**14TH ICCRTS**

"**C2 and Agility**"

**Paper # 157_S**

1. Title: **Designing and Evaluating Agile C2 Systems Based on Service Oriented Architectures**

2. Topic(s): **Track 9: C2 Architectures and Technologies**

3. Authors: **Ashraf M Abusharekh [Student*], Lee W. Wagenhals, and Alexander H. Levis**

4. Point of Contact: **Ashraf M Abusharekh**

5. Organization: **System Architectures Lab, George Mason University**

6. Address: **System Architectures Laboratory**

**Electrical and Computer Engineering Department, MSN 1G5**

**George Mason University**

**Fairfax, VA - 22030**

**703.993.1725 (v)**

**703.993.1601 (f)**

7. Email: **aabushar@gmu.edu**

***Mr. Abusharekh is a third year PhD. student in the The Volgenau School of Information Technology and Engineering at George Mason University.**

## Abstract

The Department of Defense Architecture Framework (DoDAF) [1] is one of the frameworks used to describe, document and organize an architecture, its components, their relationships and the principles and guidelines governing their design and evolution [1]. There are two fundamental approaches for designing architectures that are conformant to DoDAF, one based on structured analysis and one on object orientation [2] [3] [4] [5]. Recently, a new direction was introduced in DoDAF v1.5, which transforms DoDAF from product centricity to data centricity by using the Service Oriented Architecture (SOA) paradigm as a key enabler for implementing net-centric objectives [6]. SOA comes with the promise of solving enterprise interoperability, business and IT agility, and flexibility and resource reuse problems currently inherited from stovepipe architectures, thus allowing for a net-centric system-of-systems. A key to Command and Control (C2) agility is effective information sharing. By using SOA, alignment between business objectives and the supporting IT infrastructure is achieved and information sharing becomes the norm and data and information become ubiquitous thus allowing for an agile C2 system.

In this paper, we present an approach for constructing an event driven SOA compliant to DoDAF v1.5 capable of participating in the DoD net-centric environment (NCE) by consuming NCE services and/or serving un-anticipated NCE users. The end product of the approach is an executable model derived from the information contained in the DoDAF v1.5 artifacts to evaluate the logical, behavior and performance characteristics of architecture's business processes and services. The approach will be described using an illustrative case study.

## 1. Introduction

Systems architecting [7] [8] [2] is part of the system engineering process and relies on many of the methodologies that have been developed over time. Its end product is a detailed description of the system architecture to be instantiated. DoDAF [1] is one of the frameworks available for system architects to describe, document and organize an architecture, its components, their relationships and the principles and guidelines governing their design and evolution [1].

Levis & Wagenhals [2] introduced two approaches for designing architectures that are conformant to DoDAF, one based on structured analysis and one on object orientation. The end product in both cases is an executable model derived from the information contained in the DoDAF artifacts.

Recently, a set of concepts, objectives and strategies has been defined within the Department of Defense community to transform the department to a new type of information intensive warfare known as Net-Centric Warfare (NCW) capable of achieving Net-Centric Operations (NCO) in which the war-fighting enterprise is effectively networked making essential information and capabilities available (when, where and how they need it) for users (current and un-anticipated) to carry out their missions effectively. The DoD views architectures as the mechanism for designing solutions for this transformation, and SOA has been selected as a design paradigm to build such architectures capable of achieving many of the goals of this transformation. This new direction was reflected in DoDAF v1.5, which accordingly transforms DoDAF from product centricity to data centricity and uses the SOA paradigm as a key enabler for implementing net-centric objectives [1].

In this paper, we present an approach for constructing an event driven SOA compliant to DoDAF v1.5 capable of participating in the NCE by consuming NCE existing capabilities and/or populating NCE with new ones that can be consumed by anticipated and un-anticipated users. This architecture will define business services and business processes that are hosted by an Enterprise Service Bus (ESB)-based SOA infrastructure. These business services and processes will accomplish the architecture's operational concept; and they will be published through Communities of Interest (COI) as capabilities for other NCE users to reuse to accomplish their missions. The end product of the approach is an executable model

derived from the information contained in the DoDAF v1.5 artifacts to evaluate the logical, behavior and performance characteristics of architecture's business processes and services.

Relevant background about NCW concepts and strategies, SOA and DoDAF v1.5 is presented in section 2. The approach is presented in section 3; section 4 includes an illustrative example and a discussion of the performance analysis and the results of the computational experiment conducted. Finally conclusions and future work are presented in Section 5.

## 2. Background

The NCE is a networked environment that includes infrastructure, systems, processes and people to achieve improved situation awareness, better access to business information and a shortened decision cycle in support of NCO. Data and capabilities across the NCE should be visible, accessible and usable by anticipated and un-anticipated users. Data exchange between systems should be flexible supporting interoperability between them in a loosely coupled way avoiding point-to-point interfaces among them. In the NCE, users should be able to obtain information and capabilities hosted by the NCE when, where and in the form they need to accomplish their missions and business objectives [1].

The DoD community identified a set of high-level net-centric concepts [1], strategies [6] [9], and goals to support the transition to NCO. The NCE concepts are: (1) Populate NCE: populating NCE with data, information and capabilities that are made visible, discoverable and accessible by authorized NCE users. New net-centric architectures should provide/contribute to the NCE data, information and capabilities which can be leveraged by other NCE users. (2) Utilize the NCE: users of the NCE should be able to discover data, information and capabilities and use them to accomplish their missions. (3) Accommodate un-anticipated users: NCE users will look for (discover) data, information, and capabilities in the NCE rather than be constrained (hard-wired) to them. (4) Promote the use of Communities of Interest: As defined by the DoD Net-Centric Data Strategy [6] a COI is a collaborative group of users who exchange information in pursuit of their shared goals, interests, missions, or business processes and who therefore must have shared vocabulary for the information they exchange. They define common vocabularies, taxonomies, data standards, interchange agreements and specifications relevant to the communities' architecture [1]. COIs ensures that data, information and capabilities are developed in a manner that supports interoperability across organizational boundaries. (5) Support Shared Infrastructure: enterprise-level data, information and capabilities are being supported and used where appropriate and available. Two strategies have been developed in support of the NCE concepts, the Net-Centric Data Strategy [6] and the Net-Centric Service Strategy. The main goals of the Data Strategy are to making data visible, accessible, understandable, and trusted. The main goals of the Service Strategy are to provide and consume services from the NCE, govern these services and their infrastructure, and monitor and manage then.

SOA has been selected as an approach for implementing the net-centric concepts and objectives. The service-oriented paradigm was introduced with the promise that it will solve enterprise interoperability, business and IT agility and flexibility, and resource reuse problems currently inherited from stovepipe architectures. SOA can mean different things to different people [10] (even at different stages of the architecture design process). From the system architect point of view, SOA is an architectural style that requires a service provider, a service consumer, and a service description; the resulting architecture is SOA, and the supporting infrastructure is SOA. For the purposes of this paper, SOA is based on coarse-grained, loosely coupled, and reusable services [11] [12]. These services can be composed into business processes, and they interact through the ESB which provides a highly distributed, event-driven SOA that combines Message Oriented Middleware (MOM), orchestration and process flow, and intelligent routing based on content and data transformation. Although SOA can exist without an ESB, using an ESB makes it more efficient, scalable and more reliable [13]. The layered structure of the SOA environment is presented and explained in detail in the next section.

DoDAF v1.5, is the first transformation of the DoD Architecture Framework to support representing the net-centric architectural constructs that can capture the NCE concepts mentioned above. The reader is advised to refer to [1] for detailed description of the DoDAF artifacts.

## 3. Approach

The structure of the SOA environment is shown in Fig. 1. The environment is composed of three layers. The Operational layer contains the business processes, defined as a composition of singleton and/or composite business services. The Service layer contains the services and individual ESBs participating in a SOA federation; each ESB is composed of the MOM responsible for passing messages between services using message flow channels, the SOA Supervisor responsible for monitoring services and business processes, the Orchestration service responsible for executing business processes, the Registry service maintaining service definitions, service level agreements, and business process definitions, and the Service Containers that control and monitor the services they host by managing services' endpoints and sending periodic messages to the SOA Supervisor. SOA federation rules and policies will be enforced using border gateways interconnecting individual ESBs at the Service layer. The hosting, instantiation, orchestration, management and monitoring of the business processes defined at the Operational Layer are done at the Service Layer by the ESBs.

