Δ till the overall deadline δ_p will be violated. When the remaining time to deadline has the value of δ^* a set of expected rewards $\mathbb{E}[R_i \mid \Delta = \delta^*]$ is defined by the DP recursion:

$$\begin{aligned}
&\mathbb{E}[R_i \mid \Delta = \delta^*] = \\
&= p_{i,j_1}\Big(- c_{i,j_1} + \mathbb{E}[R_{i,j_1} \mid \Delta = \delta^*] - \mathbb{E}[V_{i,j_1} \mid \Delta = \delta^*]\Big) + \\
&+ (1 - p_{i,j_1}) p_{i,j_2}\Big(- c_{i,j_2} + \mathbb{E}[R_{i,j_2} \mid \Delta = \delta^*] - \mathbb{E}[V_{i,j_2} \mid \Delta = \delta^*]\Big) \\
&+ \cdots + (1 - p_{i,j_1}) \cdots (1 - p_{i,j_{M_i - 1}}) p_{i,j_{M_i}} \cdot \qquad (1) \\
&\cdot \Big(- c_{i,M_i} + \mathbb{E}[R_{i,M_i} \mid \Delta = \delta^*] - \mathbb{E}[V_{i,M_i} \mid \Delta = \delta^*]\Big) \\
&+ (1 - p_{i,j_1}) \cdots (1 - p_{i,j_{M_i}}) V.
\end{aligned}$$

For given task i, and without loss of generality, there is an order $j_1, \ldots, j_{M_i}$ of concrete services such that choosing j_1 would lead to the highest expected CSP revenue, choosing j_2 to the second highest expected CSP revenue, etc. This is specified as

$$\begin{aligned}
&-c_{i,j_1} + \mathbb{E}[R_{i,j_1} \mid \Delta = \delta^*] - \mathbb{E}[V_{i,j_1} \mid \Delta = \delta^*] \geq \\
&\quad \geq -c_{i,j_2} + \mathbb{E}[R_{i,j_2} \mid \Delta = \delta^*] - \mathbb{E}[V_{i,j_2} \mid \Delta = \delta^*] \geq \\
&\quad \geq \cdots \geq \qquad (2) \\
&\quad \geq -c_{i,j_{M_i}} + \mathbb{E}[R_{i,j_{M_i}} \mid \Delta = \delta^*] - \mathbb{E}[V_{i,j_{M_i}} \mid \Delta = \delta^*].
\end{aligned}$$

The selection strategy by the orchestrator for given task i is based on choosing service $\text{CS}_i(j_1)$ if available; if not, choosing $\text{CS}_i(j_2)$ if available, and so on. The term $\mathbb{E}[R_{i,j} \mid \Delta = \delta^*]$ represents the expected reward, when concrete service $\text{CS}_i(j)$ is executed for the given remaining time value δ^*. The term $\mathbb{E}[V_{i,j} \mid \Delta = \delta^*]$ represents the expected penalty for exceeding the overall deadline while executing service $\text{CS}_i(j)$. The expected reward and penalty functions take into account the impact of *future decisions*. This leads to the optimal solution of the problem, [9].

However, the evaluation of $\mathbb{E}[R_{i,j} \mid \Delta = \delta^*]$ and $\mathbb{E}[V_{i,j} \mid \Delta = \delta^*]$ requires the usage of PDF (CDF), which usually results in rather complicated integral equations, see [6]. To resolve this issue, the discretization of distributions is required. The response–time of interest (end–to–end deadline) is therefore split into segments of the same size h. For the total number of segments m^*, this leads to discretised versions of the PDF ($p_{i,j,k}$) and CDF ($P_{i,j,k}$):

$$\begin{aligned}
p_{i,j,k} &= P\big(D_{i,j} \leq h[k + 0.5]\big) - P\big(D_{i,j} \leq h[k - 0.5]\big), \quad (3) \\
P_{i,j,k} &= \sum_{l=0}^{k} p_{i,j,l},
\end{aligned}$$

where $i = 1, \ldots, N,\ j = 1, \ldots, M_i,\ k = 0, \ldots, m^*$.

