

# A HYBRID APPROACH TO MODELING SOA SYSTEMS OF SYSTEMS USING CPN AND MESA/EXTEND

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

James Solderitch and Elzbieta  
Dziembowski  
Gestalt, LLC  
680 American Avenue, Suite 302  
King of Prussia, PA 19406  
610-994-2860  
{jsolderitsch,  
edziembowski}@gestalt-llc.com

**Abstract** - Service Oriented Architectures (SOAs) are rapidly becoming an accepted means of providing network information exchange across a heterogeneous fabric of nodes. Given the complexity of such architectures and multiple levels of system interactions, creating a valid and usable SoS model for SOA application using a single technique that captures the desired level of details can be a daunting task. In this paper we give details of a hybrid approach to modeling and simulation of a specific SOA that is named MC SOA and has been configured for potential defense applications. In this approach, at the lowest system level (call it white-box level) we are using Colored Petri Nets (CPNs) to model internal protocols, communications, and resource consumption. This allows us to validate the components of the system and identify weaknesses at the component level. For the interaction at the highest system level (call it black-box level), we are using a set of discrete-event simulation tools known as MESA/Extend. The major advantage of this combination is that the assumptions at the black-box level are fully validated at the white-box level, which greatly simplifies and accelerates the design, development, and testing of the simulations of very large scale SOA-based Systems of Systems.

## INTRODUCTION

Service Oriented Architectures (SOAs) are rapidly becoming an accepted means of providing

network information exchange across a heterogeneous fabric of nodes. In terms of its architecture, a SOA is a system of systems (SoS) with complex interactions. On the other hand, SOA benefits include flexibility, interoperability, loose coupling and reusability. Owing to these advantages, the US Department of Defense (DoD) has defined a SOA framework for defense applications known as Net-Centric Enterprise Solutions for Interoperability (NESI). In this context, it is important to understand “defense” to also encompass natural disaster relief, homeland security, and other related national and global tasks because the diversity of those applications add remarkable complexity to the system requirements.

For example, in addition to the “typical” complexity associated with SOAs in other business contexts, understanding and incorporating the myriad of unique defense industry issues like huge data volumes, air and sea bandwidth challenges, scalability, diverse Communities of Interest (COI), life- and mission-critical data, and rugged security needs make accurate modeling and simulation crucial to the successful design and implementation of SOAs for such applications.

In the defense context, the SOA must also be designed to ensure robust, reliable operation in the most extreme environmental conditions (e.g., severe weather, power blackouts, erratic communication link lapses, natural disasters, etc.). Given the complexity of the architecture and multiple levels of system interactions, creating a valid and usable SoS model for this SOA

application using a single technique that captures the desired level of details can be a daunting task. In this paper we give details of a hybrid approach to modeling and simulation of a specific SOA that is named MCSOA and has been configured for potential defense applications.

In this approach, at the lowest system level (call it white-box level) we are using the formal network modeling tool Colored Petri Nets (CPNs) to model internal protocols, communications, and resource consumptions [1]. This allows us to validate the components of the system and identify weaknesses at the component level. For the interaction of the highest system level (call it black-box level), we are using a set of discrete-event simulation tools known as MESA/Extend [2]. With MESA/Extend, we can model the larger, much more complex interaction of large numbers of individual systems without the very tedious, complex, and potentially error-prone job of writing a component-by-component simulation of every system component with CPN.

The major advantage of this combination approach is that the assumptions at the black-box level are fully validated at the white-box level, which greatly simplifies and accelerates the design, development, and testing of the simulations of very large scale SOA-based Systems of Systems. The paper gives details of some of the models we have built under this approach and their simulation, verification and validation. Relevant details of MCSOA, CPN, and MESA/Extend are also included.

It should be noted that generalizations of our approach are possible. In fact, as we move towards a ubiquitous computing paradigm with complex systems and their interactions, the essence of software engineering should change to formulating, managing, and realizing models. For realistic software engineering, we will need *towers of models*, built using complex combinations of models [3].

