PERFORMANCE EVALUATION OF SOA IN CLOUDS

Ashraf M. Abusharekh
Dalhousie University
Halifax, Nova Scotia
Canada
ashraf@dal.ca

Alexander H. Levis
George Mason University
Fairfax, Virginia
USA
alevis@gmu.edu

KEYWORDS
Service Oriented Architecture, Architecture Federation, Multi-formalism Modeling, Measures of Performance, Measures of Effectiveness, Quality of Service.

ABSTRACT

An approach for constructing an Enterprise Service Bus based Service Oriented Architecture for individual clouds and then considering workflows where services from multiple clouds are used is formulated and a systematic methodology for performance evaluation of the architecture is presented. The participation in this environment is achieved by allowing the SOA to dynamically federate with services through Community of Interest registries, i.e., different clouds, and by utilizing these services to share enterprise-level information. The performance prediction methodology is based on multi-formalism modeling.

INTRODUCTION

Service orientation enables enterprise interoperability and resource re-use. However, as clouds proliferate, each with its own core and application-specific services, the problem of invoking services and creating workflows that use services from two or more clouds needs to be addressed. Consequently, a methodology to evaluate and predict the logical, behavioral, and performance characteristics of such a federated Service Oriented Architecture (SOA) becomes a necessity. Such a methodology would synthesize an executable model capable of capturing the complexity of a SOA *federation* deployed on different clouds.

The system architect or designer needs to analyze the *dynamic* behavior of the proposed workflow, identify logical and behavioral errors not easily seen in the static documentation of the architecture, and demonstrate the capabilities that the architecture enables and how well they could be used. What makes the problem challenging is that the system's behavior and performance don't only depend on the system's services but also on (a) the infrastructure services that enable loose coupling, (b) services implemented by other systems, i.e., residing in other clouds, and (c) the underlying network supporting the different cloud architectures. Sustaining acceptable end-to-end performance in such a dynamic environment becomes a challenging problem.

This paper presents a methodology for performance evaluation and prediction of an Enterprise Service Bus (ESB) enabled SOA federation. The methodology involves the development and implementation of a multi-formalism based executable model that is capable of capturing and predicting the dynamic behavioral and performance aspects of a SOA federation. The executable model aids the system architect in debugging and evaluating the architecture, and helps verify that the proposed architecture will satisfy requirements and how well it will do so.

RELATED WORK

Liu et al. (2007) presented a performance modeling and benchmarking approach that facilitates estimating performance characteristics of the Enterprise Service Bus and analyzing the performance relationship between the ESB and its composite applications. To simplify the performance analysis, their work focuses only on the performance of ESB routing and transformation; this model does not incorporate the orchestration service, or modeling of the technological network.

Sloane et al. (2007) presented a hybrid approach to modeling SOA systems of systems in which two separate models are developed, a Colored Petri Net (CPN) model used to capture internal protocols, communications and resource consumption and a discrete event simulation model called MESA (Modeling Environment for SOA Analysis) to capture the interactions between nodes in a SOA environment. Although called a hybrid approach, the CPN and the MESA models are completely isolated. This approach is hard to generalize to capture different behaviors, and doesn't fully capture the effect of the network on the behavior and performance of a SOA. Finally, the approach does not capture business processes, the very driver of employing cloud-based service orientation in the first place.

Shin and Levis (2003) use CPN and network simulator models to gain insight into the behavior and performance characteristics of architectures. Their approach has two separate executable models, the functional executable model is a CPN and the physical (communication) model is a queuing net modeled using the ns-2 (2008) network simulator. The simulation models run offline, i.e., no message exchange between the two executable models during run-time.

Abusharekh, et al. (2009) presented an approach to evaluating the end-to-end response time of business processes deployed in an ESB-enabled SOA environment. The discrete event simulator OMNeT++ (2009) provided the behavior environment in which SOA-based performance was evaluated. An abstract ESB model which supports business process orchestration, routing, and reliable messaging was introduced. The model was capable of specifying the SOA supporting network to the needed level of detail. This approach assumed that

Proceedings 30th European Conference on Modelling and Simulation ©ECMS Thorsten Claus, Frank Herrmann, Michael Manitz, Oliver Rose (Editors)
ISBN: 978-0-9932440-2-5 / ISBN: 978-0-9932440-3-2 (CD)

an architecture had been designed and that its description consisted of a set of static views describing the logic and behavior of the business services and business processes and a physical view.

