Using hybrid SoSE approaches for modeling and validating large scale Service Oriented Architecture (SOA) System of Systems as next-generation global military informatics platforms with Colored Petri Nets (CPN) and Extend/MESA

Elliot Sloane, Thomas Way, Vijay Gehlot, and Robert Beck
Villanova University
800 Lancaster Avenue
Villanova, PA 19085
610-519-6432
{elliot.sloane, thomas.way,
vijay.gehlot, robert.beck}@villanova.edu

Abstract - *Service Oriented Architectures (SOAs) are rapidly becoming an accepted means for network information exchange across heterogeneous fabrics of nodes. In architecture, an SOA is a system of systems (SoS) with complex interactions. Owing to SOA benefits of flexibility, interoperability, loose coupling, and reusability, the Department of Defense (DoD) has a future SOA framework for defense applications known as Net-Centric Enterprise Solutions for Interoperability (NESI). This paper details the modeling effort of a Multi-Channel Service Oriented Architecture (MCSOA), an enhanced SOA for DoD's requirements. MCSOA's Discovery protocol enables location transparency, availability awareness, service guarantees, context-based routing, and fault-tolerance. To handle MCSOA's complexities we adopted hybrid modeling, using Colored Petri Nets for the internal structure and interactions, and Extend/MESA for data exchanges at the network level. Reliable Extend/MESA models need error-free internal interactions, which we validate in the underlying CPN model, The paper contains a brief description of our models and validation results.*

1 Introduction

The flexibility and interoperability of SOA makes it an attractive solution for enterprise services within DoD deployments. The DoD and Defense Information Systems Agency (DISA) have defined a core set of enterprise services for use in defense-related SOA systems, as part of an initiative called Net-Centric Enterprise Solutions for Interoperability (NESI). These Net-Centric Enterprise Services (NCES) define by NESI are a set of services, nodes and utilities for use in DoD domain- and mission-related enterprise information systems (Figure 1, [1]). Development of NCES is ongoing with significant efforts currently underway in systems for various defense applications.

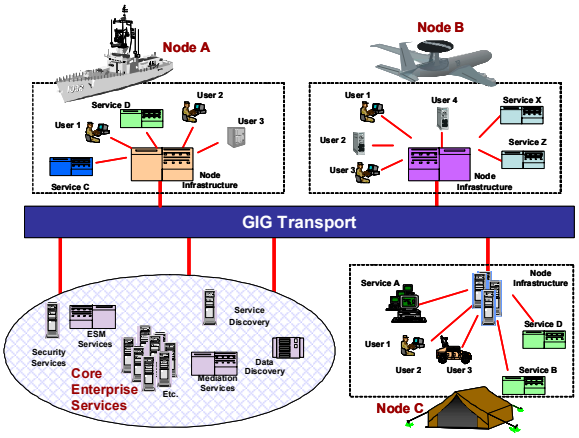

Figure 1. DoD Net-Centric View [1]

Under a contract with the USAF, Gestalt, LLC. defined a set of software components forming the architecture of their Multi-Channel Service Oriented

Architecture (MCSOA) [2]. In addition to defining a general SOA framework, the MCSOA architecture was developed to fulfill the unique requirements of the DoD. In particular, MCSOA addresses the dynamic nature of the war-fighting environment and helps extend the reach of net-centric services out to the tactical edge. One of the key components of the MCSOA architecture, therefore, is its *dynamic discovery* mechanism. Several MCSOA nodes can be connected to each other forming a MCSOA *fabric*. The fabric is a dynamic concept in that new nodes may join the fabric through the MCSOA discovery process. Furthermore, the services available at a given node may also change dynamically. In order to fully understand and explain MCSOA capabilities, it was decided to resort to a model-based approach since there was a general limitation to subjecting the actual software to potential stringent environment. Given the complex nature of MCSOA internal as well as the interactions at the fabric level, it was decided to follow a hybrid modeling approach. For the MCSOA internal, we chose Colored Petri Nets (CPNs) as the modeling language [3]. To model the fabric level interactions, we decided to use a discrete even simulation tool called Extend and its extension called MESA [4]. The focus of the CPN model was to ascertain correct message exchange in terms of underlying protocol whereas the focus of fabric level modeling effort was to examine the effect of discovery overhead assuming the discovery mechanism works correctly.