## MCSOA ARCHITECTURE

MCSOA is Gestalt's implementation of a SOA (Service Oriented Architecture) [4]. It is outside the scope of this document to describe fully what a SOA is. However, we will provide basic terms observed in some SOA contexts of interest to us:

- The Service Consumer (C) is an entity requesting a service. The Consumer may know

the endpoint URL for a Provider of the service or it may rely on the broker to determine the actual endpoint URL.

- The Service Provider (P) is an entity providing the service.
- The Service Broker (SB) is an entity between the entity requesting the service (the Consumer) and the entity providing the Service (the Provider). The Broker may know of several provider entities that can provide the same service.

MCSOA is a development, deployment and discovery framework that can be used to build, deploy and run SOA applications. In such applications there are Consumers looking for Providers capable of satisfying the Consumer's needs. The MCSOA framework facilitates this process and plays the role of the Broker between Consumers and Providers. In this capacity, MCSOA functions as an Enterprise Service Bus (ESB) providing communication pathways between potentially large numbers of Consumers and Providers and other ESBs. Figure 1 shows the internal details of the MCSOA architecture.

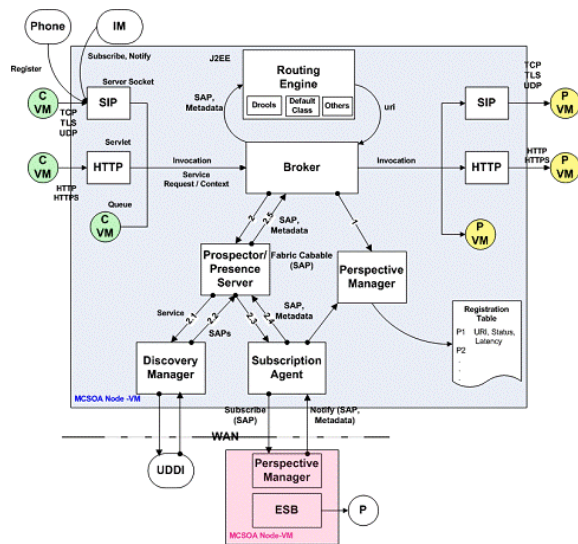


Figure 1. MCSOA Internals

A particular MCSOA node may locally know some of these and some may be remote and initially unknown to that MCSOA node. The communication channels between a MCSOA node, other MCSOA nodes, and consumers and providers may be constrained by conditions such as low bandwidth, high latency, and intermittent availability. MCSOA envisions context-based routing based on a process of "Discovery". A

“Routing Fabric” emerges through this process. The derived values associated with this fabric are:

- Fault Tolerance
- Distributed State
- Reliable Routing

Figure 2 contains a conceptual view of discovery and an inter-nodal fabric that can be used for dynamic/contextual routing.

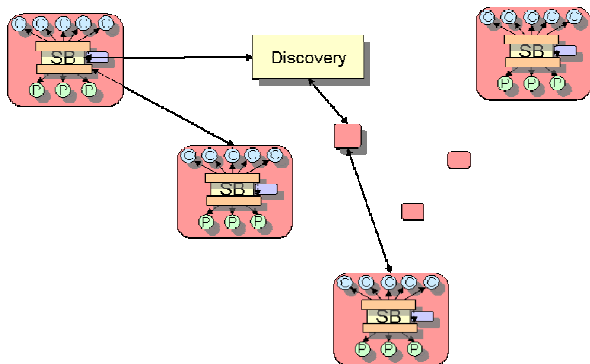


Figure 2. MCSOA Fabric Discovery

## MCSOA Presence Capabilities

The MCSOA discovery process is built on top of SIP-based presence protocol [5]. A key entity of this architecture is a Presence Server that represents the capability of MCSOA to handle presence notifications along with subscriptions for information about the presence (status) of entities known to MCSOA. An entity for which presence information is being tracked is termed a “presentity” – presence entity – and entities which are interested in presentity status information are called “watchers”.

