Theoretical Considerations for Maintaining the Performance of Composite Web Services

Shinji KIKUCHI†, Member, Yoshihiro KANNA†, and Yohsuke ISOZAKI†, Nonmembers

SUMMARY In recent years, there has been an increasing demand with regard to available elemental services provided by independent firms for composing new services. Currently, however, whenever it is difficult to maintain the required level of quality of a new composite web service, assignment of the new computer’s resources as provisioning at the data center is not always effective, especially in the area of performance for composite web service providers. Thus, a new approach might be required. This paper presents a new control method aiming to maintain the performance requirements for composite web services. There are three aspects of our method that are applied: first of all, the theory of constraints (TOC) proposed by E.M. Goldratt; secondly, an evaluation process in the non-linear feedback method; and finally multiple trials in applying policies with verification. In particular, we will discuss the architectural and theoretical aspects of the method in detail, and will show the insufficiency of combining the feedback controlling approach with TOC as a result of our evaluation.

key words: composite web services, inter-enterprise collaboration, solution-level quality of service, performance issue in service-level agreement

1. Introduction

Service oriented architecture (SOA) is now being applied to practical systems [22]. Accordingly, independent firms have begun to provide composite web services in response to practical demands. Thus, research related to non-functional requirements, such as QoS (Quality of Service)-aware composite web services, negotiation processes for SLAs (service level agreements), and SLA-based autonomic computing, has accelerated, however, it is still ongoing [1], [24].

In general, there are various requirement items for SLAs. These are related to availability, reliability, security, maintainability, etc. In particular, performance, such as the average respond time, ideally could be a part of the above items. This has a dynamic aspect on the status, and tends to be affected by several outer factors. In order to maintain these SLA items, in general, a management process named SLM (service level management) is adopted for long-term maintenance rather than short periods. Therefore, identifying points for improvement by monitoring the SLA items for fixed intervals will be executed. Then in general, if required, safer approaches such as over-provisioning of resources will be applied. However, at present, whenever it is hard to maintain the required service level, in particular around performance, assignment of the new computer’s resources is not always effective, especially for composite web service providers. One solution could be that all composite web service providers must individually and preliminarily reach agreements on service levels with each provider of the elemental web services in order to ensure the contracts with the clients of the composite web service providers. Despite that, there might be certain limitations. For example, let us assume a case of the maximum customizable mash-up service as an instance of the composite web service. In this particular case, the client expects composition of their own favorite services with current existing elemental web services based on their demand. Further to that, the rigid contract process for SLA may become an obstacle to quick updating. At the initial stage of service level management, insufficient learning may make reaching an agreement with sufficient and up-to-date information about the computer’s resources assigned to the composite web services improbable. Therefore, a new method such as using a control function in executing those composite web services safely might be required, as a complementary approach for the above potential issues with regard to maintaining SLA under uncertain conditions.

However, there are a few constraints for such a method. Firstly, only those status values which are related to the SLA and observable from the outside are available. Secondly, because we cannot expect a specialized mechanism for actuators to be installed and deployed for each elemental web services provider, only independent attachable controlling actuators are trustworthy. Based on the second constraint, it is obvious that the aforementioned over-provisioning approach is insufficient.

Furthermore, there are still more difficulties in order to establish this control function. Firstly, there is a challenge in defining a suitable model of the control function due to composite web services’ layered structure, and also another difficulty in identifying the particular composite web service affected in a set of those services, when a controlling operation is to be carried out on an elemental web service. These are caused by complexity in modeling synchronicity of several invocations, error handling, and resulting compensation activities. Furthermore, the unfamiliarity of the linear control, which the current available methods have, also affects these problems. These difficulties are obvious, compared with the bandwidth control in communication. And finally, there are really only a few well-organized and concrete conceptual models to realize the control function in

Manuscript received September 22, 2011.
Manuscript revised June 10, 2012.
†The authors are with NEC Corporation, Tokyo, 108–8558 Japan.
a) E-mail: s-kikuchi@bp.jp.nec.com
DOI: 10.1587/transinf.E95.D.2634
In order to solve the above challenges, in particular for maintaining performance, we have proposed a new control method as [23] shows. Firstly, we apply the idea of TOC (theory of constraints) [2]. Secondly, the control method contains four sub-processes. The first is to define the set of policies for control operations like retention and rectification when using external elemental services. The second is to make several adaptation plans in order to adopt them sequentially to our target. The third is to evaluate every adaptation plan with a simulation within the process. The fourth is to decide the final actuating values. TOC was proposed by E.M. Goldratt [2], [3], and is concerned with rectification for production systems. Here, the bottleneck process must be identified. Then, all of the prior sub-processes before the bottleneck process should be rectified. Continuously, the next bottleneck should be singled out and repaired sequentially. Then the optimization of the targeted production system might eventually be accomplished. The reasons why we adopt TOC into our domain are as follows: Firstly, in order to maintain the QoS under the environment, we can only administrate the attached actuators based on the observable state values from outside. Furthermore, we are not authorized to make any outside elemental web services improved except the long term negotiation in SLM process. Secondly, under the above constraints our prioritized concern is how to ease the negative influence from disturbance of performance due to rectification before the bottleneck process.

In order to clarify the aforesaid method as our proposing framework, in this paper we will explain the reference models and theoretical aspects of the proposed method, which mainly consists of TOC, a set of policies for control operations and simulation. Our contributions with the novel proposed method are: (i) Establishing a theoretical backbone for the new controlling principles in the service computing domain. (ii) Defining a control framework, which has a simulation functionality and applies the policies under verification multiple times. Furthermore we also conclude that the simple feedback controlling approach under the TOC cannot function well.

The remainder of this paper is organized as follows. In Sect. 2, we describe the architectural and conceptual models. In the first two sub-sections of this section, we mention the characteristics of modeling with TOC and the effects of adopting TOC into composite web services. Then, we explain about the main body of the architectural and conceptual models. In Sect. 3, our mathematical control model is defined, and in Sect. 4 our configuration model is explained. In Sect. 5, we mention our evaluation of the proposed approach by comparing it with a simpler method which has been found insufficient. Furthermore, we also mention the potential possibility of the extension for general SLA items. We will then mention related works and make our conclusion in Sects. 6 and 7.

2. Architectural and Conceptual Model

2.1 Notion of Controlling by TOC

It is customary to adopt a queuing network for modeling the behavior of distributed processing, especially to evaluate non-functional requirements. However, there are some difficulties in modeling web services such as mash-up with queuing networks. For example, there are cases where a computer resource will immediately be assigned when a SOAP message is received. Thus, it might be difficult to model those cases only with queuing networks. In particular, there are also other possible cases where closed loops are formed. These might become obstacles for analysis if we look at the entire service network. Therefore, it might be reasonable to take the more simplified approach shown in Fig. 1, in which we will demonstrate the following: (1) Defining start and end points virtually in the closed network. (2) Translating the above process into another open and equivalent form topologically, that is, a sequential process without any loops and with an infinite number of servers, and (3) Defining the open multiple sequential steps. However, there might be a certain limitation.

Furthermore, based on the notions of the bottleneck in TOC [2], [3], it might be possible to collapse the process model in the queuing network model into the simplified equivalent collapsed model drawn in the middle of Fig. 1. However there are some constraints, where the properties of the entire process are fully equivalent with the collapsed model. Considerations in this paper are based on this hypothesis, though the additional matter of how adoptable the aforementioned translation is in fact, should be verified.

The procedure for the collapsed model roughly consists of the four steps. The first is to identify the bottleneck in the general queuing network model based on the TOC [2], [3]. The second is to collapse all of the processes located before the bottleneck into a single synthesized process. The third is also to collapse all of the processes located after the bottleneck into another single synthesized process. The fourth is to locate the queue that serves as the buffer just before the bottleneck.