Let terms R^*_{i,m^*}, R^*_{i,j,m^*}, and V^*_{i,j,m^*} represent discretised versions of $\mathbb{E}[R_i \mid \Delta = \delta^*]$, $\mathbb{E}[R_{i,j} \mid \Delta = \delta^*]$, and $\mathbb{E}[V_{i,j} \mid \Delta = \delta^*]$, respectively. The backward recursion formulae are then as follows:

$$\begin{aligned} R^*_{i,m^*} = {} & p_{i,j_1}\Big(- c_{i,j_1} + R^*_{i,j_1,m^*} - V^*_{i,j_1,m^*}\Big) + \\ & + (1-p_{i,j_1})p_{i,j_2}\Big(- c_{i,j_2} + R^*_{i,j_2,m^*} - V^*_{i,j_2,m^*}\Big) + \\ & + \cdots + (1-p_{i,j_1})(1-p_{i,j_2})\cdots(1-p_{i,j_{M_i-1}})p_{i,j_{M_i}} \cdot \\ & \cdot \Big(- c_{i,j_{M_i}} + R^*_{i,j_{M_i},m^*} - V^*_{i,j_{M_i},m^*}\Big) + \\ & + (1-p_{i,j_1})(1-p_{i,j_2})\cdots(1-p_{i,j_{M_i}})V, \end{aligned} \tag{4}$$

where $i = 1, \ldots, N$, and, without loss of generality, $j_1, \ldots, j_{M_i}$ is a permutation of $(1, 2, \ldots, M_i)$, such that

$$\begin{aligned} -c_{i,j_1} + R^*_{i,j_1,m*} - V^*_{i,j_1,m*} \geq \\ \geq -c_{i,j_2} + R^*_{i,j_2,m*} - V^*_{i,j_2,m*} \geq \cdots \geq \\ \geq -c_{i,j_n} + R^*_{i,j_{M_1},m*} - V^*_{i,j_{M_1},m*} \end{aligned} \tag{5}$$

$$R^*_{i,j,m^*} = \begin{cases} P_{N,j,m^*}R, & i = N, \\ \sum\limits_{k=0}^{m^*} p_{i,j,k}R^*_{i+1,m^*-k}, & i = 1, \ldots, N-1, \end{cases} \tag{6}$$

and

$$V^*_{i,j,m^*} = \begin{cases} (1 - P_{N,j,m^*})\,V, & i = N, \\ (1 - P_{i,j,m^*})\,R^*_{i+1,0}, & i = 1, \ldots, N-1, \end{cases} \tag{7}$$

where $j = 1, \ldots, M_i,\ k = 0, \ldots, m^*$.

While applying formulae (4)–(7), the corresponding decisions (actions) can be obtained by storing $(j_1, \ldots, j_{M_i})$ for every task $i = 1, \ldots, N$ and every possible time interval m^*, where $(j_1, \ldots, j_{M_i})$ is a permutation of $(1, 2, \ldots, M_i)$.

We define a (deterministic) decision strategy S for every $i = 1, \ldots, N$ and every time interval m^*, as a permutation $S(i, m^*) := (j_1, \ldots, j_{M_i})$ of $(1, 2, \ldots, M_i)$. The choice to be made for task i and given time interval m^* is then preferably to choose concrete service $\mathrm{CS}_i(j_1)$. If this service is not available, then choose $\mathrm{CS}_i(j_2)$ and so on. We denote by S^* the decision strategy that gives the optimal expected revenue, i.e. this is decision strategy such that (5) is satisfied. In the next section we address the relation between the optimal revenue decision strategy S^* and end–to–end quality assurance T.

V. Relation between optimal revenue strategy and quality assurance

In general, the optimal decision strategy S^* may not necessarily guarantee the highest quality assurance $T_{\max}$. In this section we will first develop the algorithm to calculate the end–to–end assurance and show strategies to increase end–to–end assurance, while sacrificing expected revenue.

A. Calculation of end-to-end assurance

In order to calculate the end–to–end assurance we use the (discretised) setting as described in Section IV, equations (3)-(7). The deadline δ_p can be expressed as $m_p = \lceil \delta_p / h \rceil$, i.e. it is expressed as the number of discretisation segments. For a given decision strategy S, we define the end–to–end assurance $T = T(S)$ of the (discretised) decision strategy S to be the probability that the deadline m_p will be kept.