The rest of this paper is organized as follows. In the next section we give brief details of the CPN model and following this we present the Extend/MESA fabric model. We then present validation results of these two models. Finally we state our conclusions.

2 CPN MCSOA Discovery Model

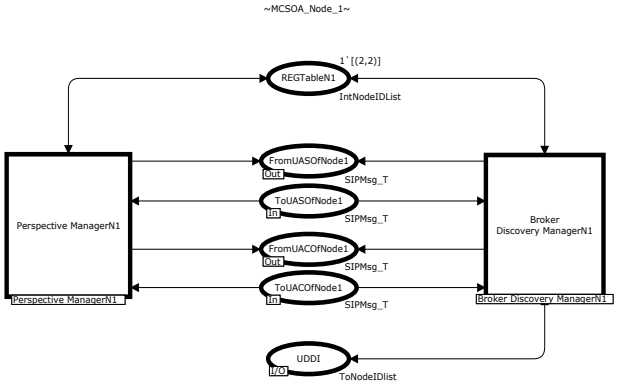

Figure 2. MCSOA Node Discovery Component CPN Model Top Level

Figure 2 depicts the top level view MCSOANode1 of the CPN model comprising the MCSOA discovery component. As depicted, a MCSOA node discovery component consists of two main parts - Perspective Manager (PM) and Broker Discovery Manager (DM). The Perspective Manager is responsible for handling registration and subscription requests. The Broker Discovery Manager handles service requests. In case of MCSOA, a service request may specify a specific endpoint (eg, service1@node1) or it may omit it (eg, service1). The latter situation is responsible for triggering the discovery process whose job is to locate a sutable endpoint for such a service and then forward the request to that service endpoint. The MCSOA requirement is that a service must be registered at some node. Thus, each MCSOA node keeps track of services known to it in a registration table. Notice that the place REGTableN1, the local registration table of MCSOANode1, is connected to both the PM and the DM, whereas the UDDI is connected only to the DM, since only the latter handles the discovery triggered by a service request. The place REGTableN1 is connected to the PM, since the PM needs to update it when a new registration or the expiration of a registration occurs. Also, the PM needs to consult REGTableN1 when a SUB message is received in order to find whether the provider mentioned in the SUB message is in REGTableN1 or not. On the other hand, the DM needs to consult REGTableN1 during local routing (i.e. routing where destination MCSOA node is the same as the source MCSOA node). The other four places insure the communication between MCSOA Node 1 and other components of Discovery CPN model. Using the module support available in CPN for hierarchical modeling, each of these two subcomponents is further refined to capture its behavior.