In Fig. 1, for example, if the process-2-n(2) is the bottleneck, all of the following processes such as process-m-n(m) and queues are integrated into the equivalent process and mapped to ‘Process-Bottleneck & Beyond’, and all of the forward processes such as process-1-1 and queues are also integrated into another equivalent process, and then mapped to ‘Process-Prior’. Any processes which are not related to the bottleneck are treated as ‘collapsed’ in the ‘Process-Bottleneck & Beyond’. If there are multiple paths, and any processes are out of the critical path and not related to the bottleneck, they should be ignored. The queue located just before the bottleneck corresponds to the buffer before the bottleneck, which is explained in the TOC [3]. It has a finite length. The queue is located in order to maximize the throughput of the bottleneck process. This queue...
has the functionality of an actuator, which enforces the entry rejection if the length of the queue is sufficiently long.

The collapsed model has three features as follows: The first is the notion and meaning of the queue. It is totally different from those explained under general queuing network theory, which identifies them as growing naturally as a result of the unbalanced traffic situation between ‘in’ and ‘out’. The queues defined in the collapsed model, on the other hand, have the functionality of actuators for controlling, and work as buffers to store the entries until the processing mentioned entries occur at the bottleneck. The second is that another conceptual process named ‘Process-Reject’ is additionally defined. It treats an explicit occurrence of rejection before the bottleneck. This is caused by the functionality of the actuator. The third is that there are multiple candidates for the bottleneck at any time. The location will move with the passing of time. Although translation into the collapsed model has been made with reference to the bottleneck as the center of modeling, we need to define the notion of multiple queue actuators.

Several parameters are defined for the collapsed model. For example, ‘Expected Mean Number of service arrivals Per Unit Time for a composite web service (abbreviated EMNAPUT)’ is expressed with the symbol $\lambda$ ($\lambda \in \mathbb{R}$), which sometimes affects the entire system as a disturbance. Here $\mathbb{R}$ means the set of real numbers. ‘Specified Flow Rate (abbreviated SFR)’ is denoted by the symbol $c$ ($c \in \mathbb{R}$), which corresponds to an actuating value onto a queue actuator. ‘Mean value of the Turn Around Time (abbreviated MTAT)’ of a composite web service and ‘Variance of the Turn Around Time (abbreviated VTAT)’ are represented by symbols $R_{ave}$ ($R_{ave} \in \mathbb{R}$) and $\sigma^2(R)$. Finally, abortion rate (abbreviated AR) and throughput are expressed with the symbols $R_f$ ($R_f \in \mathbb{R}$) and $\rho$ ($\rho \in \mathbb{R}$).

The model of a queue-actuator is also shown at the bottom of Fig. 1. From the point of view of the system architecture, this should usually be implemented on proxies in front of all the WS-BPEL (OASIS Web Services Business Process Execution Language) processors and web services, and individually deployed for the WS-BPEL processors and the web services in peer-to-peer, or in grouping them. The queue-actuator extracts the SOAP messages as the service requests in FIFO. It monitors the number of stored SOAP messages and the ‘time of stay’ of individual messages. It also evaluates if the total time of stay is less than the specified maximum allowable time. Once an entry as a SOAP message exceeds the specified maximum allowable time, that entry will be aborted as a rejection. The queue-actuator has a limit with regard to the maximum of the number of allowable stored SOAP messages. Any new SOAP messages trying to be queued after the maximum has been reached are rejected. Furthermore, the queue-actuator has another property with regard to the SFR, $c$ ($c \in \mathbb{R}$), which is the maximum number of messages extractable from the queue-actuator within a unit time. By using this property, the control variable will be specified to satisfy the control conditions.

2.2 Evaluation on Adoption of TOC

In order to confirm the effect derived from adoption of TOC on composite web services, the experimental environment shown in Fig. 2 was implemented. Here, the composite web service consists of three WS-BPEL processors, several web services and queue-actuators which were deployed just before the processors and services. As for the typical machinery environment, we used a set of Windows 2003 Servers, which had 64 bits Intel Xeon Processor (2.80 GHz Dual Cores) and 12 GBytes main memory. Furthermore, we adopted the WS-BPEL processors as a NEC product with
Fig. 2  Experimental environment.

Oracle 10g for backyard databases, and Apache Servers for servlets. Here, the elemental Web Service3 was set as the fixed bottleneck service with fixed throughput. For the evaluation concerning the bottleneck, the following three cases were prepared as Case 1, 2 and 3, where the number of steps between the bottleneck and ‘set effective’ queue-actuator were variable. In Case 1, the queue actuator 1 was activated, whereas the queue actuators 2 and 3 were set with maximum flow capability. In Case 2, the queue actuator 2 was activated; however the queue actuators 1 and 3 were set with maximum flow capability. And finally in Case 3, queue actuator 3 was activated, and the other queue actuators were set with maximum flow capability. Processes and services prior to the bottleneck were regarded under the ‘Drum-Buffer-Rope’ satiation mentioned in TOC, in other words under rectification [3]. Here, what we evaluated was which queue-actuator as ‘set effective’ would really be effective for settling the entire system. Also, we demonstrate the most effective contribution to reduce the MTAT of the entire composite web service. Measurement of MTAT and Abortion Rate were executed by calculating the differences between the time at sending the business SOAP Msg-1 and that at receiving the business SOAP Msg-13.

The result of the evaluation is shown in Table 1. The top row of this table lists ratios between extracting performance from one of the queue-actuators and the average throughput at the bottleneck, which ranges from +2% to −6%. Here, +2% means over-loading the bottleneck, whereas −6% means relieving it by storing entries in the queue-actuator as much as possible. The value of each cell in the table expresses the ratio between the MTAT of the entire composite web service and the average processing time at the bottleneck. We can obviously confirm, as shown by the data in Table 1, that the most effective throughput for the entire composite web service is achieved when the queue-actuator is located just before the bottleneck and performs rectification. On the other hand, for example, under the condition with +2% for Case 3, the overloading which occurs at the bottleneck eventually causes performance to degrade for the entire composite web service, whereas under the condition with −6% for the same case, the staying period of an entry in the queue-actuator becomes longer. Thus, the MTAT of the entire composite web service tends to increase. As shown in Table 1, we can confirm the effect of TOC on the entire composite web service.

Table 1  Result of the TOC effects.

<table>
<thead>
<tr>
<th>Case</th>
<th>+2%</th>
<th>0%</th>
<th>−2%</th>
<th>−4%</th>
<th>−6%</th>
</tr>
</thead>
<tbody>
<tr>
<td>Case 1</td>
<td>45.9 times</td>
<td>28.4 times</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>Case 2</td>
<td>41.4 times</td>
<td>11.9 times</td>
<td>12.4 times</td>
<td>13.7 times</td>
<td>14.5 times</td>
</tr>
<tr>
<td>Case 3</td>
<td>42.8 times</td>
<td>11.4 times</td>
<td>12.6 times</td>
<td>13.3 times</td>
<td>14.2 times</td>
</tr>
</tbody>
</table>

2.3 Descriptive Model of the Architecture

Figure 3 shows the basic reference model for the configuration and architecture to realize the controlling function for the composite web services. The mathematical control model explained in Sect. 3 is based on this reference model. In Fig. 2, there is only one service client, with multiple sub composite web services running on each WE-BPEL processor, whereas the reference model is generalized; thus, we assume that multiple composite web services will be deployed.

The reference model consists of four abstract elements as follows: The first is multiple service clients, which correspond to users under contract. Individual SLA instance as a contract must obviously be defined for each service client. In the case of no definitions for SLA instance, we assume the default SLA, which contains values within the bounds of common sense. The second is multiple composite web services, the contracts for which must be maintained. The third is multiple elemental web services, each of which will be invoked by the multiple composite web services. The fourth is a monitor and a controlling function. As for the relationships between the multiple composite web services and the multiple elemental web services, these are statically defined in WS-BPEL descriptions. For this example, however, they are not permanently fixed. Furthermore, there are sufficiently possible cases where an elemental web service might be shared among multiple composite web services. In particular, each of the QoS attributes of the individual composite web services will be able to be computed by sets of QoS attributes, which the individual elemental web services have and are measured with, shown in [4] and [5]. Here non-linear behaviors are also assumed.