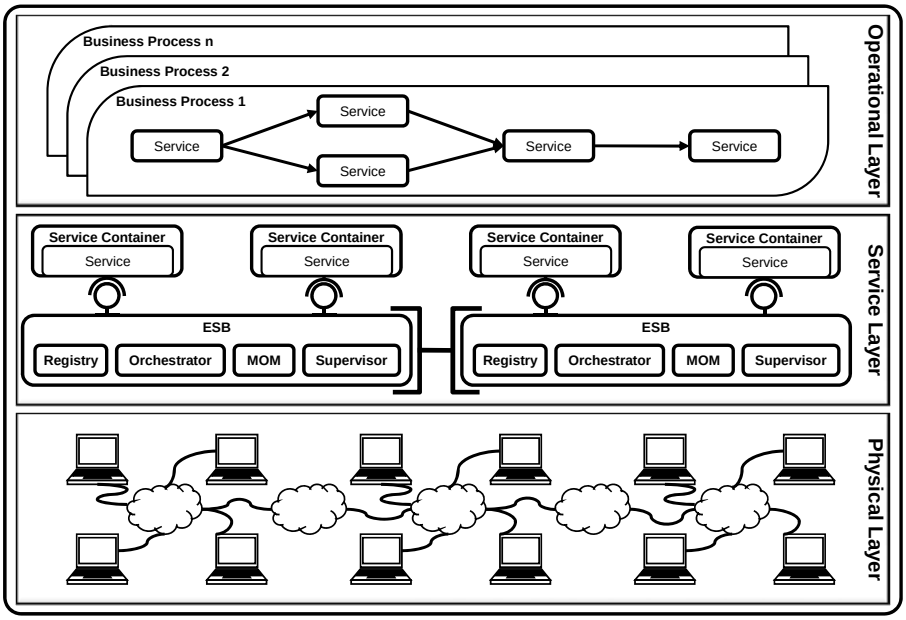


**Figure 1: SOA Environment**

The Physical layer contains physical nodes (workstations, servers, network nodes, etc…) and communication networks supporting the environment.

The interoperability between SOAs participating in a federation is depicted in Fig. 1 as a link between the ESBs. This link is a virtual link and does not indicate tight coupling of the ESBs. Its purpose is to show that the SOAs are part of a federation. In order to enable a SOA federation, a higher level form of a federation repository/registry is needed to expose federation rules, policies, and federation-wide services [14]. This repository is not shown in Fig. 1 and will be further discussed in the next paragraphs.

SOA federation is an environment that brings together multiple SOAs that have established producer-consumer relationships, such that the right rules and policies (trust, governance, security, etc…) apply

throughout the environment. This allows for local variances and autonomy of individual SOAs while implementing federation-wide rules and policies to regulate and govern interoperability, thus addressing organizational, management, political and practical issues [12] [15] [16]. A SOA participating in a SOA federation is referred to as a federated SOA. In the context of this paper, the SOA under construction will federate with existing SOAs to use their published capabilities, i.e., the new SOA is the consumer of services exposed by existing SOAs, and it will produce new capabilities to be consumed by other systems.

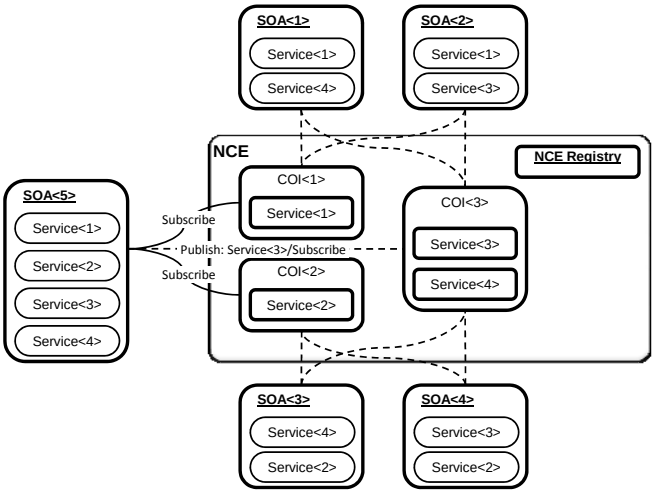


**Figure 2: NCE, COIs and SOA Federation**

The concept of COIs, will be used to enable dynamic federation with pre-defined or un-anticipated systems. In order to simplify and speed-up discovery of services (capabilities), 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 picture of the SOA federation within the NCE and how the notion of COI enables and supports such environment, an example is depicted in Fig. 2Figure 2.

In Fig. 2, SOA<1> and SOA<2> participate in COI<1> by publishing Service<1>, SOA<3> and SOA<4> participate in COI<2> by publishing Service<2>, and SOA<1>, SOA<2>, SOA<3>, SOA<4> and SOA<5> participate in COI<3> by publishing services Service<3> and Service<4>. SOA<5> needs Service<1>, Service<2>, and Service<4> in its business processes, therefore it subscribes to COI<1>, COI<2> and COI<3>. Furthermore, if SOA<5> can substitute its implementation of service<3> by federating with members of COI<3> in case its own implementation is failing. From an industrial view point, several problems with this approach need to be resolved, e.g., the location of a COI repository which should be negotiated and agreed upon among participating parties. Since this study is primarily focusing on DoD NCO, the NCE is assumed to host the COIs and, the NCE registry is the central federation repository/registry that publishes COIs information as shown in Fig. 3. An abstract layered structure of the information that needs to be maintained in the NCE registry about the COIs is shown in Fig. 3.

In the layered approach shown in Fig. 3, layer 1 shows all the COIs hosted by the NCE, layer 2 shows all services definitions published through individual COIs, and layer 3 shows the actual services implemented by SOA instances. Although layer 3 contains different instantiations, still the providers abide by the policies and rules agreed upon through the COI including service definitions and data models. The NCE registry will be used by a SOA to search for relevant COI, but in order to subscribe to a particular COI, the SOA instance needs to subscribe to the specific COI registry such that each COI will

maintain its own autonomy and self containment. The SOA instance will subscribe to a COI as a producer adding new capabilities to be published within the COI, or as a consumer of a capabilities being published by the COI or as both. In all cases, the concept of a COI is leveraged to support and enable the dynamic formation of federations among deployed SOA instances. In Fig. 3 when service failures occurs, to locate an alternative service implementation for a failing service e.g. COI<2>.Service<2>.Service<c>, the only candidate services that need to be further examined for acceptable service levels are the ones under the same COI Service Description, i.e. COI<2>.Service<2>.Service<a> and COI<2>.Service<2>.Service<b>, since they by default provide the same service as the original one.

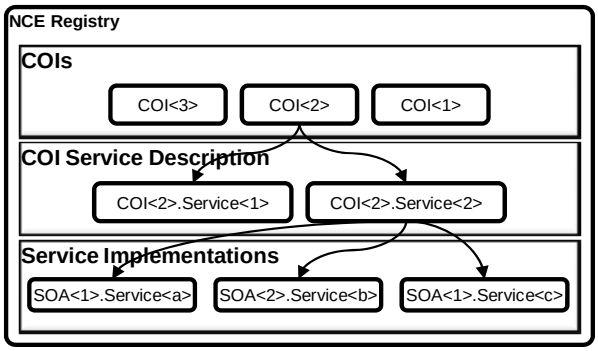


**Figure 3: NCE Registry**

A framework for achieving a reliable architecture description with rigorous evaluation was first introduced in [4]. A similar framework suitable for net-centric architectures is shown in Fig. 4. This framework provides a high level view of the life cycle of a net-centric architecture. As mentioned in [5], three processes need to be addressed, a process for creating the architecture description applied at the Architecture Design Phase, a process for converting the architecture description to the executable model, and a process for using the executable model for analysis and evaluation both applied at the Analysis & Evaluation Phase.