The end–to–end assurance can be calculated using the following recursive relations:

$$T^*_{i,j,m^*} = \begin{cases} P_{N,j,m^*}, & i = N, \\ \sum\limits_{k=0}^{m^*} p_{i,j,k}T^*_{i+1,m^*-k}, & i = 1, \ldots, N-1, \end{cases} \tag{8}$$

and

$$\begin{aligned} T^*_{i,m^*} = {} & p_{i,j_1}T^*_{i,j_1,m^*} + (1-p_{i,j_1})p_{i,j_2}T^*_{i,j_2,m^*} + \cdots + \\ & + (1-p_{i,j_1})(1-p_{i,j_2})\cdots(1-p_{i,j_{M_i-1}})p_{i,j_{M_i}}T^*_{i,j_{M_i},m^*} \end{aligned} \tag{9}$$

where $S(i, m^*) = (j_1, \ldots, j_{M_i})$. The end–to–end assurance $T(S)$ equals T^*_{1,m_p}. Clearly $0 \leq T(S) \leq 1$ for any given strategy S.

In general, not any value of end–to–end assurance may be reached. The highest possible end–to–end assurance can be calculated as described with the ordering $(j_1, \ldots, j_{M_i})$ of concrete services at each task i and time interval m^* on the assurance factors T^*_{i,j,m^*}. The highest assurance is achieved for permutation $(j_1, \ldots, j_{M_i})$ such that

$$T^*_{i,j_1,m^*} \geq T^*_{i,j_2,m^*} \geq \cdots \geq T^*_{i,j_{M_i},m^*}.$$

Remark: The optimal revenue strategy S^* defined in section IV need not be unique, because there might be different permutations $(j_1, \ldots, j_{M_i})$ of $(1, \ldots, M_i)$ such that (5) is satisfied, namely when there are equalities. However, each revenue policy S has accompanied assurance T^*_{i,j,m^*}, calculated using (8)-(9). In case when there are two or more revenue policies that yield the same revenue, these policies can be ordered according to their respective assurance. In case when two or more policies S yield the same revenue, and have the same assurance, we further order them according to the lexicographic ordering of permutations. This is how the uniqueness of the selection strategy is achieved.

B. Optimal revenue strategy with target assurance

We denote by S^*_V the unique optimal revenue decision strategy for given penalty V, by $\mathbb{E}R_V(S)$ we denote the expected revenue for strategy S and given penalty V. It holds that $\mathbb{E}R_V(S^*_V) \geq \mathbb{E}R_V(S)$ for all selection strategies S. However, the optimal revenue strategy S^*_V need not have the highest end–to–end assurance, for instance due to expensive but high assurance individual services in the chain. As explained in the remark, the assurance $T(S^*_V)$ is maximal among all strategies S *under condition* that $\mathbb{E}R_V(S) = \mathbb{E}R_V(S^*_V)$.

The composite service provider may need to guarantee a certain minimal end–to–end assurance T for given penalty V. This assurance might not be attained by the optimal revenue

selection strategy S_V^*. This establishes the need to determine a different strategy $S \neq S_V^*$ that has maximal expected revenue *under the condition* that it has end-to-end assurance greater than or equal to T. The strategy S could be determined by usage of the following theorem.

Theorem V.1. *Let penalties W and V satisfy $0 \leq V < W$. For given end–to–end deadline, let S_W be the optimal revenue strategy for penalty W with assurance $T = T(S_W)$. Then $\mathbb{E}R_V(S_W) \geq \mathbb{E}R_V(S)$ for all selection strategies S with $T(S) \geq T$.*

Proof: Let C_{11} be the expected cost paid when strategy S_W is used under the condition that the end–to–end deadline is met, and C_{12} be the expected cost paid when strategy S_W is used, under the condition that the end–to–end deadline is not met. Given the assurance $T = T(S_W)$, the expected revenue is

$$\mathbb{E}R_W(S_W) = T \cdot (-C_{11} + R) + (1 - T) \cdot (-C_{12} - W).$$

Let S be any selection strategy under condition $T(S) \geq T$. Let C_{21} be the expected cost when strategy S is used, under the condition that the end–to–end deadline is met, and let C_{22} be the expected cost paid when strategy S is used, under the condition that the end–to–end deadline is not met. Then

$$\mathbb{E}R_W(S) = T(S) \cdot (-C_{21} + R) + (1 - T(S)) \cdot (-C_{22} - W).$$

Since S_W is optimal revenue strategy for given penalty W, it holds that $\mathbb{E}R_W(S_W) \geq \mathbb{E}R_W(S)$, i.e.