The internal MCSOA Presence architecture is based on SIP. That is, SIP messages are used to establish subscriptions and to communicate presence changes. SIP REGISTER messages are used to establish initial contact with MCSOA. In MCSOA, the Presence Server and Registrar are coupled together. SIP NOTIFY messages are used to communicate presence information to Watchers. SIP SUBSCRIBE messages are used to declare an interest in presence information for particular presentities. Watchers send SUBSCRIBE messages to the Presence Server. The Presence Server sends a SIP NOTIFY response for each subscription request initially as well as whenever there is a change in the status of the subscribed presentity. The only presence

information currently maintained in the model, and in the MCSOA implementation on which the model is based, is whether an entity is registered or not. Presentities send REGISTER messages to the Presence Server. Registration is processed by the Registrar component. The Registrar notifies the Presence Server when a presentity’s presence status changes.

## MCSOA Discovery Process

As depicted in Figure 2, several MCSOA nodes may be inter-connected, however, they may be unaware of the capabilities in terms of services. The discovery process makes a MCSOA node aware of the capabilities of another MCSOA node. This information can then be used to achieve, among other things, contextual routing of service requests. Thus, if a consumer requests a service that is not known to its local service broker, the discovery process is initiated and through discovery the location of this provider is determined and the two broker nodes cooperate to route the consumer’s request and provider’s response between the consumer and provider. The key idea behind the discovery process is that, using the built-in presence capabilities, the local MCSOA node becomes a watcher for the availability of services at the remote MCSOA node.

## THE HYBRID MODELING APPROACH

As mentioned previously, our approach to modeling and validating MCSOA consists of creation of model of MCSOA internals using CPN which we call the *white-box* model. The fabric level interactions are modeled using MESA/Extend and we call this the *black-box* model. The white-box design follows standard Colored Petri Net conventions. The black-box design uses the MESA library within the Extend model discrete event simulation software.

## White-Box CPN Presence Model

The purpose of the MCSOA CPN Presence Model is to provide a detailed representation and simulation of MCSOA’s support for handling status (presence information) about entities interacting with a MCSOA node and each other. MCSOA maintains registration information about entities and entities can subscribe, through MCSOA, for updates to presence changes of other entities. CPN supports hierarchical

construction of models consisting of modules and submodules. The top-level CPN model of the MCSOA Presence component based on MCSOA internal architecture of Figure 1 is depicted in Figure 3. This top-level abstract view of the system shows two primary components: WatchersAndPresentities and RegistrarAndPresenceServer.

Each of these components is outlined with a thick line representing that it possesses additional functionality that can be located in its submodules. Watchers and Presentities are user agents that generate registration and subscription requests that are examined and routed by the Client and Server Transaction in the WatchersAndPresentities side to the Client and Server Transaction in the RegistrarAndPresenceServer side. One of the submodules of this model responsible for processing of new subscription requests is shown in Figure 4. This submodule presents the steps that a new subscription request goes through. An incoming subscription request makes AcceptSUBSendOK transition enabled. Firing this transition checks whether the watcher participating in the subscription request is already in REGTable.

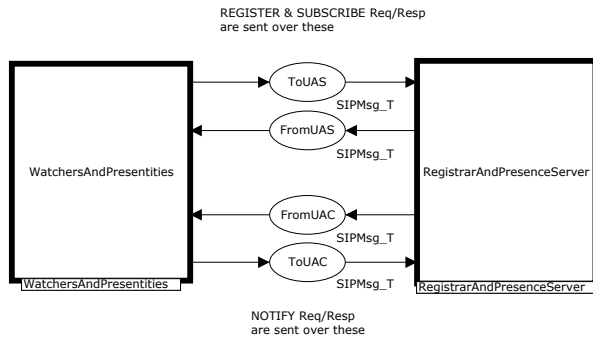


Figure 3. Top Module of CPN Presence Model

This page processes a new subscription by assuring that the watcher who has sent a subscription request is still in the REGTable. If it is not, a response with the code 404 is sent to the watcher. Before a notification is sent to the watcher about the current status of the presentity, the presence of the watcher in REGTable is checked again. If it is not there, notification is not sent.