The model of a composite web service contains three abstract elements, the interface defined in WSDL (Web Service Definition Language), a varied interface for invoking
an elemental web service, and the interface to query the status of QoS attributes at any time, which must individually be maintained for each service client. The model of an elemental web service, on the other hand, also contains several abstract elements: firstly, the interface defined in WSDL; secondly, a queue-actuator to store the SOAP messages temporarily; thirdly, two interfaces to query the status of QoS attributes which are owned by and related to the individual elemental web service and the queue-actuator; and finally, the interface to specify values of QoS attributes like the SFR on the queue-actuator from outside. The actual implementation does not matter here; suffice to say, there are sufficient possibilities that an elemental web service might be implemented as a composite web service.

However, this reference model is not yet complete from the point of view of abstraction. As drawn in Fig. 3, the practical services tend to be defined with structured and layered composite web services. In particular, the reference model in Fig. 3 assumes that a bottleneck occurs only on the layer of elemental web services. However, the fact is that a WS-BPEL processor itself can also be a bottleneck. Thus, we need to consider extending the reference model to layered composite web services. In this case, we have considered that we need to apply several rules in order to make the evaluation procedures simple and the conclusions equivalent for practical cases. For example, multiple composite web services assigned to the subordinate layer should be merged into a synthesized web service. Moreover, any composite web services in the superior layer should treat that synthesized service as an elemental web service. Furthermore, if SLA properties for the composite web services assigned to the subordinate layer are specified, they should be treated as SLA properties of an elemental web service with only few influences. Accordingly, the assignment of queue-actuators must be specified on the above synthesis. However, the reference model within the layered composite web services gives certain impacts to the mathematical control model. Therefore, evaluation with a more formal approach might be required to verify the sufficiency of the previous rules, and this is a remaining issue.

2.4 The Principle of Controlling Composite Web Services

Based on the basic reference model, the outline of the proposed principle for controlling the composite web services is explained here. We assume the following:

(i) With regard to the composite web services, the controlled variables rely on aggregation of multiple elemental web services. However, there are several difficulties in the modeling of composite web services because of owned functions, for instance, synchronicity of several invocations, error handling and caused compensation activities. Thus, the controlled variables of composite web services might mostly be computed using non-linear methods. Hence, we need to assume the aspect of non-linearity.

(ii) In the basic reference model, our valid actuator is the queue-actuator, the operational methods of which are limited by manipulating throughput and abortion of messaging. Thus it is also difficult to work out actuating values by using linear control theory. We assume operations based on applying predefined policies for controlling.

(iii) There is an obvious limitation in naively adopting a feedback controlling method because the probability of violating SLA contracts might be higher than the prediction bases.

(iv) Values and scopes of QoS attributes and SLA properties rely on the negotiation process and its reached agreements. Therefore, not only common parameters like turn around time, but new attributes might be defined as well. However, the bottleneck explained in TOC is a bottleneck from the point of view of performance [3]. Therefore, we define our bottleneck as the
lowest potential point over the whole of the treated
SLA properties, which should have aspects catego-
rized in the performance domain. The generalized
case will be considered in Sect. 5.3.

Based on the above assumptions, the entire require-
ments for our controlling model may be outlined as follows:

(v) Applying the concept of digital control, certain ob-
servable variables are sampled, and targeted status
variables are estimated by a simulation function.
Also, the feed forward controlling method should be
adopted with some restrictions.

(vi) As the basic reference model has a layered structure,
we select use of a non-linear method when identify-
ing the controllable variables of composite web ser-
vices. Actuating values are decided by applying pre-
defined policies for controlling. Applying these poli-
cies is currently the generalized and proven available
method. (For example, refer to [26].) Thus, it is nec-
 essary to verify the appropriateness of applying them
as an in-process of controlling. Here, we propose
a new approach, which is the most contributory to
solving our challenges. If a variable $p$ is defined as the
maximum allowable trial number to apply the policies
for identifying actuating values during every sampling
interval, then simulation processes will occur within $2p$
times as follows: During the first half with $p$ times,
simulation is done for estimating the targeted status
variables of the composite web services at the corre-
sponding time in the second half. During the second
half with $p$ times, simulation is done for estimating
the controlled status variables of the composite web
services with temporal adoptions with the set of some
policies as in the first half.

(vii) The simulation process for verification might be time
consuming. Thus, compensations for time delay in
calculation must be considered.

(viii) In order to maintain the precision of estimation, an
adjustment regulation process, which is outside the
controlling process, should be implemented. In the
process, comparison and related calculation between
estimation values and real measured values of the
QoS attributes and SLA properties must be done, and
a result of the adjustment must be applied as other
compensations.

Based on the above requirements (i) to (viii), we can
adopt the architecture model drawn in Fig. 4. This model
consists of five sets of functional modules as follows, Mea-
surement, Status Observer, Regulator, Controller, and Ad-
justment. The control target is as per the basic reference
model in Fig. 3.

The first set, as Measurement, consists of several func-
tional modules such as the Statistic Data Processor, the Meta
Data Management, the Measurement Raw Data Manage-
ment, and the Performance Data Management. This Mea-
surement functions for aggregating and managing data. The
Statistic Data Processor is a functional module to calcu-
late values of the parameters yielded by the simple statistics
without using any Simulation Processors. The Meta
Data Management manages definitions data with regard to
the composite web services and the elemental web services.
The Measurement Raw Data Management organizationally
manages the individual event raw data according to the data
at the Meta Data Management. The Performance Data Man-
agement manages the aforementioned values of the param-
eters to use in subsequent processes.

The second set, as Status Observer, contains several func-
tional modules like the Interval Controller and the main
body of the Status Observer. The Interval Controller man-
ages the cycle time of sampling, whereas the main body of
the Status Observer will mainly carry out the follow-
ing four activities: (1) obtaining several elemental values
from the aforementioned Performance Data Management
for subsequent processes, (2) yielding the vector data of the current status of the measured elemental web services at the current sampling time, (3) setting on the vector data yielded at the most recent sampling time as a historical status, and (4) storing the vector data of both statuses.

The third set, as Regulator, is the heart of our architecture model, and consists of several functional modules like the Simulation Processor with Compensation, the Metrics Values’ Reference, the Comparator, the Condition Provider, and the Policy List. The Policy List is a functional module to manage the predefined policies for controlling underlinkages with actuating values and conditions. The Metrics Values’ Reference is a functional module to hold the vector of the aimed values, which are defined in the contracts with service clients as the QoS attributes and SLA properties. The Simulation Processor with Compensation is a functional module to carry out simulation processes under the condition specified by the aforesaid vector data of the current status, the historical status, the target values, and temporal actuating values obtained through the predefined policies for controlling. The Comparator is the functional module to obtain the controlled deviation by comparing the results from the Simulation Processor with Compensation with the target values. The Condition Provider is the functional module to decide the order of adoption of the predefined policies for controlling and temporally to adopt actuating values in the policies for the simulation process.

The fourth set, as Controller, consists of several functional modules like the Controller, Candidates of Control Value Vector, the input Control value Vector and a set of the Actuators. In this case, there are several Actuators depicted in Fig. 2, therefore the input control values which are yielded by merging the actuating values of the adopted policies, thereby compose a vector and are selected from the Candidates of Control Value Vector. Once the controller fires the actuation process, these values input as the vector will be provided to the set of the actuators.

The fifth set, as Adjustment, contains several functional modules like the Metrics Processor and the Adjustment Regulator. This function appropriately maintains the predefined policies for controlling and for other materials which are referred to in simulations in order to maintain the quality and accuracy of processing by the Simulation Processor with Compensation.

When operating this architecture, the three loops will be executed as following: the first is the main and mandatory loop, named as the control loop, which passes the Control target, the Measurement, the Status Observer, the Regulator and the Controller. The second loop is the estimation loop, implemented as a sub loop of the first one. In this loop the evaluation process by using the Simulation Processor with Compensation is executed. And the final one is the loop for adjustment which will be carried out by the Adjustment. This is basically an independent loop from the first control loop.

3. Mathematical Model

3.1 Definition of the Conceptual Process Model

Figure 5 shows an abstract temporal behavior model of a QoS attribute. The horizontal axis means time progress, whereas the vertical axis means the value of the selected QoS attribute in scalar at the specified time.