$$\begin{aligned} T \cdot (-C_{11} + R) + (1 - T) \cdot (-C_{12} - W) \geq \\ T(S) \cdot (-C_{21} + R) + (1 - T(S)) \cdot (-C_{22} - W). \end{aligned}$$

Now, adding $(1 - T)(W - V)$ on both sides, and given the fact that $(1 - T)(W - V) \geq (1 - T(S))(W - V)$, we obtain:

$$\begin{aligned} &\mathbb{E}R_V(S_W) = T \cdot (-C_{11} + R) + (1 - T) \cdot (-C_{12} - V) \\ &= T \cdot (-C_{11} + R) + (1 - T) \cdot (-C_{12} - W) + (1 - T)(W - V) \\ &\geq T(S) \cdot (-C_{21} + R) + (1 - T(S)) \cdot (-C_{22} - W) + (1 - T)(W - V) \\ &\geq T(S) \cdot (-C_{21} + R) + (1 - T(S)) \cdot (-C_{22} - W) + (1 - T(S))(W - V) \\ &= T(S) \cdot (-C_{21} + R) + (1 - T(S)) \cdot (-C_{22} - V) = \mathbb{E}R_V(S), \end{aligned}$$

which completes the proof. ∎

The following conclusions could be made from the given theorem:

- The end-to-end assurance increases with the increase of the penalty value.
- The optimal expected revenue decreases with the increase of the penalty value.
- When optimal revenue strategy S_V^* for given penalty V does not meet target assurance, a new strategy S_W needs to be calculated. This is done by finding the smallest penalty value $W(W > V)$, for which the optimal revenue strategy S_W meets the target assurance T. The revenue strategy S_W is then optimal under condition that target assurance is met.

VI. Calculation of end-to-end response time distribution

The end-to-end assurance only describes the probability to complete the request before the deadline. The assurance corresponding to a selection strategy S could be readily calculated from the end-to-end response time distribution. However, the determination of the end–to–end response time distribution allows to identify, e.g. strategies that have response time close to the deadline with relatively high probability. Therefore, we describe here how the end-to-end response time distribution r^* of a strategy S can be computed. In section VII we will demonstrate the usage of such distributions.

Suppose we have a selection strategy S, that will give, for every abstract service i and time t, a permutation $(j_1, \ldots, j_{M_i})$ of $(1, \ldots, M_i)$, i.e. $S(i,t) = (j_1, \cdots, j_{M_i})$, such that, if available, concrete service j_1 will be selected, else concrete service j_2 if available, etc. Let $r_{s,i,j}$ be the response time distribution when $\mathrm{CS}_i(j)$ is available and selected, and the remaining time to deadline equals to s. Further, let $r_{s,i}$ be the response time distribution for task i and remaining time to deadline s, i.e. no concrete service for task i is selected yet. When no concrete services are available for given task, the composite request will not be completed, and, for such a scenario, we assume the response time to be infinite. We denote this by function $\mathbb{1}_\infty(t)$ that equals to 1 when response time is infinite, and equals to zero otherwise. Then:

$$\begin{aligned} r_{s,i} = p_{i,j_1} r_{s,i,j_1} + (1 - p_{i,j_1}) p_{i,j_2} r_{s,i,j_2} + \cdots + \\ + (1 - p_{i,j_1})(1 - p_{i,j_2}) \cdots (1 - p_{i,j_{M_i-1}}) p_{i,j_{M_i}} r_{s,i,j_{M_i}} + \\ + (1 - p_{i,j_1}) \cdots (1 - p_{i,j_{M_i}}) \cdot \mathbb{1}_\infty. \end{aligned} \tag{10}$$

The response time distribution for $\mathrm{CS}_i(j)$, with remaining time to deadline s is

$$r_{s,i,j}(t) = \begin{cases} f_{N,j}(t), & i = N, \\ \int_0^\infty f_{i,j}(y)(R_+(y) r_{s-y,i+1})(t)\, dy, & \\ \qquad\qquad \text{for } i = 1, \ldots, N-1, & \end{cases} \tag{11}$$

where $(R_+(y)g)(t)$, is given by

$$(R_+(y)g)(t) = \begin{cases} g(t-y), & t \geq y, \\ 0, & t < y. \end{cases} \tag{12}$$

Due to the lack of space, we do not show here how to calculate discretised response time distribution.