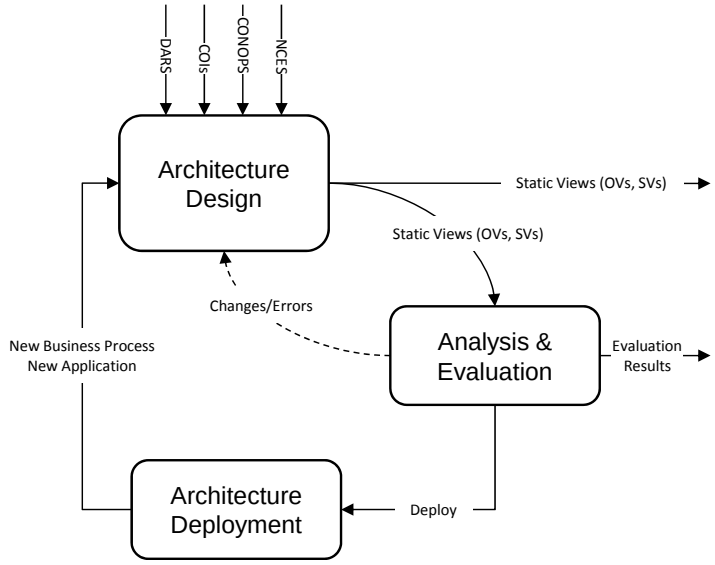


**Figure 4: Architecting Process**

The Architecture Design phase produces the SOA architecture DoDAF v1.5 products, which are used in the Analysis & Evaluation phase to synthesize executable model. Two approaches, one based on

structured analysis and one on object orientation, have been developed to support an architecture creating process [4] [3] [17]. Additional information and information sources are need during the Architecture Design Phase for these approaches to be able to construct a net-centric architecture. In the context of NCE, the new architecture is not only viewed as an un-anticipated user of existing capabilities implemented by other deployed systems, but also as a source of new capabilities. From the system architect viewpoint, he needs to design a new architecture to accomplish its intended mission/s, by leveraging -as needed- capabilities implemented by existing systems, and by populating the NCE with new capabilities that will contribute to the NCO and be used by other systems. Accordingly, three additional information sources are needed. (1) Information about existing COIs and the services they expose: the system architect will need to be aware of existing COIs in order to be able to consume capabilities and to publish new ones. Full understanding of the COIs policies and rules, their data formats and services descriptions is needed to successfully federate with their members. The new architecture will abide by the existing COI policies, rules and data and services definitions agreed upon between the COI's members in order for consumers to successfully use the architectures capabilities exposed through the COI. The NCE Registry Service will be the main source of such information (Fig. 3). (2) Information about architectures of systems implementing capabilities that might be leveraged by the new architecture: the architect needs to fully understand the capabilities exposed by other systems in order to make a decision whether or not they will full the functional and non-functional requirements of the new architecture. The DoD Architecture Repository (DARS) provides an environment for registering, posting, discovering and retrieving architecture related information [1]. In addition, the system architect has to expose the new architecture through DARS for other NCE users to make use of it. (3) Access to existing Net-Centric Enterprise Services (NCES) currently available through the NCE: The NCES will allow for trustworthy enterprise-level data, information and capability sharing.

A modified architecture creation process using the object oriented approach similar to the one explained in [5] is depicted in Fig. 5. The modified process uses the additional information and information sources mentioned above to build a net-centric architecture. The reader is advised to refer to [5] for a full description of the stages of the process. The process is divided into six stages as follows:

- *Stage 0*: includes the articulation of the purpose and scope of the architecture and the identification of the background documentation needed to create the architecture. Three new source of information the architect need to have access to at this stage:

    1. DARS.

    2. Available COIs.

    3. Available NCES.

- *Stage 1*: focuses on the development of the operational concept of the architecture. Initial depiction of COIs that the architecture will participate in.

- *Stage 2*: includes three main activities that need the additional information to build the net-centric architecture.

    1. Defining Organizations and their relationships: this activity defines organizations and sub-organizations within the architecture along with net-centric organizations that federate with the architecture providing the net-centric objectives. At this stage, the architect needs to identify the existing COIs that will provide or consume capabilities and their relationships with the internal organizations of the architecture. DARS is needed to further examine the architecture artifacts of potential existing organizations participating in these COIs. An initial examination of whether

2. Defining Operational nodes: apart from the architecture's internal operational nodes, the choice of COIs that the architecture will participate in will introduce external operational nodes provided by members of the COIs. DARS is used to examine the artifacts describing such nodes.

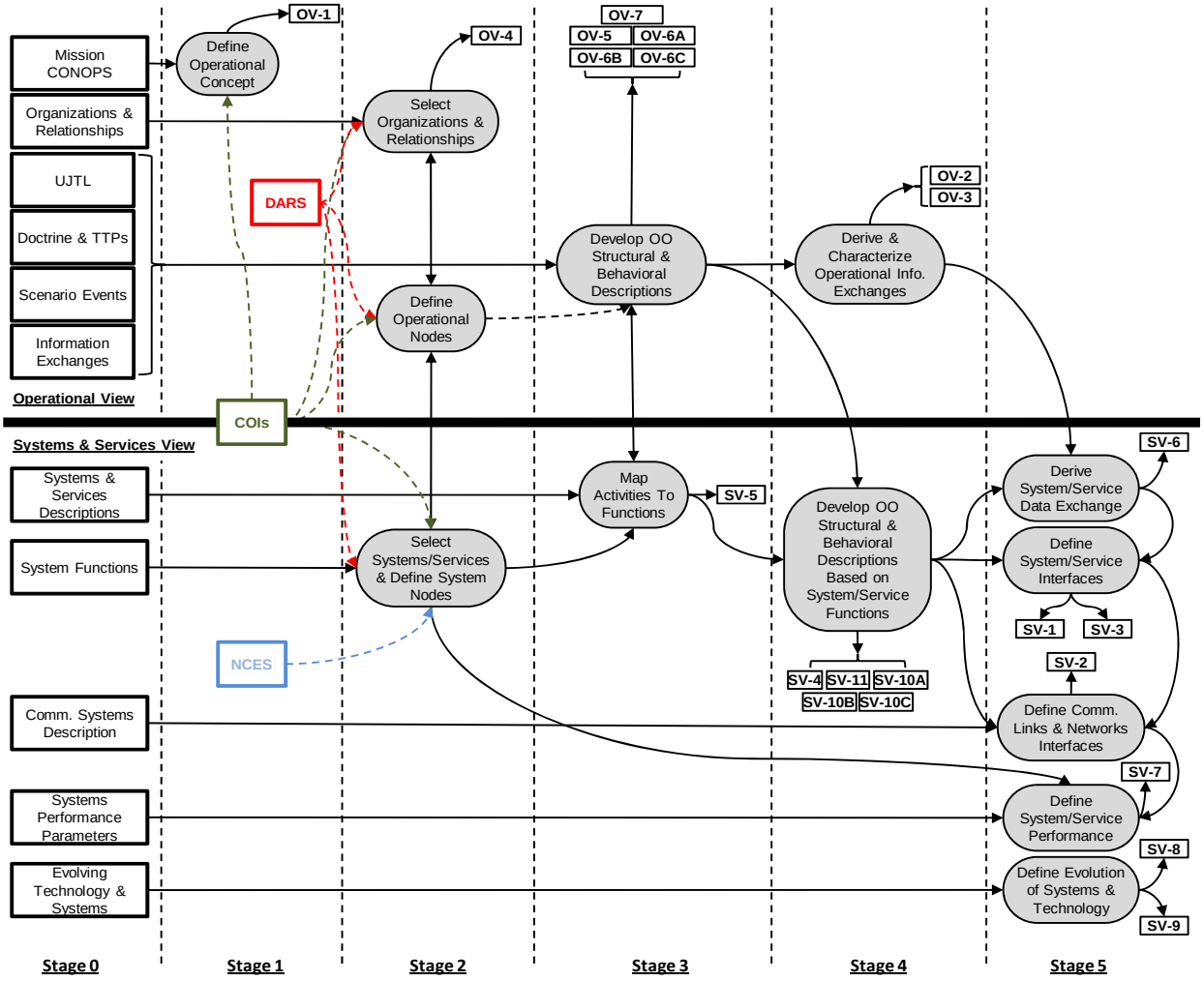


**Figure 5: DoDAF Architecture Design**

3. Selecting Systems/Services and defining System Nodes: the architect identifies specific services that will be consumed or produced as part of existing COIs the architecture is participating in. The architect needs to abide by data descriptions and service interfaces published by those COIs.