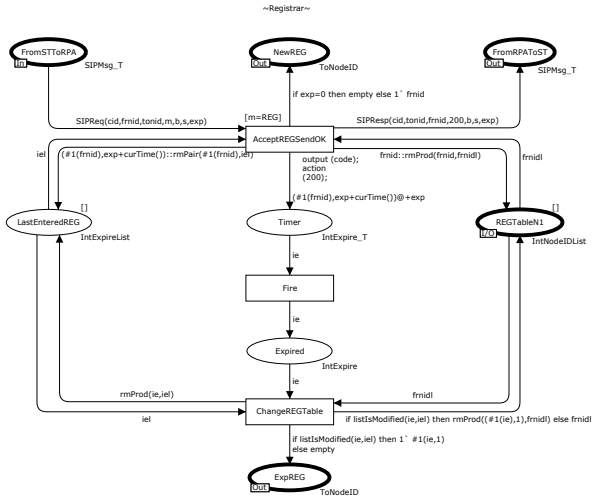

Figure 3. The Registrar Subcomponent of Perspective Manager Component

For example, Figure 3 above gives the detailed model of the registration process which is a subcomponent of the Perspective Manager shown in Figure 2. This component describes the actions taken when a new registration for either a consumer or a provider arrives. When a registration message is received, the AcceptREGSendOK transition becomes enabled. After firing the AcceptREGSendOK transition, a response is generated with the call ID equal to that of the original registration message. This response is then sent through the Server Transaction of the Registrar to User Agent 1 with the success code 200. Meanwhile, a timer is set for the entity's registration, whose value is taken from the exp parameter (expiration time) of the registration message. This is done by putting a timed token in the place Timer with a time stamp equal to the registration expiration time. In addition, firing the AcceptREGSendOK transition puts a token in the place NewREG whenever the entity that just registered is a provider. As was mentioned in the previous section, that token will be passed to the Presence Server in order to update the contents of the SubList and PendingSUB tables. Notice, that according to the SIP requirement document, there is no need to send consumers' registration messages to the Presence Server, because the subscriptions are moved between the PendingSUB and SubList tables only when providers register or unregister. Finally, firing the AcceptREGSendOK transition updates the content of the REGTableN1 by deleting any previous registration of the newly received entity and adding the ID of that entity in REGTableN1. The expression assigned to the incoming arc to the REGTableN1 ensures that the same entity will not appear in the REGTableN1 more than once, even when more than one registration or unregistration (ie, registration with expiration time equal to 0) requests are received for that entity. The LastEnteredREG table is a list which keeps the information about consumers, providers and their registration expiration times. The incoming arc expression puts the latest information about entities and their expiration times in the LastEnteredREG table. When there is a token in the place Timer with the time stamp less than or equal to the current time, the transition Fire becomes enabled, which, when fired, puts all the expired tokens in the place Expired. Firing the transition ChangeREGTable checks whether there is an entity in the LastEnteredREG table with the same ID number and expiration time as the one in the place Expired. If such an entity is found, then it is deleted from the LastEnteredREG table. The same entity is also deleted from the REGTableN1 and a token with the expired entity's ID number is put in the place ExpREG and passed to another component called Presence Server for further handling. Complete details of this model are beyond the scope of this paper and are contained in [5]. However, it should be clear that the internal details of MCSOA models are quite involved and then trying to impose the fabric level behavior on top of it would have been a very daunting task. For this reason and also for the reason of model maintainability we decided to use a complementary approach for modeling fabric level interactions which we describe next.

3 Extend/MESA MCSOA Model

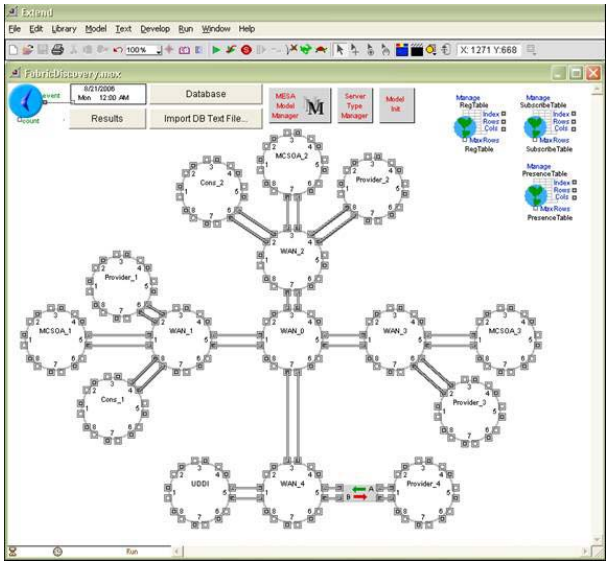

Figure 4. MCSOA Fabric Extend/MESA Model

Figure 4 shows the snapshot of the fabric model as represented using the Extend/MESA tool. Each round node shown in this figure represents a computational resource (most often a computer) on which programs can execute. Web services in a Service Oriented Architecture (SOA) are instances of such programs. With MESA, it is possible to define rather complex logical behaviors that model the effect of web services including the ability to have one service invoke another service as part of its behavior. These services can be mapped to individual nodes in the model that will execute the services. Each node is equipped with computational characteristics including CPU horsepower, number of resident processors, etc. so that execution statistics can be gathered and measured. Connections between nodes can also be modeled as shown by the rectangular box between the WAN_4 and Provider_4 nodes in the MCSOA fabric modeled with Extend/MESA in Figure 4. These connections or links can have characteristics of their own that define their behavior including properties such as link speed (in