VII. Numerical experiments

In this section we will give some numerical experiments of the theory described in previous sections, and state some observations coming from these experiments. Unless otherwise specified, we illustrate the results for the case of the sequential workflow that consists of $N = 4$ tasks (abstract services), and each task is implemented by four different concrete services (alternatives).

A. A static algorithm for comparison

We need first to determine a (static) reference to compare our results to. One approach may be, see [6], to determine an optimal path $W = (W_1, \cdots, W_N)$ in advance, such that for task i, a concrete service W_i is selected. This approach is not applicable for the problem we consider here because any of the individual services in the optimal path may be unavailable, which implies that the probability that the composite service never generates a reply can become quite large. As an illustration, let us analyse a sequential workflow with N tasks, where each concrete service has availability that is same and equals to a value $p < 1$. In case when an optimal path is determined, i.e. a single concrete alternative has been chosen for each task, and that choice is not changed, the probability that composite service request would not be served is $1 - p^N$, which goes to 1 quickly as N becomes large.

As this would mean an unfair assessment of the static approach, we adjust it as follows: For each task i, $i = 1, \ldots N$, a specific permutation $w_i^1, \ldots, w_i^{M_i}$ of concrete services $(1, \ldots, M_i)$ is chosen, such that

(1) of all possible compositions, $(w_1^1, \cdots, w_N^1)$ gives the optimal expected revenue, under the condition that these concrete services do not fail,
(2) for $i \in \{1, \cdots, N\}$, and $k \in \{1, \cdots, M_i\}$, denote $\mathbb{E}r_i^k :=$ the expected revenue for the composition $(w_1^1, \cdots, w_{i-1}^1, w_i^k, w_{i+1}^1, \cdots, w_N^1)$. Then $\mathbb{E}r_i^1 \geq \mathbb{E}r_i^2 \geq \cdots \geq \mathbb{E}r_i^{M_i}$.

This means that, in our reference case, the optimal (static) composition $(w_1^1, \cdots, w_N^1)$ is executed. However, whenever a concrete service w_i^1 at task i is unavailable, the concrete service w_i^2 is selected instead when available, otherwise w_i^3 if available, etc.

B. Revenue and assurance improvements with identical availability of services

In this subsection we illustrate the revenue and assurance improvements when our solution is applied, compared to the reference.

TABLE I: Parameters of concrete services implementing a single task i.

	Task i		
Concrete service↓	c	μ	σ
Alternative 1	1	5	2
Alternative 2	2	3	2
Alternative 3	5	2.5	2
Alternative 4	10	1.25	3

For each abstract service we assume we have four concrete services with parameters as in Table I, i.e. with cost c, and parameters μ and σ of the lognormal distribution of the response time. The end-to-end penalty deadline is $\delta_p = 18$, reward $R = 20$ and penalty value is $V = 50$. We also assume that concrete services for all tasks have the same availability probabilities, and these probabilities vary from 0.6 to 1. The results for expected revenue and end-to-end assurance are shown in Figure 2. The dynamic run–time selection strategy significantly outperforms the static reference strategy, with respect to both revenue and end-to-end assurance. The absolute difference among expected revenue for the dynamic and static strategy remains more or less constant, independent of availability. However, the end-to-end assurance comes farther apart as the availability probabilities approach one.

When the availability of concrete services equals one, the difference between expected revenue for the dynamic and static strategies results from the fact that the dynamic strategy takes into account realised response times of the subservices when selecting the concrete services for the next task. The dynamic strategy can therefore take advantage of a low realised response time, even if the probability for this realisation is small.

When availability probabilities differ from one, our dynamic strategy takes future availability uncertainties into account, while the static strategy does not. As we see in Figure 2 that the difference in expected revenue stays more or less constant, while the availability probabilities decrease, we conjecture that in this case the main contribution to the advantage of the dynamic strategy versus the static strategy lies in the fact that the dynamic strategy takes realised response times into account, and not that it takes future availability probabilities into account.