Ghasemi, et al. (2014) transformed UML activity diagrams to generalized stochastic Petri Nets (GSPN); although their approach is targeting SOA performance evaluation, it does not incorporate the complexities of SOA ESB, or capture the overhead associated with the cloud and network services. Duan (2015) focused on the critical role of network communications in cloud computing and its effect on end-to-end performance of cloud service provisioning. He presented an approach for characterizing the service capabilities of a composite network–computer system using network calculus. The approach targets the cloud platform and does not fully capture the architecture to be deployed to the cloud infrastructure. Bocciarelli, et al. (2015) introduced a model-driven approach to generate HLA-based simulation from SysML specifications of autonomous systems.

The present work extends the work done by Abusharekh, et al. (2009) by employing multi-formalism modeling in which CPN and network models interoperate during execution. Our approach goes further in allowing the architect to capture the specifics of the communication network at any level of detail through the network simulator. Furthermore, our approach allows the architect to capture the SOA components and the underlining business processes and their interaction with the technological network.

THE DESIGN PHASE

In a SOA federation, SOAs co-exist in different clouds. Each SOA has established producer-consumer relationships, such that the right rules and policies (trust, governance, security, etc.) apply throughout its environment. This allows for autonomy of individual SOAs but requires implementing federation-wide rules and policies to regulate and govern the federation. (Erl, 2004; Goodner and Nadalin, 2007)

The concept of Community of Interest (COI) is used to enable dynamic federation with pre-defined or un-anticipated systems. A Community of Interest is a problem-solving entity that utilizes services across different SOAs to implement the workflow that addresses its problem. In order to simplify and speed-up the discovery of services, COIs will not only define common vocabularies, taxonomies, data standards, interchange agreements, and specifications among COI members, but also will define service descriptions relevant to the communities and will host a repository of current implementations of those services. Each COI will have its federation repository.

To further clarify the SOA federation construct and how the notion of COI enables and supports such an environment, an example is depicted in Fig. 1. Several problems with this approach need to be resolved such as the location of a COI repository that should be negotiated and agreed upon among participating parties. The Cloud environment is assumed to host the COIs and the Cloud registry is the central federation repository/registry that publishes COI information. When a service failure occurs candidate services are examined at other SOAs to locate an alternative service implementation.

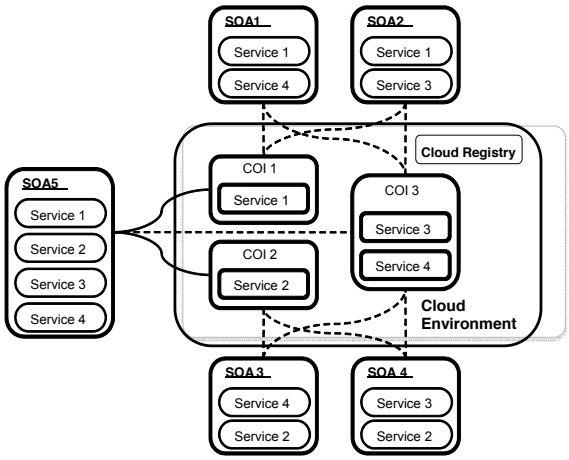

Fig. 1 Cloud Environment and SOA Federation

The objective of the approach is to construct an event-driven SOA capable of participating in the Cloud environment by consuming existing services and/or to populate it with new ones that can be consumed by anticipated and un-anticipated users. This SOA infrastructure is based on an Enterprise Service Bus (ESB). The services and processes are published through their Communities of Interest (COI) as services for other SOAs and COIs to re-use.

Two additional sources of information are needed to be able to insert a new SOA in the Cloud environment:

(1) Information about existing COIs and the services they expose in order to be able to consume services and to publish new ones. Full understanding of the other COIs policies and rules, their data formats, and services descriptions is needed to successfully federate with them; the new SOA will need to abide by them. The Cloud Registry Service will be the main source of such information. The designer needs to understand the services exposed by other SOAs in order to make a decision whether or not they will fulfill the functional and non-functional requirements of the new federate.

(2) Access to existing Enterprise Services currently available in the Cloud environment. These enterprise services will allow for trustworthy enterprise-level data, information, and services sharing.