bits per second: BPS), link latency and link background utilization. Each named service is fully defined in its own MESA database table. A service is composed of a number of *action steps*. Each action step may involve reference to a number of values obtained from the other columns in the service table. As an example, modeling of a remote request with discovery, shown in Figure 5 below, is achieved through 7 different MESA service definitions.

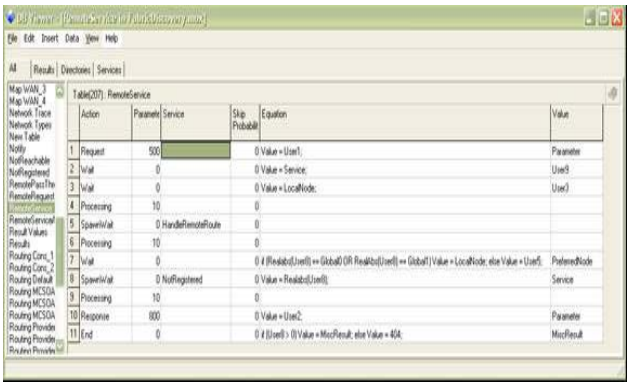

Figure 5. Remote Discovery Service

The main logic of RemoteService, shown above involves a sub-service called HandleRemoteRoute to determine the node and service that this remote service should actually invoke. User attributes (User9 and User3) are set is steps 2 and 3 and these are used within the Discovery service that is spawned by HandleRemoteRoute. The main results of the logic encapsulated in these sub-services are available as the set values of the user attributes User5 and User8. Steps 7 and 8 define the node to execute the service and the service to execute. If the HandleRemoteRoute service does not succeed in finding a provider, the service NotRegistered will be processed. Request and response sizes along with processing requirements are set with constants (steps 1, 4, 6, 9 and 10). The service result code is set as the value of the attribute MiscResult in step 11. Complete details of this model are contained in [6].

4 Model Validation

Our adopted approach to model verification and validation is based on [7] and is further described in [8]. We give a summary of the validation results of the two models next.

4.1 CPN Discovery Model Validation

The validation of this model included checking the behavior of MCSOA vs. The CPN model based on 10 use cases. These use cases were created based on discussions held with MCSOA/SIP experts at Gestalt. The Ethereal network protocol analyzer was used in the runtime lab to capture the flow of SIP packets. On the CPN Model, the built-in monitoring facilities were used to log model activities during simulation.

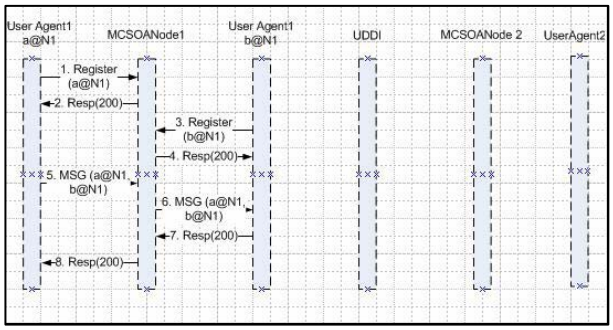

Figure 6. An Example CPN Model Validation Use Case

One of the use cases is shown in Figure 6. This use case presents the Local Routing Provider Available scenario. The sender, User Agent1 (a@N1) registers with MCSOANode1 and MCSOANode1 accepts the registration. Now, the Service Provider, User Agent1 (b@N1) registers with MCSOANode1 and MCSOANode1 accepts the registration. The User Agent1 (a@N1) sends a request for service b@N2 to MCSOANode1. MCSOANode1 has the service b@N2 registered in its local registration table so the request is forwarded to User Agent1 (b@N1). The User Agent1 (b@N1) sends the ok response to MCSOANode1, which then forwards the response to User Agent1 (a@N1). The outputs from the runtime lab and the model for this Use Case are presented in Figure 7 and Figure 8, respectively.