Here, the constant \( T \) is defined as the cycle time (interval) of sampling. And the constant \( T_{e}^{(ave)} \) is defined as the mean time of measurement and evaluation period. It means the average of total time cost to obtain the final candidate of the actuating values by using the Simulation Processor with Compensation. The constant \( PC \) means the probable maximum period for the controlling operation. Following (1) is the sampling time for \( q \) times, if the initial time is \( t_0 \). The constant \( T_{o}^{(max)} \) means maximum allowable time of operation. It must be more than the sum of the twice of \( T_{e}^{(ave)} \) and \( PC \) in (2) because of ‘almost’ safe controlling.

\[
t(q) = t_0 + Tq. \quad (1)
\]

\[
PC + 2T_{e}^{(ave)} \leq T_{o}^{(max)}. \quad (2)
\]

Figure 5 also includes the outline of the simulation processes, which occur within \( 2p \) times as mentioned above. Thus the mean time of measurement and evaluation period \( T_{e}^{(ave)} \) is divided by twice the maximum allowable trial number to apply the policies \( p \) under a constraint in (3), if \( T_{e}^{(ave)} \) is defined as the mean time of the calculation at the Simulation Processor with Compensation. This (3) means that \( T_{e}^{(ave)} \) has to be specified with a sufficiently safe period, compared with the possible maximum of the total calculation time.

\[
2p \ast \left(2T_{e}^{(ave)}\right) \leq T_{e}^{(ave)}. \quad (3)
\]

3.2 Definition of the Services

Here, the mathematical definitions of an elemental web service and a composite web service are given. An elemental web service in Fig. 3 corresponds to an element of a set, which consists of all of the available elemental web services by any composite web services as following (4).

\[
\exists s \in S. \quad (4)
\]

As a possible expression after decomposition, any transactional element which is a part of a composite web service, could be specified as the tuple expression in (5), if \( CONT \) means a set of contexts to distinguish between the equivalent service invocations, \( S_o \) means the set of observable elemental web services in a particular origin side including service clients, \( S_d \) means the set of elemental web services in a particular destination side under the condition of \( S \cup SC = S_o \cup S_d \). Here, set \( SC \) means the set of service clients. Set \( N_e \) means the total number of loops in integer equal to or larger than zero, and set \( N_l \) means numbers of
an indicator in every loop and in integers equal to or larger than zero. Accordingly, the definition of any composite web service can be expressed in (6) by using the power set of tuple (5). Moreover, when considering \( N \) as a finite integer set, a naming function which has the aspect of one-to-one mapping in (7) is definable.

\[
\exists_{\text{etp}}: \{\text{etp} \in ETP \subseteq (\text{CONT} \times S_a \times S_d \times N_i \times N_t)\}. \quad (5)
\]

\[
\exists_{\text{cws}}: \{\text{cws} \in CWS \subseteq 2^{ETP}\}.
\]

\[
\text{name}: \{\exists_{\text{cws}} \exists_{\text{etp}} (m \in N \land \text{cws} \in \text{CWS}): \text{cws} \iff m\}. \quad (7)
\]

It is assumed that the granularity of the composite web service defined by (7) corresponds to that in Fig. 3. All QoS attributes and SLA properties which at least one elemental web service has, are identifiable as elements of a set \( I \), which consists of variable scalar parameters that are observable at any time. Thus (8) is definable for all elements in the set \( S \). Here, sets \( R, R^* \) mean the set of real numbers, and the set of real numbers without those less than zero. Whenever an elemental web service has no relationships with an element in the set \( I \), the corresponding values in (8) always become zero.

\[
\forall t(t \in R^*, t_0 \leq t) \forall i(i \in I) \forall s(s \in S): Y_{(i,s)}(t) \in R. \quad (8)
\]

All QoS attributes and SLA properties which at least one composite web service in the set \( \text{CWS} \) has, are identifiable as elements of a set \( J \), which consists of variable scalar parameters that are yielded through non-linear simulation processes with \( Y_{(i,s)}(t) \) at any time. Thus (9) is definable for all elements in the set \( \text{CWS} \). Whenever a composite web service has no relationships with an element in the set \( J \), the corresponding values in (9) also become zero.

\[
\forall t(t \in R^*, t_0 \leq t) \forall j(j \in J) \forall m(m \in N \land \text{name}^{-1}(m) \in \text{CWS}): Y_{(j,m)}(t) \in R. \quad (9)
\]

According to (9), regardless of real adoption of any policies, an estimated value as a result of the simulation process is expressible as (10).

\[
\forall t(t \in R^*, t_0 \leq t) \forall t'(t', t_0 \leq t, t') \forall j(j \in J) \forall m(m \in N \land \text{name}^{-1}(m) \in \text{CWS}): Y_{(j,m)}(t + t') \in R. \quad (10)
\]

According to Fig. 3, the contracted SLA properties and QoS attributes with each service client are expressible in (11). Here, it is assumed that all of the properties might be constant, thus in the practice phase, (11) becomes the aimed values for controlling.

\[
\forall j(j \in J) \forall m(m \in N \land \text{name}^{-1}(m) \in \text{CWS}): Y_{\text{Const}}(j,m)(t + t') \in R^+. \quad (11)
\]

Finally, the constraints on (9) should to be clarified. Here, we would regard a control system as settled on the safe side, if (12) were satisfied. Furthermore, we would regard the control system improved, if (13) were satisfied. And we would also regard the system as ‘Not necessary to be operated’, if both of (12) and (14) were satisfied. Naturally, all of the elements of the set \( J \) must be expressed to satisfy (11) (12) (13) and (14).

\[
\forall t(t \in R^*, t_0 \leq t) \forall j(j \in J) \forall m(m \in N \land \text{name}^{-1}(m) \in \text{CWS}): Y_{(j,m)}(t) > Y_{\text{Const}}(j,m). \quad (12)
\]

\[
\forall t(t \in R^*, t_0 \leq t, t') \forall j(j \in J) \forall m(m \in N \land \text{name}^{-1}(m) \in \text{CWS}): \forall t'(t', t_0 \leq t, t') \forall j(j \in J) \forall m(m \in N \land \text{name}^{-1}(m) \in \text{CWS}) \exists d(d \in R^+): |Y_{(j,m)}(t) - Y_{\text{Const}}(j,m)| \leq d. \quad (14)
\]

### 3.3 Outline of Procedure of Calculation

Figure 6 depicts the outline of the procedure of the calculation based on the aforesaid set of formulas. The explanation of the main stream of the calculation will be explained in Sect.3.6. And the important concepts as a background beyond the individual steps will be mentioned in the next two Sections.
3.4 Method to Identify a Bottleneck

As mentioned before, a generalized bottleneck is defined as the lowest potential point over the whole of the targeted space for SLA properties, especially in the performance. In particular, it is assumed that a real bottleneck would be identified by observing a phenomenon and carrying out a root cause analysis process. Based on this, in the process for identifying a bottleneck, the appropriate reverse procedure against the non-linear simulation process, yielding estimation in (9) from values in (8), might be required. However, it might be difficult to identify it by computation with the inverse function, because of few possibilities of the existence of the inverse function, and certain possibilities about irreversibility caused by the non-linear simulation. Thus, we need to consider a new method for identifying a bottleneck, for example, adopting policies generated in the system identification process. In our method, any bottleneck is identified through two phases as follows: The first phase is identifying the targeted point and it is related to only the composite web services. This phase corresponds to step.2 and step.3 in Fig. 6. According to TOC, at first the bottleneck should be identified [2].