The products of the Design phase are then used in the Analysis and Evaluation phase to construct an executable model. Structural and behavioral models of a system architecture (in this case of a workflow) are static representations of an unprecedented, complex, and dynamic system. These models are capable of describing the behavior of the architecture only in a limited way. Consequently, there is need for evaluation techniques that go beyond static diagrams. An executable model of the

workflow design enables the designer to analyze its dynamic behavior, identify logical and behavioral errors not easily seen in the static descriptions, and demonstrate to the user the capabilities that the workflow enables.

THE ANALYSIS AND EVALUATION PHASE

A key concept in the Analysis and Evaluation phase is that all elements of the executable model must be traceable to elements in the architecture description of the workflow; any corrections or changes introduced in the executable model should be reflected back in the architecture data.

In order to predict the dynamic behavior and performance of a process or application composed of multiple services, behavior and performance characteristics of participating services and the supporting network must be captured. To accomplish this, the executable model makes use of two model formalisms, a discrete event system model expressed as a Colored Petri Net and a network model expressed as queueing network.

Inputs to the Analysis and Evaluation phase are the structural and behavioral models that describe the functional, the services, and the systems views of the architecture. Outputs of this phase are either changes to the architecture, or an architecture ready for deployment along with its Measures of Performance (MOPs) and Measures of Effectiveness (MOEs).

Stage 1: Behavioral and logical evaluation. Verification of the logical and behavioral aspects of the architecture must be done before performance evaluation. An executable model of the business processes and services is built using CPNTools (2016). The inputs to this stage are the data in the functional architecture models, and the outputs are the services definitions, the services implementations as CPN Models, and the business processes as XML files, all of which are fed to Stage 2.

Step 1: Synthesizing CPN Executable Model

The structure of the CPN model includes organizational nodes, the services under their organizational boundaries, and the business processes they own. Although the CPN model is for the functional viewpoint of the workflow design, Services and Systems models are also needed to define and construct services definitions, interfaces and implementations.

Two types of services are modeled in CPN, singleton services and composite ones. A singleton service has one input place representing service requests and one output place representing service response. A composite service has two additional places, an output place representing a process request and an input place representing a process response. For simplicity, each service is assumed to have one and only one activity (function) and the current composite service model allows for one process to be requested.

Step 2: Evaluation using the CPN Executable Model

At this step, scenarios need to be defined to evaluate the logical and behavioral aspects of the design and any logical or behavioral errors will be captured and fixed and be reflected directly to the architecture models to maintain traceability. The state space analysis tool embedded in CPNTools is used to generate and examine the model's state space to detect errors or unwanted behavior. After the behavioral and logical analyses are done, processing delays of the services are added to the CPN model (Timed CPN) and the performance of the processes is analyzed. This performance reflects processing delays of services only; additional overhead due to SOA infrastructure services and the underlying network infrastructure is not captured here. If the performance of the CPN model does not meet the requirements of the architecture, the designer must make changes to the architecture to improve the performance. This CPN model performance serves as the best case (baseline) performance of the design; adding the SOA and network infrastructure will degrade performance.

Stage 2: Performance prediction and evaluation. The inputs to this stage are the outputs of Stage 1 and the Services viewpoint models; the outputs are changes to the design. The tools suite used at this phase is the C2 Wind Tunnel. (Balogh et al., 2008) This is a High Level Architecture (HLA, 22000) simulation environment that integrates various simulation platforms, including CPNTools and OMNeT++. The C2 Wind Tunnel integration is done on three levels: the API level, the interactions level, and the model semantics level. API level integration provides basic services such as message passing and shared object management, while the interaction level integration addresses the issues of time synchronization and coordination. Semantic integration is more subtle and depends upon the goals and the context of the overall simulation environment. At the API and interactions level the C2 Wind Tunnel uses Portico (2009) an open-source, cross-platform HLA RTI implementation. At the level of model semantics it uses the meta-programmable Generic Modeling Environment (GME, 2009) engine to integrate the operational semantics of multiple simulation platforms, to manage the configuration and deployment of scenarios in the simulation environment, and to generate the necessary interface code for each integrated simulation platform. GME is used to generate configuration files and HLA interactions and glue code for the C2 Wind Tunnel to host the two interoperating models that make up the multi-formalism based executable model. Configuration is accomplished through meta-models that formally specify the modeling paradigm of the application domain. A meta-model is used to define all the syntactic, semantic, and presentation information regarding the domain.