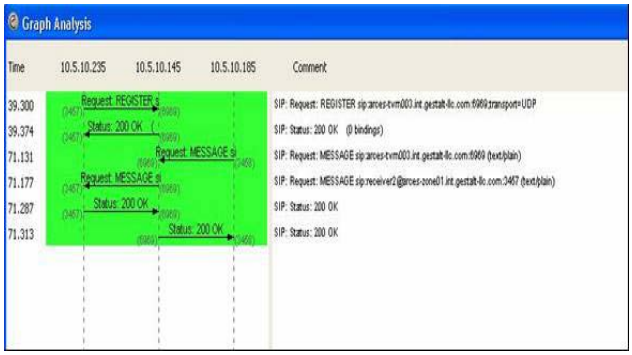

Figure 7. Runtime Lab Output

```
 2  Registration for Consumer from (3,1) to
    MCSOA_1
    Exp. Time 3000
 7  Response for Registration from MCSOA_1 to
    (3,1)
     code = 200
     REG Table          1`[(3,3010)]
15  Registration for Provider from (17,1)to
     MCSOA_1
    Exp. Time 3000
19  Response for Registration from  MCSOA_1 to
    (17,1)
    code = 200
    REG Table  1`[(17,3020),(3,3010)]
27  (MSG, (3,1), (17,1)) from User_Agent_1
31  Forward (MSG, (3,1), (17,1)) to User_Agent_1
35  Response for (MSG, (17,1), (3,1)) from
    User_Agent_1
    to MCSOA_1 code = 200
40  Response for (MSG, (17,1), (3,1)) from
    MCSOA_1
    to User_Agent_1 code = 200
```

Figure 8. CPN Model Output

The validity of the model for this use case behavior can be checked by relating the time stamps in the runtime lab output, with the step number in model output. Time 39.300 corresponds to step number 15. Here, the Provider asks for the registration of the service, with MCSOA_1. Time 39.374 corresponds to step number 19. MCSOA_1 sends the Ok response to the Provider. The local registration table for MCSOA_1 is updated with the registration of the service from the Provider with the MCSOA_1. Time 71.131 corresponds to step number 27. User_Agent_1 (the consumer) sends the request for the service (which is registered with MCSOA_1) to MCSOA_1 Time 71.177 corresponds to step number 31. MCSOA_1 finds the registration for the service by the Provider in its registration table and hence forwards the request to the Provider. Time 71.287 corresponds to step number 35. The Provider sends the Ok response to MCSOA_1. Time 71.313 corresponds to step number 40. MCSOA_1 forwards the Ok response to the consumer (User_Agent_1). Other use cases were executed and validated in a similar manner.

4.2 Extend/MESA Fabric Model Validation

There are 3 kinds of nodes supported in the current fabric model:

- Consumers, which are the sources of requests
- Providers, which are the sources of responses to requests
- Brokers (MCSOA nodes), which are enablers for consumers to locate providers and route requests from consumers to providers and responses from providers back to consumers.

Fabric model concentrated on creating a usable model featuring multiple consumer, provider and broker nodes which forms the basis for an increasingly complex set of behavioral use cases showing consumer requests and responses being routed through the fabric. In addition, benchmarking use cases were developed to validate the model's performance. The fabric model's handling of these use cases was designed based on interviews with MCSOA experts. In all cases, average round-trip time, the standard deviation of round-trip time, and error rates were recorded for 1-100 simultaneous users. All requests and responses were made via HTTP transport.