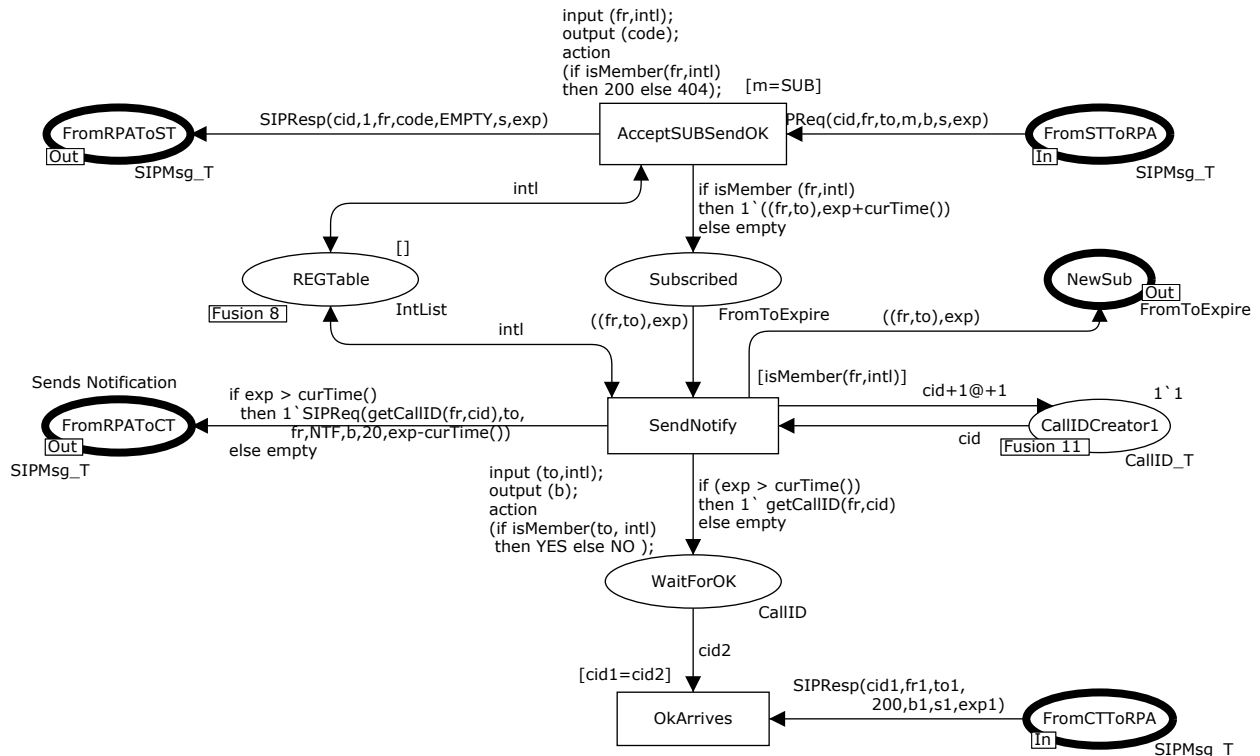


Figure 4. The ProcessNewSUB Submodule

If the watcher is in REGTable the SIP response with the code 200 is sent to the watcher as an indication that everything is OK. If the watcher is not in the REGTable, the response with the code 404 is sent to the watcher. If watcher is in REGTable, firing AcceptSUBSendOK transition puts a token in the Subscribed place which enables SendNotify transition. Firing this transition sends a SIP notification request to the watcher and a token is placed in WaitForOK place. This token remains until a response is received from client transaction. In addition to sending notification, the firing of SendNotify transition forwards the subscription request to the ProcessExpSUB module responsible for handling expiration of subscriptions. At the lowest level of this hierarchical model are the SIP client and server transactions.