The information articulated in *Stage 2* will affect all stages of the process. These affected DoDAF artifacts will be presented with the help of the case study in the following section.

- *Stage 3*: involves:

1. Full analysis of the Operational View. It is during this stage that structural and behavioral diagrams are developed to understand and describe the operational activities carried out by

2. Develop mappings from the operational activities to the systems, services and system functions.

- *Stage 4*: involves

1. Finalizing the Operational View analysis by deriving and characterizing the operational information exchanges.

2. Full analysis of the Systems and Services View, focusing on services, system components and their functions along with system data that is exchanged.

- *Stage 5*: involves

1. Finalizing the Systems and Services View products by extracting data and concepts from *Stage 4* analysis and generating systems and services interface descriptions, the communications infrastructure description, the systems and services performance parameters documentation, and the system, service, and technology evolution descriptions.

The products of the Architecture Design phase will be used in the Analysis & Evaluation Phase to construct an executable model. DoDAF artifacts are static representation of an unprecedented, complex, and dynamic architecture. These static artifacts are capable of describing the behavior of the architecture only in a limited way. Hence there is a strong need for architecture evaluation techniques that go beyond static diagrams to examine behavior and performance in detail. An executable model of the architecture enables the architect to analyze its dynamic behavior, identify logical and behavioral errors not easily seen in the static descriptions, and demonstrate to the customer or user the capabilities that the architecture enables.

A process for converting the architectural artifacts produced during the Architecture Design Phase to an executable model is needed at this phase; we will use the existing process defined in [4] and [5] in which Colored Peti-Net (CPN) executable model of the architecture is synthesized from the DoDAF artifacts. CPNs offer more than just simulation to support the analysis and evaluation. CPNs in general (and CPNTools [18] in particular) allow behavioral properties to be verified by analysis without resorting to simulation. State Space Analysis is an analysis technique that provides a variety of properties about a CPN [19]. State Space Analysis techniques have been implemented in CPNTools.

A key concept in the Analysis & Evaluation Phase is that all elements of the executable model must be traceable to elements in the architecture description. As more is learned about the behavior and performance of the architecture from the creation of the executable model and the detailed analysis of its behavior and performance, any corrections or changes introduced to the executable model are reflected back in the DoDAF products. It is during this phase that the designer makes sure that a new process provides dependable service level within the architecture whether the architecture is new or already deployed. The Analysis & Evaluation phase will also produce the evaluation of Measures of Performance (MOPs) and Measures of Effectiveness (MOEs) of the architecture [20].

Finally, the architecture is instantiated and deployed in the Architecture Deployment phase, in which the need for new processes, applications or services will trigger the design process again. Exploring the implications of adding new business processes (implications on the instance itself and other systems reusing the instance's capabilities) while SOA instance is deployed through modeling and simulation is also necessary before the actual deployment of the new business logic.

## 4. Case Study

The case study presented here is based on a hypothetical operational concept for a new Theater Ballistic Missile Defense (TBMD) system called Airborne Theater Ballistic Missile Interceptor System (ATIS). The case study was first introduced by [5] to illustrate the process of completing a full DoDAF v1.5 compliant architecture. Our goal is to fully create a DoDAF v1.5 net-centric architecture capable of intercepting and destroying Theater Ballistic Missile (TBMs) and capable of providing and consuming information and capabilities to and from the NCE. To achieve this, we will:

1. Define business services and business processes to be hosted by the ATIS (internal capabilities),

2. Re-use business services and/or processes implemented by other NCE systems (external capabilities).

3. Publishing relevant ATIS capabilities to be used by other NCE systems.

To re-use existing capabilities and populate NCE with new ones, ATIS must join relevant COIs. ATIS business services and processes will be hosted by the SOA environment shown in Fig. 1.

For simplicity and brevity, only key DoDAF v1.5 artifacts will be presented. The rest of the products will not be presented and the reader is referred to [5] for a richer set of DoDAF artifacts. In addition, the following assumptions have been made:

1. The following COIs exist:

    a. Ballistic Missile Response COI

    b. Intelligence, Surveillance, and Reconnaissance (ISR) COI

2. A Ballistic Launch Warning System (BLWS) is deployed and is hosting a Global Ballistic Missile Warning (GBMW) Service. BLWS is a member of the Ballistic Missile Response COI and has already published the GBMW Service through the COI.

3. Net-Centric Enterprise Services (NCES) and capabilities are available and accessible.

The purpose of the architecture is to develop an understanding of the arrangement and interoperation of organizations and systems that support the concept of operations for ATIS, and to be able to assess the ability of the proposed system to destroy incoming TBMs based on the capabilities of the adversary to launch them. The architecture is designed to (1) determine if the operational concept can be made to work, (2) assess the impact of evolving this system into the NCE by creating business services of its own and (3) determine how to make its business services or their composition (business processes) accessible by anticipated and un-anticipated users of the NCE. The operational concept graphic of the architecture (OV-1) is shown in Fig. 6.

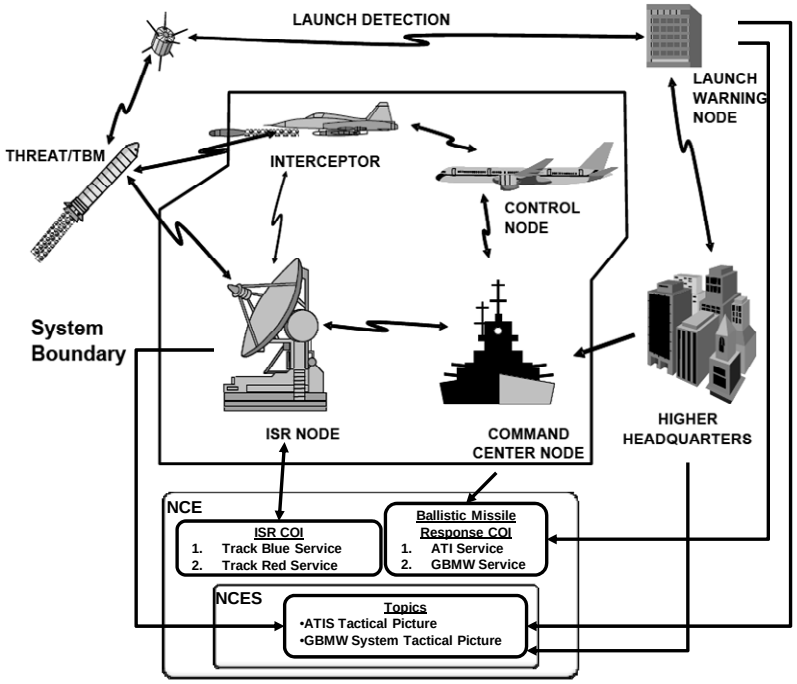


**Figure 6: Operational Concept (OV-1)**

OV-1 shows that ATIS will participate in two COIs:

1. ISR COI by contributing to two services, *TrackBlue* Service and *TrackRed* Service. Due to the critical mission of the ATIS it was decided that the ATIS will have its own ISR capabilities instead of leveraging ISR capabilities of other NCE systems, but in the case of failures or degradation of service levels, ATIS can re-use existing ISR capabilities through the ISR COI. Also other systems can leverage the ATIS ISR capabilities if ATIS is not engaged.

2. Ballistic Missile Response COI by contributing the *ATIS* Service as a composite service that exposes the ATIS main business process. The *ATIS* Service can be triggered by authorized users to react to a launched TBM; in this case the user is pre-defined as the BLWS *GBMW* Service which detects the TBM launch and triggers *ATIS* Service to destroy it. The interoperability between BLWS and the ATIS is facilitated through the Ballistic Missile Response COI.

The NCE will govern, manage, and monitor the COIs (and their services), and furthermore ATIS will utilize the NCES to publish its Tactical Picture as a Topic (ATIS-TPT). Figure 6 also shows that an external system, Higher Headquarters (HHQ) subscribes to get the ATIS-TPT. Propagation of the ATIS-TPT is done outside the ATIS boundary, i.e. ATIS will only post its Tactical Picture and the NCES will take it from there. Users interested in the ATIS-TPT are considered by ATIS as un-anticipated.