The goal is to compute the creation time (T_C) and end-to-end response time (T_R) of processes deployed on such an environment as defined in Abusharekh, et al. (2009). T_C and T_R depend on the behavior and performance of the ESB services and the business services contributing to the business process, the underlying network supporting the SOA and the cloud environment, and the request load of business processes deployed on the SOA at a given time. In order to capture the above factors and the related characteristics of the environment, five profiles

need to be created: (1) Network profile, (2) ESB profile, (3) Services Profile, (4) Processes Profile and (5) Scenario profile. Abusharekh et al. (2009) provide a full description of the profiles and their structure.

Step 3: Building the Multi-formalism based Executable Model

In the executable model services are modeled in CPN-Tools while the network infrastructure is modeled in OMNeT++. The C2 Wind Tunnel instance that was used hosts two types of federates, CPN federates and Network federates. The structure of the network model is shown in Fig. 2. MOM is the Message oriented Middleware module.

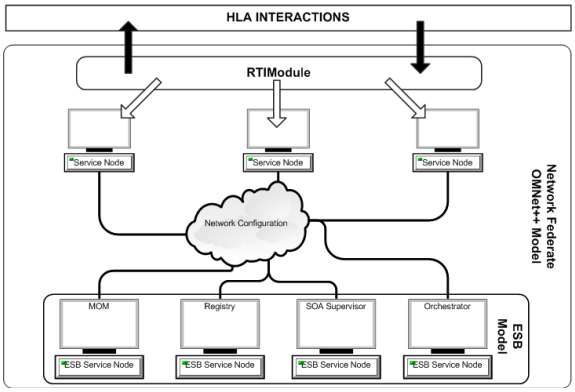

Fig. 2 OMNeT++ Model Structure

The OMNeT++ model implements the RTI module, a generic SOA ESB model, service nodes, and the network topology. The RTI module supports communication with the C2 Wind Tunnel through HLA interactions. The ESB model is the same as that introduced in Abusharekh et al. (2009) but enhanced by including a Service Application module managing interactions from/to the CPN model through the RTI Module.

Step 4: Evaluation Using the Multi-formalism based Executable Model

This step needs a scenario profile to define the request loads on different business processes and within different classes of a business process. The results of the execution are analyzed, after which the designer either is satisfied with the results or decides to make changes to one or more of the profiles in order to improve the overall performance.

The performance evaluation of the architecture is done using the System Effectiveness Analysis methodology (Levis, 1997). The MOP of interest are *Timeliness* and *Accuracy* of a federated SOA architecture. Accuracy is defined as the expected cost of a business process producing an outcome different from the desired one. The variables quantifying timeliness are business process response time, creation time, and throughput rate. Identifying variables quantifying accuracy depends on the specific mission and objectives of the workflow.

After the executable model has been configured, the Simulation panel of the C2 Wind Tunnel is used to generate all necessary files for the executable model to run.

A CASE STUDY

The case study results presented here are based on a hypothetical operational concept for a system called Airborne Theater Ballistic Missile (TBM) Interceptor System (ATIS). This case study is used here because a documented complete architecture exists. (Abusharekh et al. 2007)

In order to create a cloud based architecture capable of intercepting and destroying TBMs and capable of providing and consuming information and services to and from this particular cloud environment we need to: (a) define services and processes to be hosted by the ATIS (internal services); (b) re-use business services and/or processes implemented by other and residing in different clouds systems (external capabilities); and (c) publish relevant ATIS services to be used by other systems in this cloud environment. To re-use existing services and populate the cloud environment with new ones, ATIS must join relevant COIs that have their own SOA in different clouds. Two COIs are modeled: (a) the Ballistic Missile Response (BMR) COI: a collaborative group of cloud users who exchange information regarding ballistic missile response; (b) the Intelligence, Surveillance and Reconnaissance (ISR) COI: a collaborative group of cloud users who exchange information related to ISR.

The operational concept graphic of the architecture is shown in Fig. 3. It is assumed that Core Enterprise Services (CES) are available and accessible. The Systems and Services Interface Description is shown in Fig. 4 as a Unified Modeling Language Deployment diagram.