The detailed model of Client Transaction is shown in Figure 5. A complete description of this model is beyond the scope of this paper. Some details and discussion of the Presence model and its validation are given in [6].

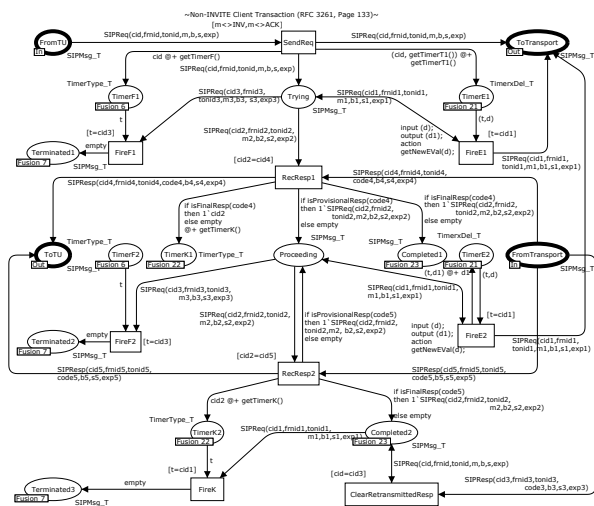


Figure 5. Presence Client Transaction Submodule

### Black-Box Extend/MESA Fabric Model

The purpose of the black-box Fabric Discovery and Routing Model is to focus on inter-nodal behaviors among MCSOA nodes, Consumers and Providers interoperating in a fabric. This model concentrates on the high-level message flow between nodes and the simulation of realistic behaviors observed in a real world interconnection among computers serving as consumers and

providers of services and service brokers facilitating message exchange between consumers and providers. This model has been designed and created using MESA (Modeling Environment for SOA Analysis) and Extend and simulated using the standard functionality and monitoring facilities available in MESA. The model offers a black-box view of the nodes themselves and the ability to interconnect multiple nodes in a fabric. The process of creating a MESA model conceptually consists of the following steps:

- Define node names
- Define service names
- Create model layout from MESA node library
- Define Service definitions

Some of these steps are facilitated by MESA user interface extensions to Extend and some of them are performed using Extend model creation actions.

The current fabric model consists of 15 nodes, which is described using three elements: a node table, services table and graphical model. Figure 6 contains the MESA table describing nodes in the fabric model. Each node is identified by name and by NodeID. Machine properties of each node are adjustable by changing values in the other columns. In the figure all nodes are defined to have "Default Server" properties and are shown to be available.

Node	NodeID	Available	Sniffer	Lan	ServerType	ServerSp
1	Default	1	Off	Off	Default Server (SF)	-1
2	Cons_1	2	1	Off	Default Server (SF)	-1
3	Provider_1	3	1	Off	Default Server (SF)	-1
4	MCSOA_1	4	1	Off	Default Server (SF)	-1
5	MCSOA_2	5	1	Off	Default Server (SF)	-1
6	UDDI	6	1	Off	Default Server (SF)	-1
7	Provider_2	7	1	Off	Default Server (SF)	-1
8	Provider_4	8	1	Off	Default Server (SF)	-1
9	WAN_1	9	1	Off	Default Server (SF)	-1
10	Cons_2	10	1	Off	Default Server (SF)	-1
11	Provider_3	11	1	Off	Default Server (SF)	-1
12	MCSOA_3	12	1	Off	Default Server (SF)	-1
13	WAN_2	13	1	Off	Default Server (SF)	-1
14	WAN_3	14	1	Off	Default Server (SF)	-1
15	WAN_0	15	1	Off	Default Server (SF)	-1
16	WAN_4	16	1	Off	Default Server (SF)	-1

Figure 6. Fabric Model Node Table

MESA Services are listed in summary form in the MESA Directory Services table as shown in Figure 7. Like nodes, services are identified by both name and ServiceID. When a MESA model executes a service, it is modeled to run at a

particular model node. Such mappings can be declared globally or locally. In this services table, the Global Name column defines which node will ordinarily be the execution spot for each service.