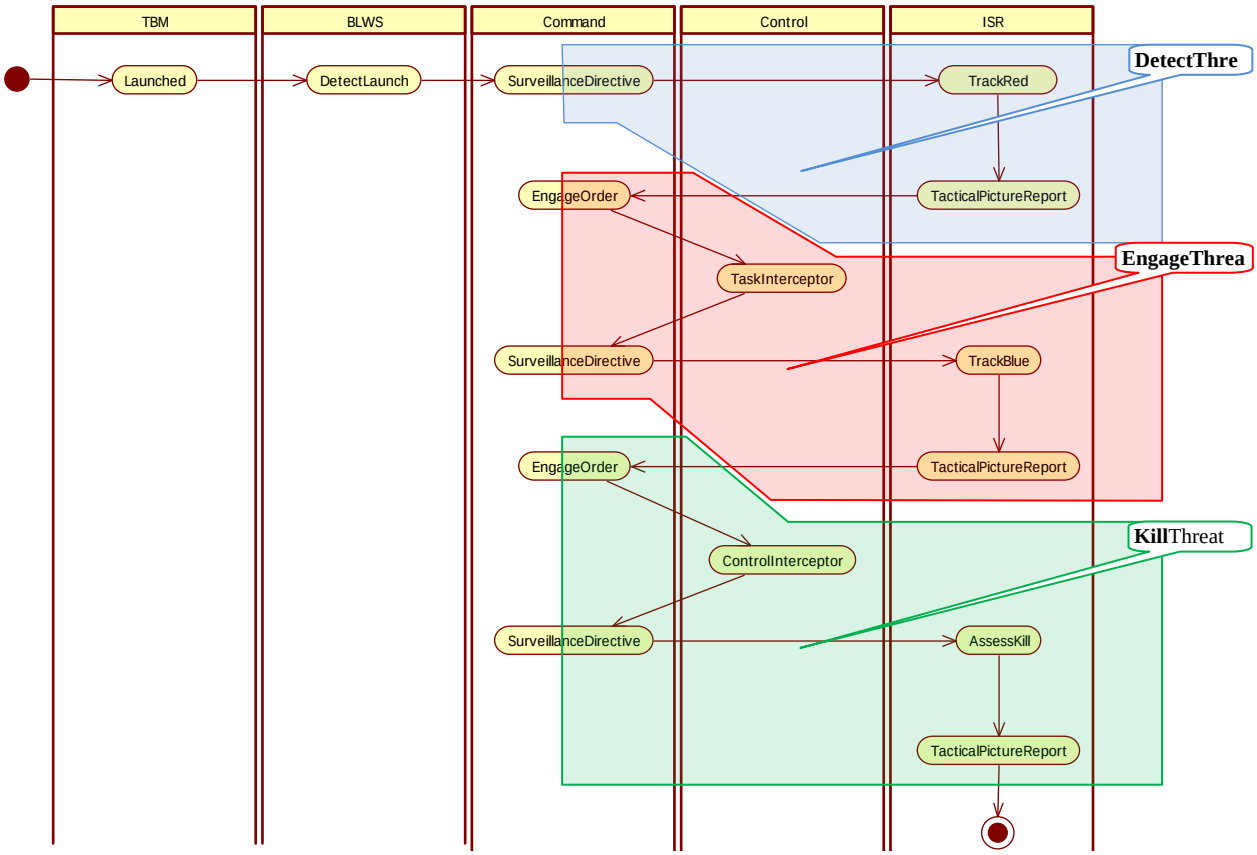


**Figure 7: ATIS Operational Activity Model (OV-5)**

Figure 7 shows the ATIS Operational Activity Model (OV-5). We identified three operational capabilities:

1. Initial threat detection (*DetectThreat*): involves triggering the ATIS to start tracking an incoming threat and propagating its new tactical picture. Initial threat detection is triggered by the external GBMW service.

2. Threat interception (*EngageThreat*): involves steps necessary for the allocation of an interceptor and engaging the threat. Threat interception is triggered by the initial threat detection.

3. Threat destruction (*KillThreat*): involves steps necessary to destroy the threat and its kill assessment. Threat destruction is triggered by the threat interception capability.

Accordingly we defined four business processes to expose these operational capabilities. Figure 8 shows ATIS business process triggered by the Launch Warning Node. The ATIS business process is a composition of three sub-business processes, *DetectThreat*, *EngageThreat* and *KillThreat* each of which is exposed by a composite service. Although the business processes involve activities in all operational nodes, all four business processes are owned (triggered) by the Command node and are shown to reside in the Command Node in Fig. 8.

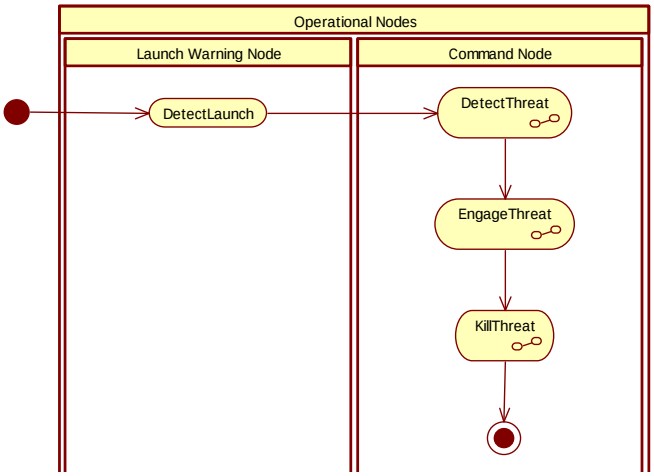


**Figure 8: ATIS Business Process**

The initial threat detection and tracking is done by the *DetectThreat* business process shown in Fig. 9 which involves the generation of an incoming threat surveillance directive, tracking the incoming threat, and generating a new tactical picture.

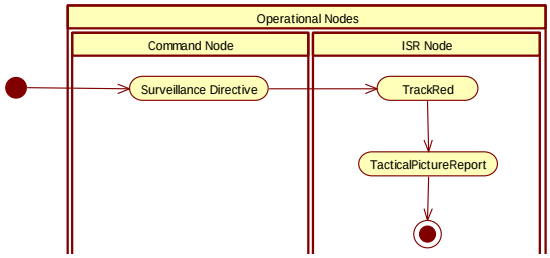


**Figure 9: DetectThreat**

The new tactical picture is passed to the *EngageThreat* business process depicted in Fig. 10. The *EngageThreat* business process is responsible for allocating an interceptor to destroy the TBM and tracking it (interceptor). The new tactical picture sent from the *DetectThreat* business process is passed to the *EngageOrder* activity which sends an intercept order to the *TaskInterceptor* activity in the Control Node.

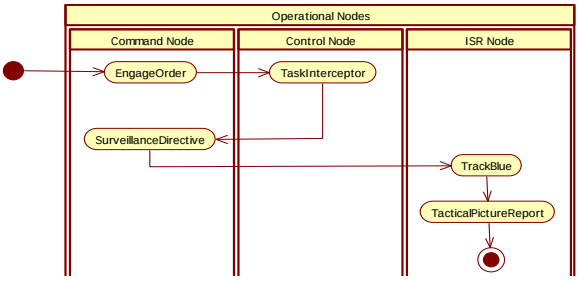


**Figure 10: EngageThreat**

The *TaskInterceptor* activity tasks a free interceptor and requests a track interceptor surveillance directive. The track interceptor surveillance directive is sent to the ISR node *TrackBlue* activity. As soon as the *TrackBlue* activity detects that the interceptor is in firing range, an in firing range tactical picture is generated and sent to the *KillThreat* business process. The *KillThreat* business process shown in Fig. 11 is responsible for destroying the TBM and assessing its kill status. First a fire order is generated by the

*EngageOrder* activity, which is sent to the Control node *ControlInterceptor* activity to task the interceptor, a kill assessment is generated by the *AssessKill* activity, and finally a tactical picture reflecting the kill status of the engaged threat is generated.