Our method generalizes the concept of finding a bottleneck. The first phase is mandatory and furthermore roughly consists of two conceptual elements. The first is defined as identifying a deviation against the aimed value at any time \( t \). It is obviously related to the service having a large controlled deviation against the required and contracted SLA properties of the composite web service. The second is to specify priorities managed by service providers. Regarding the first conceptual element, the related formalization will be explained in the next section, whereas the second is definable in (15) (16). (15) means the assignment of suitable number as the priority to a composite web service, whereas (16) means the assignment of suitable number as the priority to a contracted SLA property of the composite web service. Zero as values refers to the lowest priority, here.

\[ k \in \mathbb{R} \land (0 \leq k \leq 1); \quad pt(m) \Rightarrow k. \]  
(15)

\[ m \in \mathbb{N} \land name^{-1}(m) \in CWS \land j \in J \]  
(16)

The second phase is to pinpoint the candidates of a bottleneck specifically and identifying the current situation by using the outputs of the first phase. This corresponds to step.4 and step.5 in Fig. 6. But results of the second phase remain as just candidates. Here, it should be assumed to adopt temporally selected policies generated in the system identification process. Adopting policies for verification corresponds to step.6 and step.7 in Fig. 6.

The system identification process is usually carried out before the practice of controlling. As the Simulation Processor with Compensation is used for the simulation process, it is also usable for the system identification process in order to identify the relationships between the values of (8) and the estimation values of (9) as a preparation. However, regardless of the degree, the calibration and the adjustment with measured data should also naturally be included for verification. Here, after setting suitable predefined values into variables of the QoS attributes in (8) as inputs, simulation will be executed. Then, outputs as results of the system identification process will be yielded. In this case, input data are elements of the set \( G_e \), the type of which is expressed in \( |J| \times |S| \) type matrices, if the number of elements in the set \( S \) is expressed as symbol \( |S| \). The set \( G_e \) is a finite countable set due to dependency on the number of invocations of a simulation process, despite the real number data of SLA properties.

The results of the simulation identification processes are elements of the set \( G_e \), the type of which is expressed in \( |J| \times |CWS| \) type matrices. The set \( G_e \) is also a finite countable set. Thus, it might be possible to pre-aggregate the data, features of which are specified in (17). The meaning of (17) is that the results of the non-linear simulation process will be identifiable when specifying input values. Therefore, if the existence of an inverse mapping with a multi-valued function were assumed, (19) might be definable based on (17) and (18). The function \( g(m) \) in (19) means identifying an equivalent element of the power set yielded by the merged set of the empty set and the set \( G_e \). This means that there are certain possibilities to identify the multiple candidates of a bottleneck, including difficult cases, when SLA properties of any composite web services are estimated with a set of values. The empty set corresponds to impossible cases to be identified. The functions in (17) and (19) should be predefined through the system identification process before the practice of controlling, and should also be implemented as a policies’ library or instances of case-based reasoning.

\[ m \in \mathbb{N} \land name^{-1}(m) \in CWS \land j \in J \]  
(17)

\[ G = \{ \emptyset \} \cup G_e. \]  
(18)

\[ m \in \mathbb{N} \land name^{-1}(m) \in CWS \land j \in J \]  
(19)
3.5 Formalization of Predictions

The deviation for aimed values at any time \( t \) is defined as a controlled deviation. This is yielded in (9) and (11) and expressed in matrices as in (20). Although (20) is expressed by using a simplified subtraction, and available for identifying the target point roughly at step.3 in Fig.6, the controlled deviation should rely on the characteristic matters of the controlling system and designing policies. Thus, there are obvious possibilities to be expressed in other formulas. In order to yield the controlled deviation, multiple properties might be calculated through the simulation process. For example, in the case of performance including the turn around time, the method with Timed Petri Net has been proposed in [6]. When evaluating SLA properties of all composite web services at any time in order to identify the candidates of a bottleneck, (21) should be yielded based on (15) and (16). (21) is expressed as a vector with the number of composite web services with treating the priorities, and all of the elements of it have values in real numbers equal to zero or larger than zero. As all of the elements more than satisfy the condition derived from (14), the ratio derived from (21) will be more similar with one specified by (15).

To carry out step.4 in Fig. 6, we firstly need to seek the elements \( G_c\{m\} \) of the set \( G_c \), which correspond to the situations that have similarity with one yielded by (21) at the time. Then, by (19) we also identify the subset of the set \( G_c \), the elements of which lead the aforementioned situation at the time. An elemental web service related to the identified element of subset of the set \( G_c \) might probably be related to a service corresponding to the bottleneck. In order to seek them, the following procedure is concretely done: Firstly, elements \( G_c\{m\} \) as inputs for (19) are sequentially extracted from set \( G_c \), then the equivalent calculations with (21) are done by using extracted \( G_c\{m\} \) and obtaining output results. Secondly, calculations with (21) are executed and \( CWB(t) \) at the time \( t \) is independently yielded, then comparison between the previous output results and \( CWB(t) \) is individually done by using the inner product.

If the scalar value yielded by the inner product is larger than a threshold value specified previously, there are certain possibilities that \( G_c\{m\} \) might be related to the current situation. Furthermore, by (19) we also identify the aforementioned subset of the set \( G_c \), as depicted as step.5 in Fig. 6. An elemental web service related to the identified element of the subset might be the service related to the bottleneck, although it is not always strictly identified. However, due to narrowing the scope of the probable candidates as the bottleneck, it might be assumable to adopt some policies by which some services would tentatively be prioritized as the bottleneck. This assumption should naturally be verified by adopting a simulation process at step.7 in Fig. 6. If the result of the verification could not satisfy the specified conditions and more trials for seeking would be allowable, another prioritized service to be manipulated would be listed. Therefore, based on the resulting list, it might be possible to identify the elemental web services to be manipulated. Accordingly, based on the above tentative identification, the mentioned Condition Provider would specify acting values and conditions managed as predefined policies for controlling.

\[
DV_{(i,m)} = \begin{bmatrix} Y_{(1,1)}(t) - Y_{\text{Const}}(1,1) \frac{1}{Y_{\text{Const}}(1,1)} & \cdots & Y_{(1,m)}(t) - Y_{\text{Const}}(1,m) \frac{1}{Y_{\text{Const}}(1,m)} \\
\vdots & \ddots & \vdots \\
Y_{(j,1)}(t) - Y_{\text{Const}}(j,1) \frac{1}{Y_{\text{Const}}(j,1)} & \cdots & Y_{(j,m)}(t) - Y_{\text{Const}}(j,m) \frac{1}{Y_{\text{Const}}(j,m)} 
\end{bmatrix}
\]

(20)

\[
CWB(t) = \begin{bmatrix} pl_{(1)} \sum_{j=1}^{x} \left( \frac{pp_{(x,1)}}{Y_{\text{Const}}(x,1)} \right) Y_{(1,1)}(t) \\
\vdots \\
p_{(m)} \sum_{j=1}^{x} \left( \frac{pp_{(x,m)}}{Y_{\text{Const}}(x,m)} \right) Y_{(1,m)}(t) 
\end{bmatrix}
\]

(21)

3.6 Formalization Related to the Procedure

In order to decide the controllable variables of composite web services based on the predefined policies, verification on the appropriateness will be carried out in multiple trial times. If the variable \( p \) is defined as the maximum allowable trial number to apply the policies for identifying actuating values at every sampling interval, simulation processes will occur within \( 2p \) times as follows: During the first half with \( p \) times, simulation processes are carried out for estimating the targeted status variables specified in (9) at the corresponding time in the second half. This set of \( p \) times’ invocations corresponds to step.1 in Fig. 6. If a variable \( r \) \( (1 < r < p) \) is given, the estimated SLA properties expressed in (23) are yielded at the time specified in (22). In this case, as the values in the near future are treated, compensating by linear extrapolation with values of (8) at both of times \( t = t_0 + T(q - 1) \), \( t = t_0 + Tq \) should be used.