Service	ServiceID	Global Name	Global Map	Service Type
1 Delay	1		0	Serial
2 LocalService	2	MCSOA_1	0	Serial
3 RemoteService	3	MCSOA_1	0	Serial
4 LocalServiceActual	4	Provider_1	0	Serial
5 RemoteServiceActual	5	Provider_2	0	Serial
6 LookupService	6	UDDI	0	Serial
7 RemoteRequest	7	Cons_1	0	Serial
8 LocalRequest	8	Cons_1	0	Serial
9 Notify	9	MCSOA_2	0	Serial
10 ManagedServiceActual	10	Provider_4	0	Serial
11 RemotePassThru	11	MCSOA_2	0	Serial
12 ManagedService	12	MCSOA_1	0	Serial
13 ManagedRequest	13	Cons_1	0	Serial
14 Subscribe	14	MCSOA_2	0	Serial
15 NotRegistered	15	MCSOA_1	0	Serial
16 NotReachable	16	MCSOA_1	0	Serial
17 Discovery	17	MCSOA_1	0	Serial
18 HandleRemoteRoute	18	MCSOA_1	0	Serial
19 HandleNotify	19	MCSOA_2	0	Serial

Figure 7. Fabric Model Services Table

When services are invoked at a particular node, a Map table for that node can override the global mappings but it need not do so. For example, as shown in Figure 7, both the LocalService and RemoteService services will run on MCSOA\_1 while the LocalRequest and RemoteRequest services will run on Cons\_1 and the LookupService will run on the UDDI node. Complete description of the black-box fabric model and its validation results are contained in [7].

The overall graphical view of these nodes and their interconnections is given in Figure 8. There are two consumer nodes: Cons\_1 and Cons\_2; four provider nodes: Provider\_1, Provider\_2, Provider\_3 and Provider\_4; three MCSOA nodes: MCSOA\_1, MCSOA\_2, and MCSOA\_3; a UDDI node and several nodes to represent the fact that parts of the fabric may themselves be interconnected over a Wide Area Network (WAN). There are several WAN nodes so that multiple network “hops” can be traversed in the course of consumer’s request for a provider’s service.