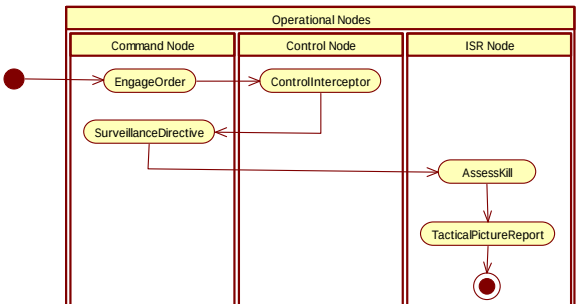


**Figure 11: OV-5 KillThreat**

The Systems and Services Interface Description (SV-1) is shown in Fig. 12. The ATIS business services are shown in SV-1 and detailed in Tab. 1. SV-1 also shows the ESB services located at the Command Center System Node. For simplicity, it was assumed that the *TPReport* Service is responsible for maintaining a tactical picture database and posting the changing Tactical Picture to the NCES. Systems and Services Communication description (SV-2) and Operational Activity to Services Traceability Matrix (SV-5c) are shown in Appendix A, Fig. A 4 and Tab. A 1 respectively.

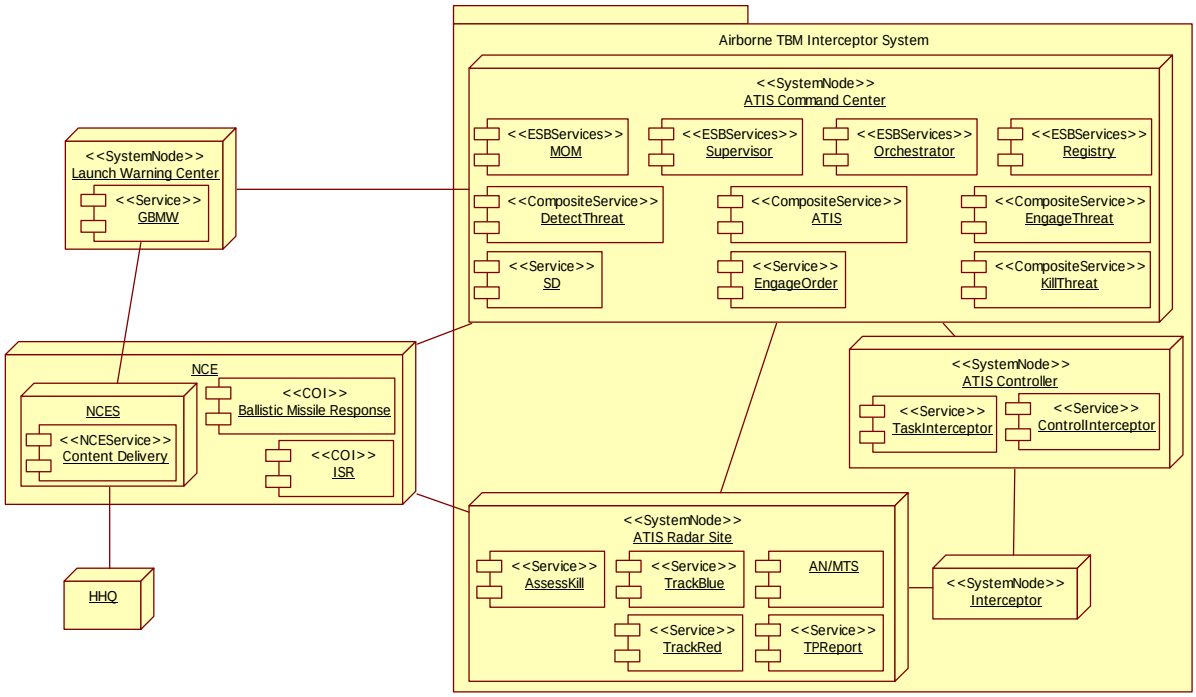


**Figure 12: Systems & Services Interface Description SV-1**

**Table 1: ATIS Services**

| Services | System Node | Composite | Business Process Name | NCE Availability | COI | Anticipated User | Un-anticipated User |
|---|---|---|---|---|---|---|---|
| ATIS | ATIS Command | Yes | ATIS | Yes | Ballistic Missile Response | Launch Warning Center (Global Ballistic Missile Warning Service) | Members of Ballistic Missile Response COI |
| DetectThreat | | Yes | DetectThreat | No | N/A | N/A | N/A |
| EngageThreat | | Yes | EngageThreat | No | N/A | N/A | N/A |
| KillThreat | | Yes | KillThreat | No | N/A | N/A | N/A |
| SD | | No | N/A | No | N/A | N/A | N/A |
| EngageOrder | | No | N/A | No | N/A | N/A | N/A |
| TrackRed | ATIS Radar | No | N/A | Yes | ISR | N/A | Members of ISR COI |
| TrackBlue | | No | N/A | Yes | ISR | N/A | Members of ISR COI |
| AssessKill | | No | N/A | No | N/A | N/A | N/A |
| TPReport | | No | N/A | No | N/A | N/A | N/A |
| ControlInterceptor | ATIS Control | No | N/A | No | N/A | N/A | N/A |
| TaskInterceptor | | No | N/A | No | N/A | N/A | N/A |

Table 1 shows all business services hosted by the ATIS and their description. Figure 13 shows Services Event Trace Description (SV-10c) for the *ATIS* business process. For simplicity the execution of *DetectThreat*, *EngageThreat* and *KillThreat* business processes and the interaction between the MOM and the rest of the services was hidden. SV-10c describing the *DetectThreat* business process is shown in Appendix A, Fig. A 5.

All business services involved in the ATIS are registered in the ESB Registry and Orchestrator Services. In Fig. 13, the GBMW Service detects a TBM, it sends *RequestService* message to the ATIS composite Service. The ATIS composite service requests an ATIS business process instance by sending a *RequestBusinessProcess* message to the ESB Orchestrator. The *RequestBusinessProcess* message contains the name of the business process to be executed and its parameters. The ESB Orchestrator looks up the requested business process in its repository. The business process definition stored in the Orchestrator's repository contains the names of the services participating in the business process rather than specific service nodes, thus after reading the business process definition, the Orchestrator constructs a *RequestServices* message and sends it to the Registry requesting service nodes that can participate in the business process instance. The Registry looks up the service nodes to satisfy the request in its registry and responds back to the Orchestrator with a list of service nodes that can participate in the business process instance. After receiving the service nodes, the Orchestrator requests message flow between itself and the service nodes from the MOM by sending a *RequestMessageFlows* message to the MOM. The MOM configures the requested message flows and responds back to the Orchestrator with a *MessagesFlowsResponse* message containing a list of message flow handlers. The last step in the instantiation of a business process instance is to request that the SOA Supervisor monitor the business process instance by sending a *MonitorBusinessProcess* message to the SOA Supervisor containing the newly created business process instance handler. As soon as the Orchestrator sends the *MonitorBusinessProcess* message it starts executing the business process. The execution of an ATIS business process involves the instantiation of three business processes instances not detailed in the figure.