During the second half with \( p \) times, the following procedures are done: at first, based on the estimated SLA properties specified by (23), the controlled deviation expressed in (20) is yielded. After evaluation with (12), (14) and (20), only if any operation is required, the Condition Provider will continuously identify the current status and evaluate the bottleneck by using (19) and (21). These correspond to step.4 and step.5 in Fig. 6 and are carried out at the time in (24). Then, the Condition Provider extracts actuating values and conditions managed as predefined policies for controlling. After that, the Condition Provider temporarily adopts them. The Simulation Processor with Compensation will predict values specified in (25). Continuously, the estimated controlled deviation in (26) which corresponds to (20) is yielded. These correspond to step.6 and step.7 in Fig. 6.

\[
t = t_0 + Tq + \frac{r}{2p} T^{(\text{ave})}.
\]

(22)
\[
Y_{(j,m)}(t_0 + T q + \frac{(p + r) T_{e(\text{ave})}}{2p}) \in \mathbb{R}.
\]
(23)

\[
t = t_0 + T q + \frac{(p + r) T_{e(\text{ave})}}{2p}.
\]
(24)

\[
Y_{C,(j,m)}(t_0 + T q + \frac{(p + r) T_{e(\text{ave})}}{2p}) \in \mathbb{R}.
\]
(25)

\[
\frac{Dv_C^{(j,m)}}{\text{ave}}.
\]
(26)

If the estimated controlled deviation in (26) can satisfy the condition specified in (12) and (14), controlling shifts into the practice phase with the final acquired actuating values. Accordingly, the selected queue-actuators will be manipulated. Otherwise, if the estimated controlled deviation does not satisfy the condition, the next trial in the \((r + 1)\) times will be done at the time specified by (24) by replacing \(r\) with \((r + 1)\). Here, the Condition Provider extracts other actuating values, and enhances the previous actuating values by merging with the newly extracted one. Then the Condition Provider adopts the enhanced one again.

4. Configuration Model

Implementing a verification system has not been completed yet. However, we can show the configuration model as a basic design of the architecture in order to realize our proposal. The outline of the configuration model is shown in Fig. 7. It reflects to Fig. 6 and roughly contains five parts as follows. The first is ‘Sampling & Measurement’ corresponding to the Measurement in Fig. 4. The second is the part that includes the process called ‘Yielding Status Variables’, which corresponds to invoking the Simulation Processor with Compensation in Fig. 4 in the first \(p\) times. As mentioned, the status variables will be generated through the simulation process, and the results are temporally stored for the following \(p\) times’ invocations. The third is the biggest part, which includes the processes, for example the duplicated processes of ‘Comparing with SLA’, ‘Controlling Action Plan’ and ‘Predicting Manipulated Status’. This corresponds to the series of the processes of the estimation loop explained in Sect. 2.4. SLA is stored in SLA Management, as shown in Fig. 4. Formulas (19) and (21) are the basis for the process of ‘Seeking Similar Situations’ and by adopting the inner product, a suitable situation will be identified. The fourth part is ‘Carrying out Actions’ corresponding to the Controller in Fig. 4. The fifth part is the combination between ‘System Identification’ and ‘Transforming’ shown at the bottom of Fig. 7. This part is carried out in offline processes by executing simulation, and the results expressed in (19) are implemented as the policies’ library and they will be applied in the Regulator in Fig. 4.

5. Evaluation and Consideration

As mentioned, implementing this architecture to verify its effectiveness has not been completed yet. Therefore, it might be preferable as an evaluation to consider the possibility of this approach by comparing it with another approach, which was concluded with a negative result through our experiments. Here, the following three points will be discussed: The first is the appropriateness for adoption in practice; the second is about the granularity of the model; the third is about possibility to make our approach generalized to other areas.

5.1 Unsuitability of the Feedback Controlling

We had already implemented the feedback controlling module for the environment shown in Fig. 2. It was different from the aforementioned method in the previous sections. It evaluated its controllability in the feedback controlling method. However, our evaluation with the adoption of the feedback controlling method has proved to give a negative result. Figure 8 shows the outline of the procedure of the feedback controlling method.

In this procedure, step.1 and step.2 are executed in the preparation phase before practicing the feedback controlling. At step.1 the system identification processes for all of the aforesaid queue-actuators should be carried out. In these processes, we statically measure the MTAT of a composite
web service \( R_{\text{av}} (R_{\text{av}} \in R) \), the VTAT \( \sigma^2(R) \), and the AR \( R_j \) \((R_j \in R) \) under the fixed values of three parameters, which are (i) EMNAPUT for a composite web service \( \lambda (\lambda \in R) \), (ii) \( \text{SFR} \) \((c \in R) \) and (iii) throughput of the specified elemental web service as the bottleneck \( \rho (\rho \in R) \). Then we identify the effects on the previous MTAT of a composite web service \( R_{\text{av}} \), the VTAT \( \sigma^2(R) \), and the AR \( R_j \). Here, we intend to treat them as steady-state values without any transient characteristics. In the initial experiment, we evaluated the targeted system with only one composite web service and no variation of the throughput of the specified elemental web service as the bottleneck \( \rho \), and identified the characteristics of that targeted system by each queue-actuator with an evaluation function \( E_1(t) \) given in (27). The values of this function are yielded with the MTAT of the specified composite web service \( R_{\text{av}} \), the VTAT \( \sigma^2(R) \), and the AR of it \( R_j \) \((t_0 + Tq) \) at the current interval time \( t_0 + Tq \). We also assume the steady-state characteristics of this evaluation function.

\[
E_1(t) = f(R_{\text{av}}(c, \lambda), R_j(c, \lambda)).
\]  

(27)

Continuously the initial values of specified flow rate \( c(t_0) \) of all queue-actuators are set at the initial time \( t_0 \) in step.2. As mentioned before, constant \( T \) is defined as the cycle time (interval) of sampling, and we assume the situation at sampling time for \( q (q \in N^+) \) times. At step.3, the EMNAPUT for the specified composite web service \( \lambda(t_0 + Tq) \) at the split time \( t_0 + Tq \) is measured, and the MTAT of that composite web service \( R_{\text{av}}(t_0 + Tq) \), the AR of it being \( R_j(t_0 + Tq) \), are measured as well. Then at step.4, the value of the evaluation function \( E_1(t_0 + Tq) \) for the specified composite web service at the split time \( t_0 + Tq \) is calculated according to the results at step.3. Then at step.5, the location of the bottleneck is sought and identified. And at step.6, the ideal value of the SFR \( c(t_0 + Tq) \) in front of the bottleneck service at the split time \( t_0 + Tq \) is yielded through (27) with the results of the system identification processes practiced at step.1. This is concretely carried out with the procedure specified in Fig.9. And the yielded results are set at the queue-actuator just before the bottleneck service at step.7, whereupon waiting for the next invocation at the next split time \( t + T(q + 1) \) is done at step.8, and finally the process returns to step.3.

Figure 9 shows a simplified model of the results which are gained through the system identification processes on a specified queue-actuator. And this model has a curved surface formed by the evaluation function \( E_1(t) \) given in (27), where the SFR \( c(t) \) of the specified queue-actuator and the EMNAPUT for specified composite web service \( \lambda(t) \) are variables at the same time. Here, we regard that the evaluation function \( E_1(t) \) has three characteristics as follows: the first is to have the steady-state characteristics without any transient ones, as mentioned before. The second is to have larger values than zero. And the third is that it is more preferable as the values of this function approach zero. Here, we assume there is a change with regard to the EMNAPUT for the specified composite web service \( \lambda \) between the sampling intervals at \( t_0 + Tq \) and \( t_0 + T(q + 1) \). In this case, this change is expressed as shifting along the arrow from point C to another point D as in Fig.9. Thus we could assume that point A on the curved surface might be ideal due to the evaluation function \( E_1(t) \), because the current valid value of the SFR \( c(t) \) of the queue actuator in front of the bottleneck is specified at the previous interval time \( t_0 + T(q - 1) \). On the other hand, we also assume that the real value of the evaluation function \( E_1(t) \) might be smaller, and corresponds to point B on the curved surface. In this case, there is a certain difference between the value of \( E_1(t) \) at point A and the real value at point B. And a certain value of the SFR \( c(t_0 + Tq) \), which yields the value of the evaluation function \( E_1(t_0 + Tq) \) at point B, definitely exists. This SFR \( c(t_0 + Tq) \) will be plotted on the curved line formed by the evaluation function \( E_1(t) \) through fixing the EMNAPUT for the specified composite web service \( \lambda \). Thus, it is possible to select a more suitable and reasonable value of the SFR \( c(t_0 + Tq) \) as an actuating value.