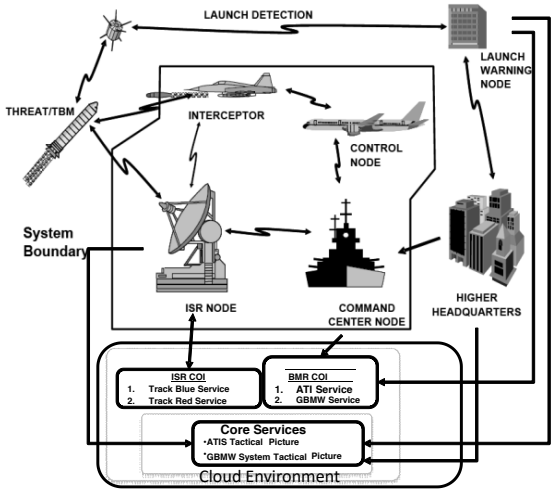

Fig. 3 Operational Concept Graphic for ATIS

A CPN executable model using CPNTools was created to evaluate the operational aspects of the design. This first model included the processes and services of the ATIS, but no ESB interaction was included. Once created, the executable model was used to check the logic and behavior of the services and their composition into processes. As errors were detected, fixes were made to the CPN model and reflected back to the architecture models. The performance of the ATIS capabilities was tested by converting the CPN model into a Timed CPN.

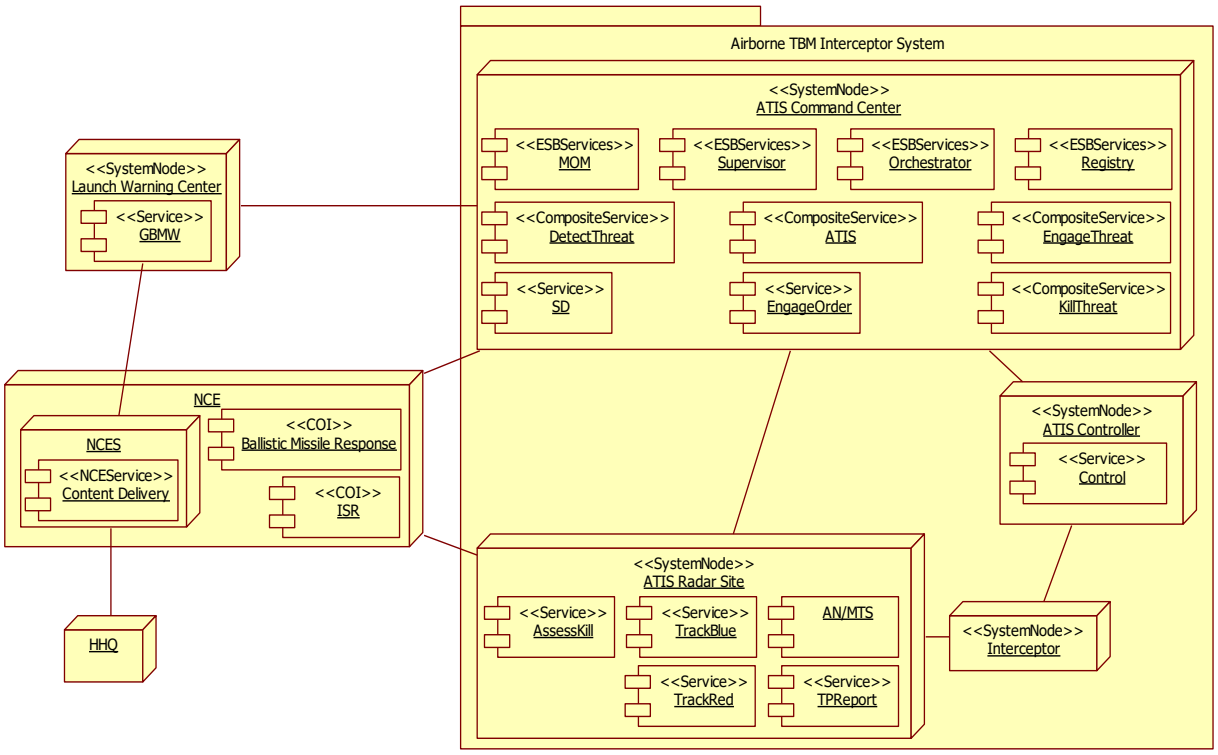

Fig. 4 Services & Systems Interface Description

The main questions to be addressed are: (1) Is the ATIS capable of intercepting in a timely manner incoming TBMs? (2) How many interceptors are required to handle various adversary capabilities while keeping the average response time and number of leakers within the requirements? Therefore, the parameters of interest are:

1. TBM inter-arrival time: a continuous parameter with values used in the experiments of 0, 25, 50, 75 and 100 seconds.
2. The number of ATIS interceptors: a discrete parameter with values 3, 4 and 5 interceptors.