C. Expected revenue and quality assurance

In this section we illustrate an application of Theorem V.1. Again, the setting consists of a sequential workflow with four tasks, of which each has four alternatives with parameters specified in Table I. Furthermore, we assume that reward $R = 20$, the penalty $V = 50$, the penalty deadline $\delta_p = 18$, and the concrete services all have an availability of 1. The dynamic strategy in this case realises an end-to-end assurance of 0.9537, see also Figure 2. Let us now suppose that the minimal target end-to-end assurance is $T_t = 0.97$, and let us determine the strategy that yields optimal expected revenue under this additional requirement. We first check whether target assurance is achievable at all. To do so, we determine maximum assurance $T_{\max}$, as described at subsection V-A. We obtain the value of $T_{\max} = 0.98$, with an (negative!) expected revenue of -5.862. The target assurance of 0.97 is therefore possible.

Applying the Theorem V.1 we develop the following steps to determine the strategy:

1) Increase the initial penalty $V = 50$ by increments ΔV, say $\Delta V = 10$.
2) In $n-$th iteration, determine the strategy S_n that gives the optimal expected revenue for penalty $V + n\Delta V$. Calculate the resulting end-to-end assurance under strategy S_n, i.e. $T(S_n)$
3) For the first iteration n which results in assurance $T(S_n) \geq T_t$ compute the expected revenue for strategy S_n using the initial penalty $V = 50$.

The approach is illustrated in Figure 3 for the initial penalty value $V = 50$, $\Delta V = 10$ and $T_t = 0.97$. The target assurance T_t is reached for iteration $n = 44$, i.e. the penalty value of 490.

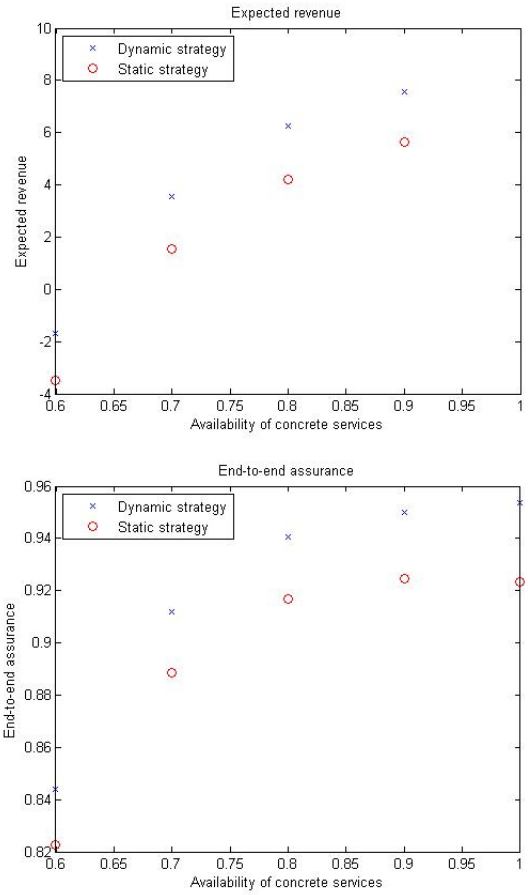

Fig. 2: Comparison between the static and dynamic strategies for service parameters given in Table I.

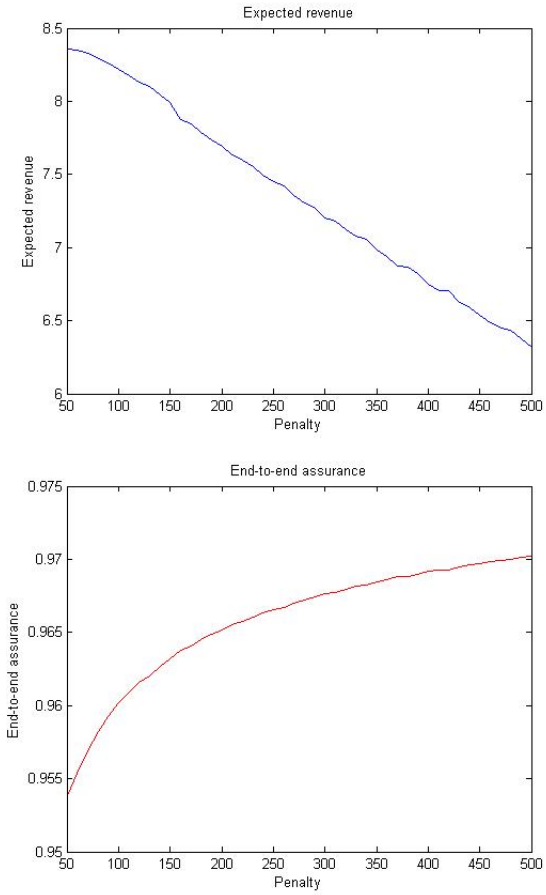

Fig. 3: Expected revenues and end-to-end assurance for different penalty values.