Figure 10 shows the results of the specified evaluation function in the system identification process. The environment for the evaluation was completely the same as that ex-

**Fig. 9** Principle of deciding the actuating value in the initial method.

**Fig. 10** Practical evaluation function and behavioral characteristics.
plained in Sect. 2.2. Thus, our measurement was done under the environment, where the Elemental Web Service 3 was set as the bottleneck with the fixed value of the throughput \( \rho \), and variables, the EMNAPUT for the entire composite web service \( \lambda \), and the SFR \( c \) of the front queue-actuator were statically changed. The evaluation function was specified in (28), which had the steady-state characteristics.

\[
E_1(t) = R_{ave}(t) \cdot \left(1 + 10^3 R_j(t)\right).
\]  

(28)

Since it is preferable to have the values of this function as close to zero as possible, the lower MTAT \( R_{ave}(t) \) of the entire composite web service indicates more improvement in (28), whereas a larger \( AR_j \) of it is more degraded. Coefficient \( 10^3 \) means an accommodating parameter that would enhance the sensitivity of the evaluation function. In Fig. 10, scales of the axes of the horizontal plane are inverses of the parameters. The axis on the left side means the inverse of the EMNAPUT for the entire composite web service \( \lambda \) and the expected mean number increases more as it shifts to the left more. And this axis corresponds to the uncontrollable value. On the other hand, the axis on the right side means the inverse of the SFR \( c(t) \) and controllable. And the number of the arrived service requests, which are extracted from the queue-actuator and sent to elemental web service 3 as the bottleneck, increases more as it shifts to the right more. In Fig. 10, the area around point A has the worst \( AR_j \), because as the frequency of violating the allowable time of the queue-actuator to stay becomes higher due to shrinking capacity against the arrived service requests, accordingly the abortion rate also becomes higher.

As shifting takes place from point C to point D in Fig. 10 during \( T \), the EMNAPUT for the entire composite web service \( \lambda(t) \) increases, accordingly the value of the evaluation function \( E_1(t) \) deteriorates. This is caused by increasing the MTAT of the entire composite web service \( R_{ave}(t) \) during \( T \). On the other hand, as shifting takes place from point G to point H in Fig. 10 by setting the SFR \( c(t) \) larger in the scope of the small values of the EMNAPUT \( \lambda(t) \), the \( AR_j \) is improved; accordingly the value of the evaluation function \( E_1(t) \) is also improved. In particular, the area around point B has one of the best \( AR_j \), however it is substantially the pseudo-best due to forcing the arrived service requests to be sent into the bottleneck in order to avoid any rejection. And as shifting takes place from point E to point F in Fig. 10 by setting the SFR \( c(t) \) smaller than equivalent values corresponding to the fixed value of the throughput \( \rho(t) \), accordingly the value of the evaluation function \( E_1(t) \) deteriorates. This is caused by decreasing the throughput of extracting arrived service requests from the queue-actuator, accordingly increasing the contribution from the MTAT of the entire composite web service \( R_{ave}(t) \). Obviously, the feature of the evaluation function \( E_1(t) \) is largely changed around the particular value of the SFR \( c(t) \) corresponding to the fixed value of the throughput \( \rho(t) \). In the scope of the smaller values of the SFR \( c(t) \) than equivalent ones corresponding to the fixed value of the throughput \( \rho(t) \), the formed curved surface is relatively smooth, whereas in the scope of the larger side we can find a strong convex.

We tried several variable forms of the evaluation function in order to get a smoother surface; however, all of the forms equally had the same aspect on their shapes. This is caused by forcing the arrived service requests to be sent into the bottleneck in order to avoid any rejection. In this case, all of the arrived service requests are accepted, and the new threads are invoked. However, their performance deteriorates, and accordingly the substantial throughput itself also deteriorates. Thus, the MTAT of the entire composite web service \( R_{ave}(t) \) increases. If we adopt this function having the shape shown in Fig. 10 for feedback controlling, selectable SFR \( c(t) \) will become ‘super’ discrete, and controlled behavior will also become discrete, without stability and smoothness.

As shown in Fig. 11, if the controlling is carried out within the smaller values of the SFR \( c(t) \) than the values corresponding to the throughput \( \rho(t) \) at the bottleneck (left side of the line formed from point I to J), the changes of actuating value remain small around point K, as long as the changes of the EMNAPUT \( \lambda(t) \) remain small. However, once the EMNAPUT \( \lambda(t) \) changes considerably, and shifts to point L, it is possible that a certain large value of the SFR \( c(t) \) as the controlled value is selected, that is plotted as point M. Therefore, it is possibly expected to minimize the \( AR_j \) temporally by forcing all of the arrived service requests to be sent into the bottleneck. This means that the controlling policy here prioritizes the bad throughput rather than poor quality of services caused by abortions. According to the features of the evaluation function \( E_1(t) \), it looks correct. Then, if the EMNAPUT \( \lambda(t) \) changes and the resulting value of the evaluation function \( E_1(t) \) is plotted around point O on the convex shape, it is required to reset the SFR \( c(t) \). In this case the value corresponding to point P will be selected, where it is located over the convex shape of the evaluation function \( E_1(t) \). Here, the number of the arrived service requests staying as the entire composite web service decreases by shrinking the number of the arrived service re-

![Fig. 11 Behavioral characteristics under the feedback controlling.](image-url)
quests to be sent into the bottleneck and deteriorating the AR $R_j(t)$. Therefore, the behaviors of the entire composite web services become unstable due to the behaviors of the queue-actuators.

Furthermore, as shown in Fig.10 obviously, we can find out the strong dependency of the feature of the evaluation function $E_1(t)$ on the value of the throughput $\rho$ at the bottleneck, thus we need to take it into account when carrying out the system identification processes. Concretely, we need to adopt another evaluation function $E_2(t)$ given in (29), which consists of the three elemental parameters. We also assume that (29) has steady-state characteristics.

\[ \forall t \exists i_0(t, i_0 \in R; i_0 \leq t); E_2(t) = f(R_{ave}(c, \lambda), R_j(c, \lambda), \rho). \]  

(29)

Figure 12 shows the outline of the improved procedure of the feedback controlling method, which explicitly treats the dependency on the throughput $\rho$ at the bottleneck. Compared with the original procedure defined in Fig.8, steps 1, 5 and 6 are enhanced. In particular, we need to regard individual throughput with a reasonable value, because it is difficult to define the individual throughput of the elemental web service corresponding to each queue-actuator strictly.

However, some issues might be caused because of the adoption of (29). The first is that we have three duty controlled variables in real numbers as follows, when carrying out the system identification processes: (i) EMNAPUT for a composite web service $\lambda (\lambda \in R)$, (ii) SFR $c$ ($c \in R$) and (iii) throughput of the specified elemental web service as the bottleneck $\rho (\rho \in R)$. Uncontrollable variables which decide the status of the entire composite web service, (i) EMNAPUT for a composite web service $\lambda (\lambda \in R)$ and (ii) throughput of the specified elemental web service as the bottleneck $\rho (\rho \in R)$, take values in real numbers, thus the amount of tasks are truly huge in the system identification processes. As it is really difficult to carry out the completed tasks of these processes, it might be practical to identify the invisible real throughput $\rho(t)$ as following: (i) selecting suitable scope with real specified values of the throughput $\rho_1, \rho_2$ in the system identification processes, which are expressed in (30), and (ii) interpolation with them.

\[ \forall \xi_1, \xi_2, \rho_1, \rho_2 \in \text{Set of Evaluated Throughput Data} \]

\[ \rho_1 = \max(\xi_1 \leq \rho(t)), \rho_2 = \min(\rho(t) \leq \xi_2). \]  

(30)

However, we have not proved the appropriateness of this approach because we had not been able to confirm the linear characteristics in the results of the identification for the interpolation. Therefore, we concluded that the feedback controlling approach in this way might comprehensively not be appropriate.

In order to solve the negative result, we have developed aforementioned method combined with the simulation processes. However, our method still requires the system identification process for implementing policies’ library related with formulas (17) (19), but it might be limited because of following two reasons: Firstly, actuating system is done by applying the predefined policies for controlling. Secondly, verifying the appropriateness of applying them is also done as in-process of controlling. Moreover, as the adjustment regulation process is done, it might be possible to save efforts in the system identification process. Due to a hybrid approach among several technical elements as our unique point, it is obviously predictable that our approach overcomes the limitations caused by existing feedback controlling approach.