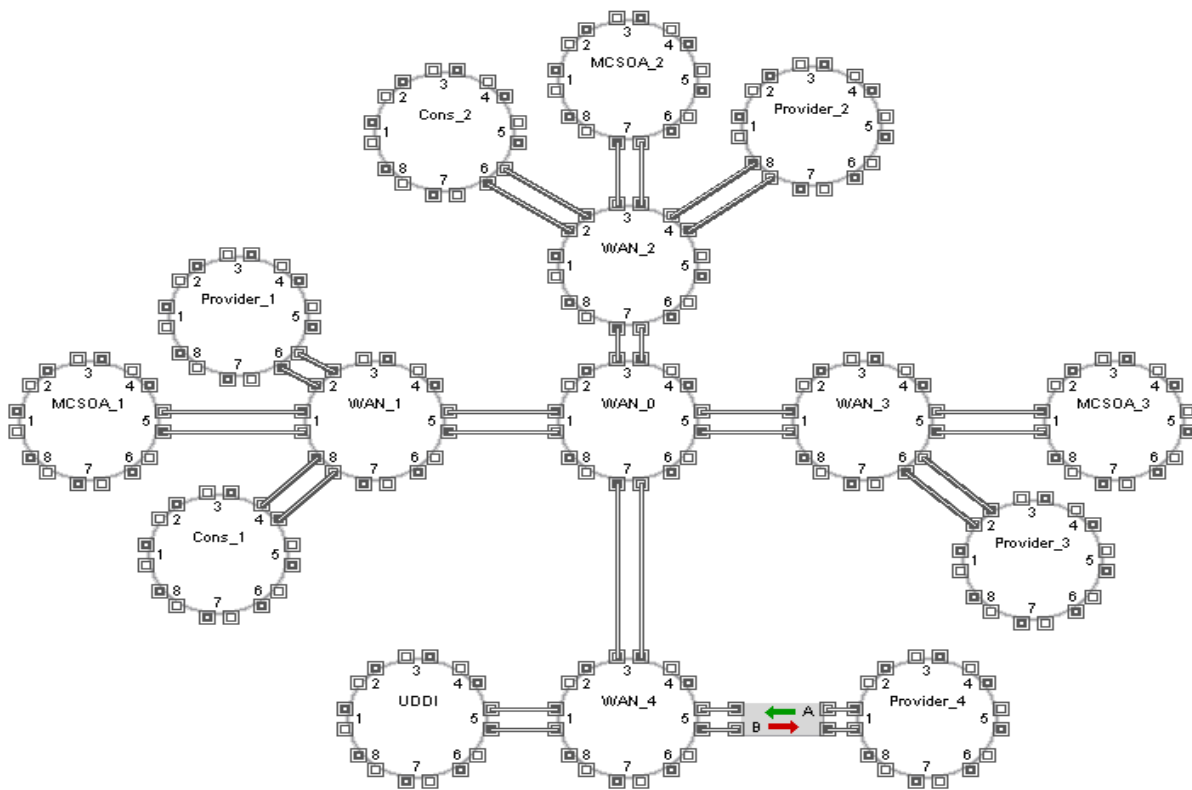


Figure 8. MESA/Extend Fabric Model

## CONCLUSIONS

We presented a hybrid approach to modeling a system of systems such as a Service Oriented Architecture (SOA) using Colored Petri Nets and Extend/MESA by separating the levels of details for each. The particular SOA we examined consisted of inter-connected homogeneous nodes. The interactions of these nodes were dependent on the internal details of protocols employed. This allowed us to separate internal communications from the external communications and model and validate the fabric level behavior under the assumption that the internal protocol and its workings were validated. With this approach we were able to make a rapid progress not only in terms of model creation but also in terms of validation. This was especially important since our project was time-boxed. This decoupling also had its benefits in terms of overall model maintenance and refinements. The white-box level model could be refined without affecting the black box model and vice versa. For example, if MCSOA were to use the Extensible Messaging and Presence Protocol (XMPP), only the white-box model will need to be revalidated. We would like to add that as we move towards building more and more complex systems, a hybrid approach to modeling and analyzing such systems would make the task manageable.

## ACKNOWLEDGEMENTS

This Advanced Research for Computing Enterprise Services (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] K. Jensen, *Coloured Petri Nets*, Volume 1 of Monographs in Theoretical Computer Science, Springer-Verlag, 2nd edition, 1996.
- [2] D. Krahl, "The Extend Simulation Environment," *Proc. 2002 Winter Simulation Conference*, pp. 205-213.
- [3] R. Milner, "Memories of Gilles Kahn, and the informatic future," Online at <http://www.inria.fr/gilleskahn/presentation/milner.pdf>
- [4] Multi-Channel Service Oriented Architecture, Internal Report, Gestalt LLC, April 2005.
- [5] J. Rosenberg et al., SIP: Session Initiation Protocol, RFC 3261. IETF, June 2002.
- [6] V. Gehlot and A. Hayrapetyan, "A Formalized and Validated Executable Model of the SIP-Based Presence Protocol for Mobile Applications". *Proc. 45th ACM Southeast Conference (ACMSE)*, 2007 (to appear).
- [7] ARCES Reports, Release 3, Villanova University and Gestalt LLC, August 2006.