This results in strategy S_n which yields expected revenue of 6.37 for $V = 50$. By Theorem V.1 this is the highest expected revenue that can be achieved under the condition that the end-to-end assurance is greater than or equal to 0.97.

D. Impact of availability

In this set of experiments we investigated the alternating individual service availability, and the impact it has to expected revenue and end-to-end assurance. The parameters of the concrete services are specified in Table I. However, not all services have the same availability; we set the availability of all concrete services implementing task 1 to 0.55, while concrete services implementing tasks 2, 3 and 4 have availability of 0.65, 0.75 and 0.85, respectively. In the Table II all 24 possible compositions are shown. For example, configuration with label a uses alternative 4 at position (i.e. task) 1, alternative 3 at position 2, and so on. For respective configurations the resulting revenues and end–to–end assurance are illustrated in Figures 4 and 5, respectivelly. Both Figure 4 and Figure 5 show "gaps" of dramatic revenue increase (resp. assurance increase), noted by A, B and C in the figure. Careful examination of permutations from Table II indicates that gap A in Figure 4 results from the fact that the alternatives for the last two tasks change from 2 and 1 (with availabilities 0.65 and 0.55 respectively) to alternatives 3(4) and 1 for the same tasks. Similarily, gap B results from the fact that the alternative for the last task is not 1 anymore, but 2.

TABLE II: Indices of alternative assurance configurations

label →	a	b	c	d	e	f	g	h	i	j	k	l
position 1	4	4	4	4	4	4	3	3	3	3	3	3
position 2	3	3	2	2	1	1	4	4	2	2	1	1
position 3	2	1	3	1	2	3	2	1	4	1	2	4
position 4	1	2	1	3	3	2	1	2	1	4	4	2
label →	m	n	o	p	q	r	s	t	u	v	w	x
position 1	2	2	2	2	2	2	1	1	1	1	1	1
position 2	3	3	4	4	1	1	3	3	2	2	4	4
position 3	4	1	3	1	4	3	2	4	3	4	2	3
position 4	1	4	1	3	3	4	4	2	4	3	3	2

This leads to the following observations:

- The highest optimal revenue is achieved when the services with highest availability probabilities are placed at or near the end, and services with lowest availability probabilities are placed at or near the beginning of the service chain. One logical explanation for this is as follows: when optimal concrete services are unavailable at the beginning of the service chain, there is still enough "room" (in the time budget) to compensate this with a different strategy. When they are unavailable near the end,

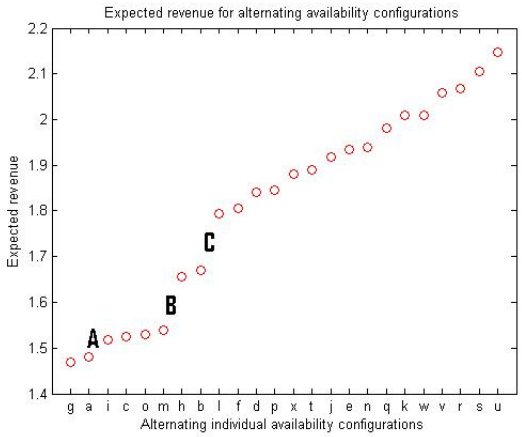

Fig. 4: Expected revenues for different availabilities, sorted by permutation indices.

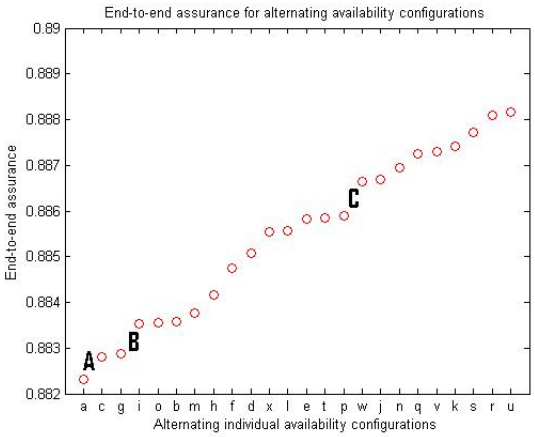

Fig. 5: End-to-end assurance for different availabilities, sorted by permutation indices.