The resulting parameter locus is shown in Fig. 5. It consists of the three vertical lines.

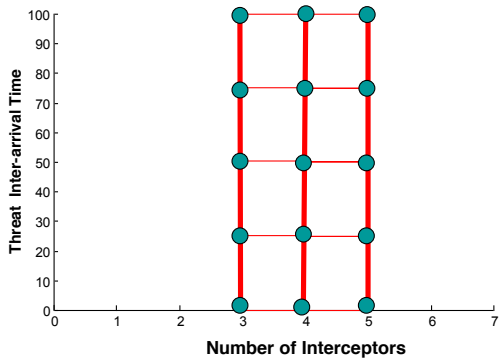

Fig. 5 The Parameter Locus

Three Measures of Performance were used to assess ATIS: (a) *Average response time:* the average time between ATIS detecting the threat and the threat being engaged by ATIS interceptors. The requirement is for ATIS to be able to destroy a TBM within 400 seconds of detecting it. (b) *Accuracy*: the number of leakers (TBMs not destroyed after 400 seconds of being detected by ATIS). This must be no more than 2. (c) *Throughput rate*: the number of TBMs processed per unit time. There is no constraint so it is parameterized.

The following scenario was used for performance analysis. An adversary capable of launching multiple missiles exists and the ATIS and the Global Ballistic Missile Warning service provided by the BMR COI have been deployed. The summary of the results of the 15 simulation runs for the baseline case in which the SOA and network infrastructures are not present are in Table 1. They show that 3 interceptors can handle the 10 threats with a maximum of two leakers if the threats arrive at a rate slower than 1 in 25 seconds. These results represent the best performance the architecture could accomplish.

Table 1. Number of Leakers vs. Number of Interceptors

# of Interceptors	TBM Inter-arrival	Average Response Time	# of Leakers
3	0	347.1	4
	25	270.1	1
	50	180.6	0
	75	159.0	0
	100	159.0	0
4	0	283.9	2
	25	212.9	0
	50	159.0	0
	75	159.0	0
	100	159.0	0
5	0	245.5	0
	25	180.0	0
	50	159.0	0
	75	159.0	0
	100	159.0	0

The multi-formalism based executable model captures the communications systems and the SOA infrastructure and shows how they will interact to enable the composition of services in processes for successfully executing the mission.

To predict the performance of federated SOAs, the scenario was modified so that the first ATIS unit, ATIS_A, federates with another unit of ATIS, ATIS_B in order to overcome failures in services during an attack. The two ATIS units cover adjacent geographical regions and are both members of the ISR COI. The region under ATIS_A is under attack, and after 350 seconds, ATIS_A's ISR node fails. ATIS_A then federates with ATIS_B (not under attack) and re-uses its *TrackRed*, *TrackBlue*, *TPReport* and *AssessKill* Services published through the ISR COI. The goal is to show the effect of SOA federation on the performance of the architecture.

Figure 5 shows the average response time of the ATIS for three different message sizes (1KB, 500KB, and 1000KB with the latter containing images) and how it is affected by increasing the number of interceptors under the federated network infrastructure. With long TBM inter-arrival times, increasing the number of interceptors produces no significant gain in response time for large message sizes. The best achievable performance for different messages sizes is:

1. 1KB message sizes, average response time of 184.5 sec with no leakers.
2. 500KB message sizes, average response time of 238 sec with 4 leakers.
3. 1000KB message sizes, average response time of 290 sec with 10 leakers.

Sensitivity analysis of the number of leakers to the number of ATIS interceptors and message size, respectively, with different adversary launch capabilities, showed that, with large message sizes, increasing the number of interceptors will not decrease the number of leakers.

For each operating point in the parameter locus, a value for each Measure of Performance is computed using the executable model. The set of all such values is the Performance Locus. The Requirements locus is the set of admissible values of the two measures of performance: Average Response Time $\leq$ 400; Leakers $\leq$ 2.

The ATIS Measures of Effectiveness are calculated by comparing the measures of performance against the requirements. (Levis, 1997) This is computed as the fraction of the Performance locus that intersects the Requirements locus.