5.2 Granularity of Modeling

Here, the relationship between the granularity of the model and its capability for controlling will be discussed. There have appeared some other control methods including QoS-Aware reconfiguration. Then depending on the methods, there is an obvious divergence in modeling and its granularity. A. Akzhalova et al. have established the feedback control method with QoS based on queuing theory [7]. Here, a process is mapped to the server with its queue, and Auto-Regressive Moving Average techniques are applied to predict the arrival rate of the service request and others. The structural aspect of a composite web service is explicitly treated [8]; however, the main usage of it is for selecting and re-assigning suitable service instances to maintain the QoS, and there might be some conceptual differences about the controlling between [8] and our work. On the other hand, in our model, the structural
aspect of a composite web service is expressed with some formulas like (5) and (9), which become inputs for the Simulation Processor with Compensation. Hence, the dynamic and behavioral aspects caused by service executions will be imitated by the Simulation Processor.

Consider a case of executing a distributed long lived transaction under a composite web service. If the transaction failed and some relative compensation transactions are carried out, the number of related messages for service requests, including ones for the compensation transactions, might certainly increase and impact the performance of the whole of the services. However, depending on the granularity of the model, prediction of the reliability might be influenced. In the case of only capturing the number of messages without identifying types of messages, the reliability might simply get worse, whereas in the case of identifying types of messages like a retry, the reliability might be independent. The judgment of the above cases is obviously impossible without a context due to the structural aspect of a composite web service. Consequently, the degree of precision in controlling might also be affected. If compensational behaviors are explicitly specified, our model could treat the matters caused by handling the exceptions.

5.3 Potentiality to Extend the Generalized SLA Properties

As far as our consideration, the definition of the bottleneck has basically remained as the performance area. However, as mentioned before, the QoS attributes and SLA properties rely on the negotiation process. This means that it is naturally required to expand the concept of the bottleneck. If a generalized bottleneck means the lowest potential point over the whole of the targeted space, defined by the targeted attribute, our formula set could be applicable. However, some SLA properties such as the security level could have an aspect as a duality, for example ‘Supported’ or ‘Not supported’, and once they are defined with one value, they could permanently be maintained without any change. In this case, a dynamic feature such as changing the location of the bottleneck will never be required. These features should be solved by the usual approaches, including over-provisioning. Therefore, there is obviously some limitation in extending the generalized SLA properties. We still need to identify the applicability of our proposed approach one by one regarding SLA properties.

6. Related Works

Our original idea had already been proposed in [23]. In this paper, however, we elaborate on its theoretical aspects and discuss the insufficiency of combining the simple feedback controlling approach with TOC, made clear in the results of our evaluation.

The latest related works to ours are [24] and [25]. In [24], D.F. Garcia et al. mention the results of their survey with regard to QoS control mechanisms and define the set of requirements for this domain. We should also evaluate the degree of filling their requirements independently; however, that is a remaining issue. Their proposed method also has dynamic aspects and is executed under service differentiation, prioritization and calculation of the number of the assignments. However, it primarily aims for load balancing independent from the composite web services. Thus, there are obvious differences from ours. In [25], Q. Liang et al. mention the optimization of services on the application level, primarily aimed at robustness, system-oriented degree, and dynamics. Utility theory has been applied as their fundamental framework. However, we think that their approach tends to be ambiguous and to run into difficulties when applied to typical composite web services, which have complicated structural aspects.

As this covers multiple areas, so do the other related studies. As for applying control theory, models and methodologies to IT facilities, there are many existing studies, such as [7], [9]–[11] and [12]. In [9], the non-linear behavior aspect of delay control, the concept of feed forward controlling, and the topic of bottlenecks are discussed. However, the method is limited to within a web site, and control operations are treated only with resources. Thus, it is different from our research from the point of view of the architectural layers. In [10], the queue and proxy are treated; however, there is a difference in the targeted points. In [11], S. Graupner et al. treat the classification of automation controllers based on feedback control and mention the model expressed by Petri-Net. However, its configuration differs from ours. The research by D. Gmach et al. [12] partly has some similarities with our research. In this research, the controlling and optimization are classified into three levels, and our target corresponds to their ‘small level’. Their approach also uses queues and scheduling. Queue length is similarly used as an indicator for system control; however, actuations are done through resource control. Therefore, it might be effective for controlling a single elemental web service, but probably not for a composite web service.

As for the negotiation and service selection processes to make the control process function well, there are some remarkable research studies. With regard to the method assuming the use of negotiation processes for agreements, there is the research by I. Brandic et al. [13]. This is one of the generalized approaches in such cases as where a preliminary negotiation procedure is mandatory. On the other hand, with regard to selecting services in response to the QoS requirements, there are the exciting works done by L. Zeng et al. [8] and G. Canfora et al. [14]. They take one of the major approaches in this area to find the suitable combination of services under the given QoS constraints. However, this is an approach for selecting services rather than a methodology based on control theories. Thus, there are only a few indirect relationships with our proposed approach.

As for the QoS attributes of the composite web service, there are also many existing studies, including [4], [5], [15], [16] and [18]. The aforementioned [8] and [14] also address defining the attributes. In particular, in [14] one of the possible fitness functions is defined. Further-
more, there is the research with regard to the comprehensive approach by C. Wang et al. [19]. For instance, in [4] and [5], there are certain contributions of concrete definitions of QoS attributes, methods to compute them and collapsing transformation. On the other hand, S. Rosario et al. [17] treat the probabilistic QoS and software contracts for composite web services, mainly for monitoring the delays in these services. They propose this in order to solve the impractical status caused by inflexible fixed values. Currently, our assumptions rely on fixed values like the results of studies [4] and [5], and the methodologies in defining our QoS attributes and SLA properties are implicitly discussed and abstract in general. Therefore, we might have some remaining room for treating the probabilities.

As for the reference model, aforementioned studies [4], [5] and [8] may be related. Furthermore, the research by G. Wang et al. [20] is partly related. However, a reference model might be related to the mathematical model, and we need to evaluate the relationship in greater depth. With regard to prioritization related to bottlenecks, a related basic concept is mentioned by Y.S. Wang et al. [21], but these concepts are generalized in our approach.

7. Conclusion

In this paper, we presented a proposal for a new conceptual approach to realizing a control in performance for composite web services. Compared with existing methods, the uniqueness of our method is a hybrid approach consisting of several technical elements such as TOC. The prominent contributions of this work are; (i) Establishing a theoretical backbone of new controlling principles in the service computing domain. (ii) Defining a control framework with simulation functionality and applied policies under verification in multiple times. Due to the novelty of our approach, there are many unresolved issues. For instance, the effectiveness of implementing this architecture still remains to be proven. Extending the reference model with the layered structure, making the mathematical control model stricter and designing expression for policies are other issues which need to be considered. In particular, having controllable resources in a data center would require a layered hybrid method with resource control at the local sites. Thus, enhancing the model on this point also merits attention. Finally, designing a cost effective algorithm for the Simulation Processor with Compensation is also a crucial remaining issue.

Acknowledgment

Prof. Subhash Bhalla of the University of Aizu gave us many helpful comments on this research. Here we express our appreciation and gratitude.

References


Shinji Kikuchi received the M.E. from Tohoku University, Japan in 1987. After that he joined NEC Corp., where he previously belonged to the laboratory but now works in the Innovative Solution Division. He is also a doctoral candidate in database systems at the University of Aizu, Japan. He is a member of the Information Processing Society of Japan and IEEE Computer Society.

Yoshihiro Kanna received the M.E. from the University of Tokyo, Japan in 1995. Since then he has worked for NEC Corporation. He is a member of the Information Processing Society of Japan.

Yohsuke Isozaki received the M.Sc. from the University of Tokyo, Japan in 2002. After that he joined NEC Corp., where he previously belonged to the laboratory but now works in the 2nd Network Software Division.