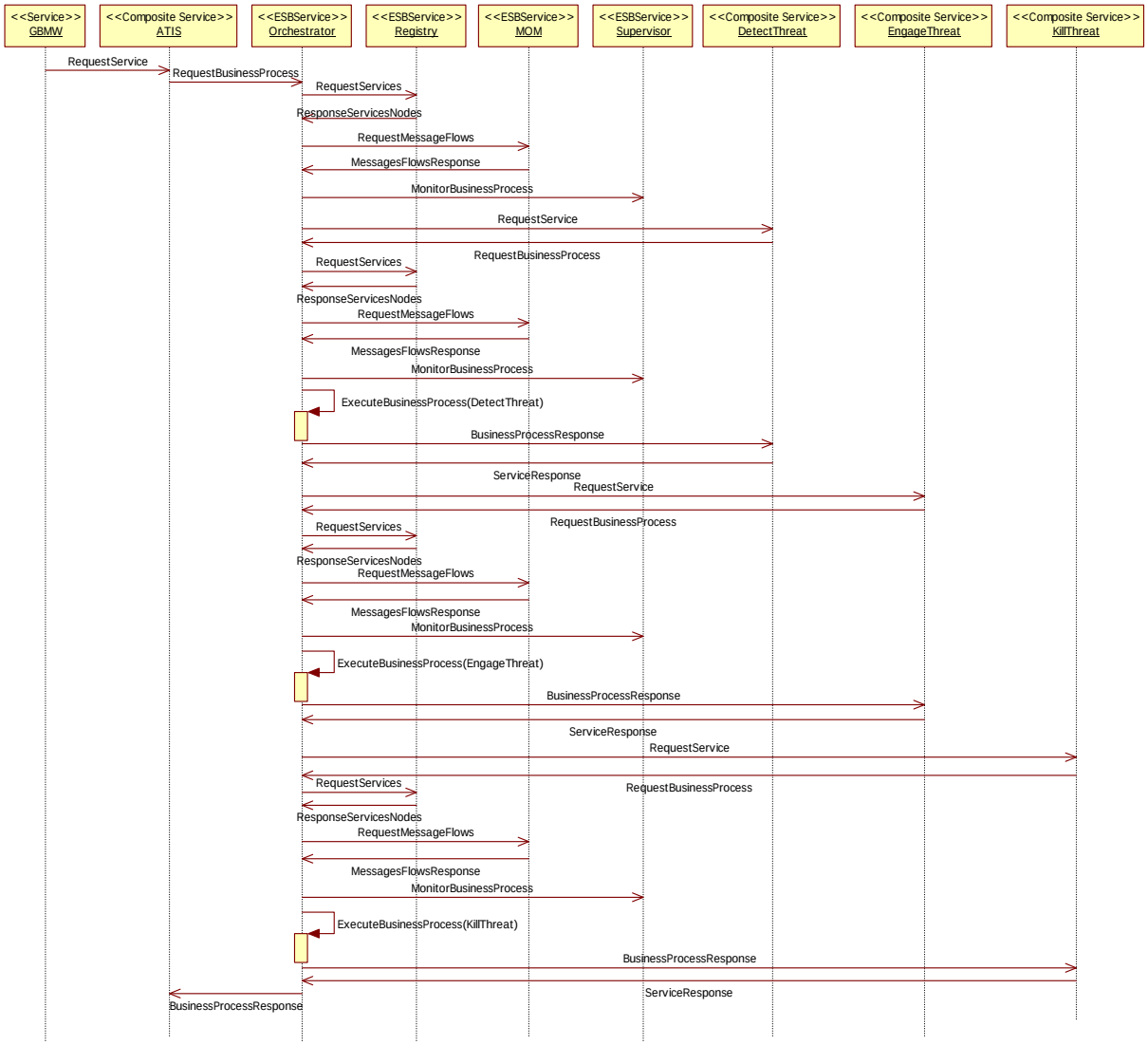


**Figure 13: SV-10c ATIS Business Process**

A CPN executable model using CPNTools was created to evaluate the Operational View of the architecture. This model included the business processes and business services of the ATIS, but no ESB interaction is included. The top page of the CPN model is shown in Fig. 14. Additional pages of the model are shown in Appendix B.

Once created the executable model was used to check the logic and behavior of the architecture business services and their composition into business processes. As errors were detected, fixes were made to the CPN model and reflected back to the architecture artifacts. After the logic and behavior of the architecture was verified, the performance of the ATIS capabilities was tested by converting the CPN model into a timed CPN.

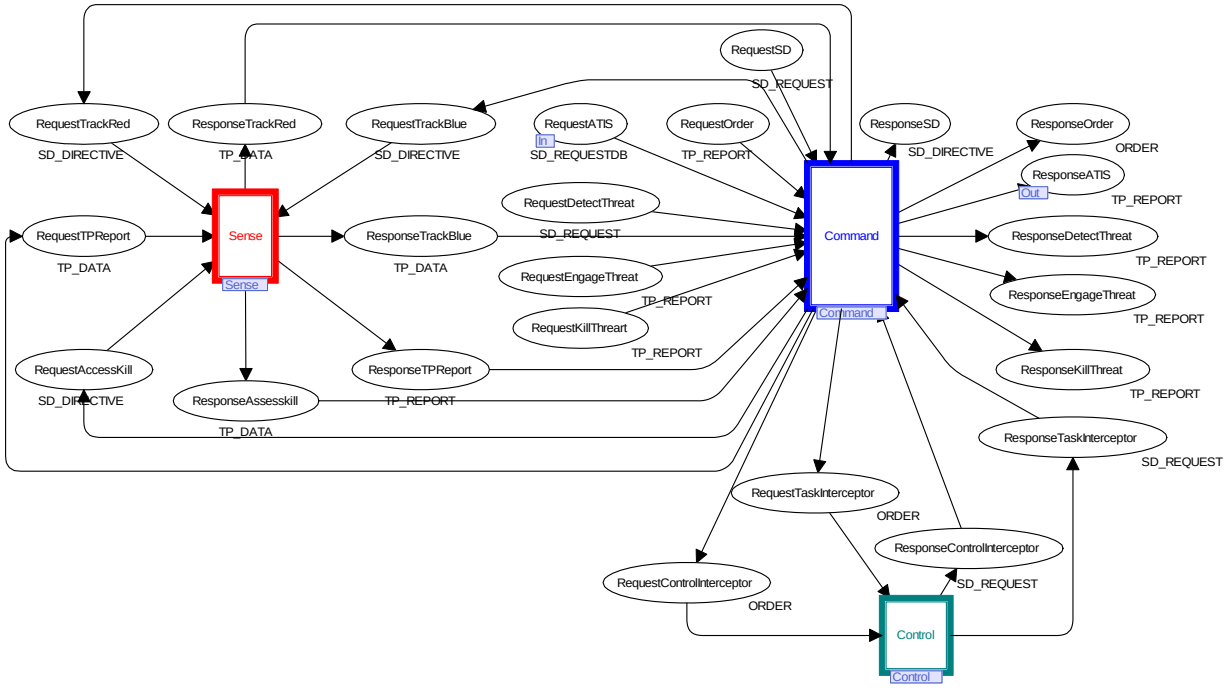


**Figure 14: AITS Page**

The following scenario was used for performance analysis: the ATIS and the GBMW service are deployed, and an adversary capable of launching multiple TBM exists. The ATIS must be able to launch its interceptors at the TBM within 400 seconds of initially detecting it with the ATIS Radar site. If it takes more than 400 seconds to launch an interceptor, the TBM is considered a leaker. The main questions to be addressed are:

1. Is the ATIS capable of shooting down TBMs when launched?

2. How many interceptors are required to handle various adversary capabilities with the response time requirement of 400 seconds?

The input and output parameters of the simulation and their description is listed in Tab. 2.

**Table 2: Scenario Input and Output variables**

| **Input Variables** | | | |
|---|---|---|---|
| **Name** | **Values** | **Description** | |
| Number of TBMs | 10 | Total number of TBMs launched by adversary (fixed). | |
| TBM Inter-arrival | 0, 25,50,75,100 (seconds) | Time interval between TBM arrivals (continuous variable). | |
| Number of Interceptors | 3, 4, 5 | Total number of ATIS Interceptors (discrete variable). | |
| **Output Variables (Measures Of Performance – MOPs)** | | | |
| **Name** | **Description** | | **Requirement** |
| Average Response Time | ATIS average response time in seconds defined as the average time between the ATIS detecting the TBM until the TBM is destroyed. | | <= 400 seconds |
| Throughput Rate | Number of killed TBMs per second | | |
| Number of Kills | Total number of destroyed TBMs within 400 seconds of being detected by ATIS | | |
| Number of Leakers | Total number of destroyed TBMs after 400 seconds of being detected by ATIS. | | <=2 |

Processing delays were estimated for each service and applied as delays for each transition representing an operational activity. Estimates were based on the notion that the systems, system functions, and services supporting the operational activities would be able to accomplish each task in a given amount of time. These processing delays are shown in Tab. 3.