ATIS Requirements and Performance loci for the single SOA without communication delays as a function of message size are shown in Fig. 7. Considering the performance requirements for the average response time and the number of leakers, the Measure of Effectiveness of the single SOA model can be calculated by considering the projection of the performance locus on the average response time against the number of leakers. The resulting value is 93%, i.e., the single SOA architecture without communication delays is 93% effective.

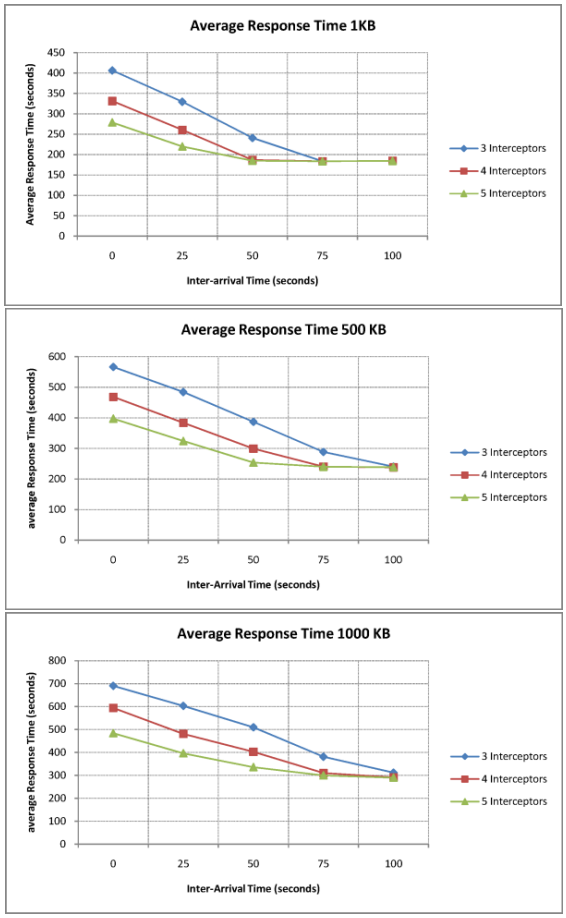

Fig. 6 Average Response Time vs. Inter-arrival time

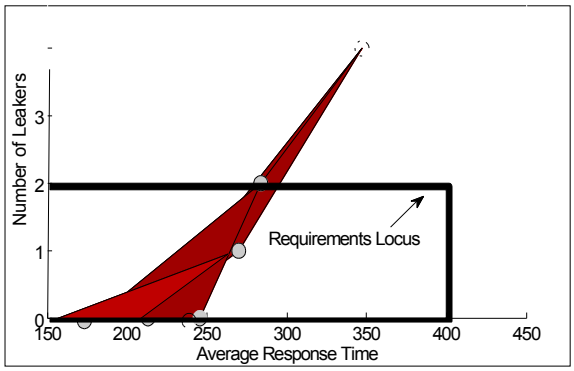

Fig. 7 Evaluation of Measure of Effectiveness

When the same analysis is done for the federated SOA architecture, the Measure of Effectiveness drops to 73% for 1KB message size and to 0% for the larger message sizes. This is a significant result because it shows that the infrastructure (computers and communications) needed to implement a federated SOA architecture exact a very substantial Quality of Service cost. For time sensitive operations, this cost may be unacceptable. For other types of operations, this type of cost should be considered when determining whether to use federated

SOAs, i.e., creating workflows that use services residing in different clouds, or not.

CONCLUSIONS

This paper introduced a *SOA federation framework* that allows dynamic federation between SOA instances that are implemented in different clouds. This design enables the provision of new services and the consumption of existing ones through dynamic federation.

The evaluation approach is based on synthesizing an executable model that is capable of capturing and predicting the dynamic behavioral and performance aspects of the Service Oriented Architecture. The executable model employs multi-formalism that makes use of two models, CPN models for the SOAs and the workflows and an OMNeT++ network model that captures the underlying technological infrastructure in some detail.

One of the main conclusions of this analysis is that using the multi-formalism based executable model gave more insight and understanding of the dynamics of a federated SOA design. While SOA offers many advantages such as information sharing, reuse, and interoperability it also has its drawbacks. More information sharing across the federated SOAs introduces more latency and can cause substantial degradation in the quality of service (QoS). For time-critical system such as ATIS, more information sharing is not sufficient to produce better actions due to time constraints. Services receive better information but too late to act on it.