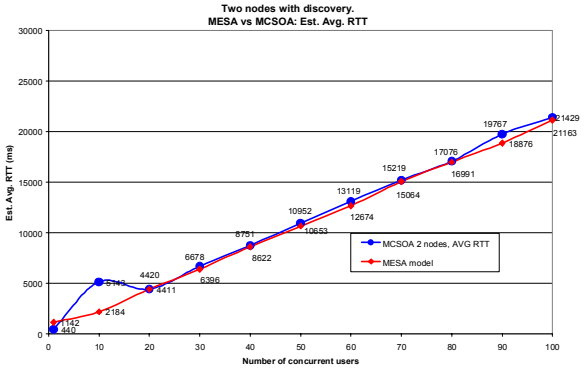

Figure 9. MCSOA vs. MESA Model Average RTT Comparison

As shown in the graph of Figure 9, the MESA model matches the runtime lab performance very closely beyond the 10-simultaneous users case. The model does not simulate initialization, and conducts discovery faster than MCSOA does. These factors become much less significant at numbers of simultaneous users larger than ten. Complete details of model validation and results are contained in [9].

5 Conclusions

In this paper we gave details of a hybrid approach to SoSE modeling, simulation, and validation of a Service Oriented Architecture called MCSOA that addresses unique defense needs. MCSOA possesses dynamic discovery capabilities. A model driven approach was undertaken to validate MCSOA

behavior. Because of complexities of the underlying architecture and interactions at the network level, we adopted a hybrid approach to modeling such systems. In this approach, we used the formal modeling tool called Colored Petri Nets (CPNs) to model internal components of MCSOA. This allowed us to validate the components of the system and identify weaknesses at the component level. To model the interactions in the fabric of interconnected MCSOA nodes we used a discrete-event simulation tools known as MESA/Extend. We presented details of validation of the two models and showed how behavioral validity ascertained by the component level model can be used to model and validate network level interactions and performance measures.

Acknowledgements

This Advanced Research for Computing Enterprise Systems (ARCES) project was supported in part by the Air Force Materiel Command (AFMC), Electronic Systems Group (ESG) under contract number FA8726-05-C-0008. The views and conclusions contained here are those of the authors and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of USAF, AFMC, ESC, or the U.S. Government.

References

[1] NESI Project. Net-Centric Implementation, part 1: Overview (version 1.3.0). Available online at http://nesipublic.spawar.navy.mil/docs/part1, last accessed 2006/08/08, June 2006.

[2] Multi-Channel Service Oriented Architecture, Internal Report, Gestalt LLC, April 2005.

[3] K. Jensen, *Coloured Petri Nets*, Volume 1 of Monographs in Theoretical Computer Science, Springer-Verlag, 2nd edition, 1996.

[4] D. Krahl, "The Extend Simulation Environment," *Proc. 2002 Winter Simulation Conference*, pp. 205-213.

[5] V. Gehlot and A. Hayrapetyan, "A CPN Model of a SIP-Based Dynamic Discovery Protocol for Webservices in a Mobile Environment". *Proc. 7th Workshop and Tutorial on Practical Use of Coloured Petri Nets and the CPN Tools*, Aarhus, Denmark, 2006

[6] ARCES Project, Model Software Design Report, Release 4, Villanova University and Gestalt LLC, November 2006.

[7] R. D. Sargent, Verification and validation of simulation models, *Proceedings of the 2003 Winter Simulation Conference*, S. Chick, P. J. Sánchez, D. Ferrin, and D. J. Morrice, eds., 2003.

[8] V. Gehlot, T. Way, R. Beck and P. DePasquale, "Model Driven Development of a Service Oriented Architecture (SOA) Using Colored Petri Nets." *Proc. First Workshop on Quality in Modeling, ACM/IEEE 9th International Conference on Model Driven Engineering Languages and Systems*, October, 2006.

[9] ARCES Project, Software Test Report, Release 4, Villanova University and Gestalt LLC, November 2006