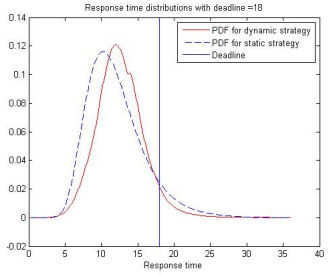

Fig. 6: Comparison of response–time PDFs for dynamic and reference (static) strategy when deadline=18.

this "room" to maneuver is much smaller.

- The above observation indicates that when considering which service alternatives could play a role in constructing a composite service, extra care must be taken to ensure that the service alternatives near the end of the chain have sufficient availability probabilities.

E. Response time distribution

We showed in Section VI how the response time PDF can be determined, and we illustrate the results in Figure 6. The same setting and parameters as in subsection VII-C are taken. From the respective PDFs we notice that dynamic response time PDFs shift more towards the deadline compared to the reference strategy. The dynamic strategy focuses on optimal expected revenue, and when more time is available, the strategy uses this time and select cheaper but slower services. It is also clear from Figure 6 that dynamic strategy results in higher assurance than the reference strategy.

VIII. Conclusions and Future Work

In this paper we presented a dynamic programming based solution for dynamic, run–time web service composition with two conflicting goals: revenue maximization and quality assurance. An important feature of our analysis is that we take both service availability and stochastic model of services' response time distribution. We have shown how to determine the policy that would result in maximum revenue for given target quality assurance. The promising results from this paper will be further investigated for larger workflow settings, real service performance data and for time–varying services' models.

Acknowledgment

Part of the work done in this paper has been performed within the scope of the project "Towards Trustworthy ICT Service Chains" (TTISC), funded by the Dutch government agency ICTRegie. The authors would like to thank Joost W. Bosman for his useful comments on an earlier version of this paper.

References

[1] L. O'Brien, L. Bass, and P. Merson: "Quality Attributes and Service–Oriented Architectures".

[2] C. Preist, "A conceptual architecture for semantic web services," in *Proc. International Semantic Web Conference (ISWC 2004)*, 2004.

[3] A. Gao, D. Yang, S. Tang, and M. Zhang, "Web Service Composition Using Markov Decision Processes", in *P*roc. of the 6th Iternational Conference on Web–Age Information Management, (WAIM 2005), pp. 308–319.

[4] P. Doshi, R. Goodwin, R. Akkiraju, and K. Verma, "Dynamic Workflow Composition using Markov Decision Processes". J. of Web Services Research (JWSR), vol. 2, no. 1, pp. 1-17, 2005.

[5] V. Cardellini, E. Casalicchio, V. Grassi, and F. L. Presti, "Adaptive management of composite services under percentile-based service level agreements," in *Proceedings of International Conference on Service-Oriented Computing 2010 (ICSOC 2010)*, 2010, pp. 381–395.

[6] M. Živković, J. W. Bosman, H. van den Berg, R. van der Mei, H. B. Meeuwissen, and R. Núnez–Queija, "Run–time revenue maximization for composite web services with reponse time commitments" in *Proc. of IEEE 26th International Conference on Advanced Information Networking and Applications, (AINA 2012)*, 2012, pp. 589–596.

[7] H. Zheng, J. Yang, W. Zhao, and A. Bouguettaya, "QoS Analysis for Web Service Compositions Based on Probabilistic QoS", in *Proceedings of the 2011 International Conference of Service Oriented Computing, (ICSOC 2011)*, G. Kappel, Z. Maamar, H.R. Motahari-Nezhad (Eds.): ICSOC 2011, LNCS 7084, pp. 47-61, 2011.

[8] M. Živković, H. van den Berg, "Analysis of Revenue Improvements with Runtime Adaptation of Service Composition Based on Conditional Request Retries" accepted to *European Conference on Service-Oriented and Cloud Computing, (ESOCC 2012)*.

[9] R. Bellman, *Dynamic Programming*. Mineola, NY: Dover Publications, Inc., 2003.

[10] G. L. Choudhury, D. J. Houck, "Combined queueing and activity network based modeling of sojourn time distributions in distributed telecommunication systems," in *Proceedings of 14th International Teletraffic Congress, (ITC 14)*, 1994, pp. 525–534.