The critical conclusion is that while a single SOA offers many performance and agility advantages, a federation of SOAs introduces Quality of Service issues that may indicate that this is not always the right solution. Thus it is necessary to analyze alternative architectures to determine the appropriate solution for a particular problem. The executable model presented is capable of verifying, evaluating, and demonstrating the capabilities of a SOA and how well it performs as a single instance and also as part of a federation. In addition, the executable model can be used as a test-bed to evaluate new algorithms and protocols to enhance the design.

With the emergence of cloud and containerization (PaaS/IaaS) technologies, deploying a federated SOA that satisfies the functional and non-functional requirements is becoming a challenge for the architect/designer. Future work includes exploring and evaluating the use of these technologies to achieve the targeted non-functional requirements of the architecture, such as timeliness and availability. COI concepts can be leveraged to allow dynamic evolution of the federated SOA at runtime to fulfill its mission or the COI mission, e.g. using service mirroring to improve performance and/or availability. These issues need to be explored and evaluated in a quantitative manner, before investing in implementing and deploying the architecture.

REFERENCES

Abusharekh, A., S. Kansal, A.K. Zaidi, and A.H. Levis, 2007. "Modeling Time in DODAF Compliant Executable Architectures," in *Conf. on Systems Engineering Research*, Hoboken, NJ.

Abusharekh, A.; L. E. Gloss; and A.H. Levis, 2009. "Evaluation of SOA-based Federated Architectures," *Systems Engineering*.

Balogh, G. et al., 2008. "Rapid Synthesis of HLA-Based Heterogeneous Simulation: A Model-Based Integration Approach.," Unpublished manuscript.

Bocciarelli, P., A. D'Ambrogio, A. Giglio, and E. Paglia. 2015. “A model-driven framework for distributed simulation of autonomous systems.” In Proc. Symp. on Theory of Modeling & Simulation: DEVS Integrative M&S Symposium (DEVS '15), F. Barros, M. H. Wang, H. Prähofer, and X. Hu (Eds.). Soc. for Computer Simulation International, San Diego, CA.

CPN Tools, 2016 [Online]. http://cpntools.org/

Duan, Q. 2015. “Modeling and performance analysis for composite network–computer service provisioning in software-defined cloud environments.” Digital Communications and Networks, (1)3.

Erl, T., 2004. *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*, Prentice Hall.

Ghasemi, A., A. Harounabadi and S. J. Mirabedini. 2014. “Performance Evaluation of Service-Oriented Architecture using Generalized Stochastic Petri Net.” Int. J. of Computer Applications 89(6).

GME, 2009. Generic Modeling Environment. [Online]. http://www.isis.vanderbilt.edu/projects/gme/

Goodner, M. and A. Nadalin, 2007."Web Services Federation Language (WS-Federation) Version 1.2," September 26.

HLA, 2000 IEEE Standard for Modeling and Simulation (M&S) High Level Architecture.

Levis, A. H. 1997. "Measuring The Effectiveness of C4I Architectures," in 1997 *Int. Symp. on Defense Information*, Seoul, Republic of Korea.

Liu, Y.; I. Gorton, and L. Zhu, 2007. "Performance Prediction of Service-Oriented Applications based on an Enterprise Service Bus," in *Computer Software and Applications Conference*, vol. 1, Beijing, pp. 327-334.

NS-2, 2016. The Network Simulator - ns-2. [Online]. http://www.isi.edu/nsnam/ns/

OMNet++ 2016. [Online]. http://www.omnetpp.org/

Portico, 2009. The Portico Project. [Online]. www.porticoproject.org/

Shin, I. and A.H. Levis, 2003."Performance prediction of networked information systems via Petri nets and queuing nets," *Systems Engineering*, vol. 6, no. 1, pp. 1 - 18.

Sloane, E.; T. Way; V. Gehlot; R. Beck; J. Solderitch; and E. Dziembowski, 2007. "A Hybrid Approach to Modeling SOA Systems of Systems Using CPN and MESA/Extend," *Systems Conference*.

Dr. Ashraf M. Abusharekh is a senior research scientist at the Faculty of Computer Science at Dalhousie University, NS, Canada. **Dr. Alexander H. Levis** is University Professor of Electrical, Computer, and Systems Engineering at George Mason University, Fairfax, VA., USA.