**Table 3: Processing Delays**

| Services | Processing Delay Description | Processing Delay (seconds) |
|---|---|---|
| ATIS | Business Process Request Delay | 2 |
| | Business Process Response Delay | 2 |
| DetectThreat | Business Process Request Delay | 2 |
| | Business Process Response Delay | 2 |
| EngageThreat | Business Process Request Delay | 2 |
| | Business Process Response Delay | 2 |
| KillThreat | Business Process Request Delay | 2 |
| | Business Process Response Delay | 2 |
| SD | Surveillance Directive Generation Delay | 4 |
| EngageOrder | Order Generation Delay | 10 |
| TrackRed | Processing Delay | 5 |
| TrackBlue | Processing Delay | 5 |
| AssessKill | Processing Delay | 5 |
| TPReport | Processing Delay | 5 |
| ControlInterceptor | Delay for an interceptor to shoot a TBM | 5 |
| TaskInterceptor | Delay for a free interceptor to fly to a given TBM location | 120 |
| | Delay for an Interceptor to find and lock on a TBM | 20 |

The summary of the results of the simulations are presented in Tab. 4, the CPN model used for the simulations is shown in Appendix B.

**Table 4: Simulation Results**

| Number of Interceptors | TBM Inter-arrival | Average Response Time | Throughput Rate | Number of Leakers |
|---|---|---|---|---|
| 3 | 0 | 347.1 | 0.0178571 | 4 |
| | 25 | 270.1 | 0.018 | 1 |
| | 50 | 180.6 | 0.018 | 0 |
| | 75 | 159 | 0.0133333 | 0 |
| | 100 | 159 | 0.0100446 | 0 |
| 4 | 0 | 283.9 | 0.0261628 | 2 |
| | 25 | 212.9 | 0.0252101 | 0 |
| | 50 | 159 | 0.0201794 | 0 |
| | 75 | 159 | 0.0133333 | 0 |
| | 100 | 159 | 0.0100446 | 0 |
| 5 | 0 | 245.5 | 0.0436893 | 0 |
| | 25 | 180 | 0.0340909 | 0 |
| | 50 | 159 | 0.0201794 | 0 |
| | 75 | 159 | 0.0133333 | 0 |
| | 100 | 159 | 0.0100446 | 0 |

Table 4 shows the results of the 15 simulation runs. The values of three key Measures of Performance (MOPs) were calculated from the data in the simulation runs: average response time, throughput rate and number of leakers. Requirements were established for these MOPs (no more than two leakers and maximum allowed average response time of 400 seconds).

The results of the analysis are summarized as follows:

1. 3 interceptors can handle the 10 threats (with a max of four leakers) if they arrive at a rate slower than 1 in 25 seconds

2. 4 interceptors can handle the 10 threats (with a max of two leakers) if they arrive at a rate slower than 1 in 25 seconds

3. 5 interceptors can handle the 10 threat with no leakers.

The results presented in Tab. 4 are based on an executable model of the Operational View, with estimates of the operational activities processing times shown in Tab. 3. We assumed that communications delays would be negligible compared to the processing and human decision making delays and therefore zero time delay for the communications network was assumed. Indeed the communications network was not modeled explicitly. Furthermore, processing delays and overhead of the SOA infrastructure services was not captured in this executable model. Constructing an executable model based on the Systems and Services View should give a better understanding of the system's performance by capturing the communications systems involved and the SOA infrastructure and how they interact to enable the composition of business services in business processes (capabilities) to successfully execute the mission/s of the architecture. The results of the OV-based executable model can be used as an upper bound for the performance of the SV-based executable model.

## 5. Conclusion and Future Work

A key to Command and Control (C2) agility is effective information sharing that allows for better decision making and effective use of resources. SOA provides alignment between business objectives and the supporting IT infrastructure resulting in an agile information system in which information sharing becomes the norm and data and information become ubiquitous. The importance of SOA and its role in achieving agile C2 systems has been recognized by the DoD, and was reflected in DoDAF v1.5 which uses SOA as the key enabler of NCOs. We have presented an approach for constructing an event driven SOA compliant to DoDAF v1.5. The architecture defines business services and processes necessary to accomplish its operational concept, and is capable of participating in the NCE. The participation in NCE was achieved by allowing the SOA instance to dynamically federate with NCE systems through COI registries and by utilizing the NCES to share enterprise-level information. The architecture uses an ESB-based SOA infrastructure to govern, manage, and monitor its services and processes. The end product of the approach is an executable model representing the Operational View of the architecture. The executable model is derived from the information contained in the DoDAF v1.5 artifacts and is used to evaluate the logic, behavior, and performance characteristics of the architecture's business processes services.

Since SOA behavior and performance do not only depend on its business services, but also on the infrastructure that enables loose coupling, services implemented by other systems, and the underlying technological network supporting the SOA environment, there is a need for a more elaborate executable model that captures the Systems and Services View of the architecture and the protocols involved in executing the business processes. Traditional executable models such as the one presented in this paper are not sufficient to capture the complexity of a SOA. They can only serve as upper bounds for the actual performance of the system. If the Operational View executable model satisfies the functional and non-functional requirements of the architecture, we can proceed in constructing the Systems and Services executable model; otherwise the architecture should be revised or even dropped completely since the Operational View performance does not satisfy the requirements.

The executable model of the Systems and Services View that captures SOA is expected to be complex. Layered approaches that connect CPN models with communication network specific simulators such as the approach presented in [21] has proven to enhance the evaluation process. Additional work is required to: (1) extend such approach to capture SOA complexity, (2) define a process for constructing such an executable model from the architecture's artifacts, (3) define a process for using the executable model for analysis and evaluation.

## References


[1] "DoD Architecture Framework (DoDAF), V1.5, Vols I, II, III," April 23, 2007.

[2] Alexander H Levis and Lee W Wagenhals, "C4ISR Architectures: I. Developing a Process for C4ISR Architecture Desgin," *Systems Engineering*, vol. 3, pp. 225-246, 2000.

[3] Lee W. Wagenhals, Insub Shin, Daesik Kim, and H. Alexander Levis, "C4ISR Architectures: II. A Structured Analysis Approach for Architecture Design," vol. 3, no. 4, p. 248–287, 2000.

[4] Lee W Wagenhals, Sajjad Haider, and Alexander H Levis, "Synthesizing Executable Models of Object Oriented Architectures," *Systems Engineering*, vol. 6, p. 266–300, 2003.

[5] W Lee Wagenhals and Alexander H Levis, "Service Oriented Architectures, The DoD Architecture Framework v. 1.5, and Executable Architectures," *Systems Engineering*, vol. 12, 2009.

[6] DoD CIO, "Department of Defence Net-Centric Data Strategy," DoD, May 9, 2003.

[7] Alexander H Levis, "Systems Architectures," in *Systems Engineering and Management Handbook*, Andrew P Sage and William P Rouse, Eds. New York: Wiley, 1999, pp. 427-453.

[8] Mark W Maier and Eberhardt Rechtin, *The Art of Systems Architecting*, 2nd ed. Boca Raton: CRC Press, 2000.

[9] DoD CIO, "Department of Defense Net-Centric Services Strategy," DoD, March 2007.

[10] Andrew P Sage. DoD Use of SOA:The Role of Service Oriented Architectures in Enterprise Architecting. IBM 'DoD Use of SOA' Workshop, January 9th 2008.

[11] Thomas Erl, *Service Oriented Architecture: Concepts, Technology, and Desgin.*: Prentice Hall PTR, 2005.

[12] Judith Hurwitz, Robin Bloor, Carol Baroudi, and Marcia Kaufman, *Service Oriented Architecture FOR Dummies.*: WILEY, 2007.

[13] David A. Chappell, *Enterprise Service Bus.*: O'REILLY, 2004.

[14] Lawrence Gloss, "Service Oriented Architecture Federation Framework: An Approach For Service-based Domain Integration," Fairfax, VA, 2007.

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

[16] Marc Goodner and Anthony Nadalin, "Web Services Federation Language (WS-Federation) Version 1.2," September 26, 2007.

[17] Michael P Bienvenu, Insub Shin, and Alexander H Levis, "C4ISR architectures: III. An object-oriented approach for architecture design," *Systems Engineering*, vol. 3, no. 4, pp. 288-312, 2000.

[18] CPN Tools. [Online]. "http://wiki.daimi.au.dk/cpntools/cpntools.wiki" http://wiki.daimi.au.dk/cpntools/cpntools.wiki

[19] Lars M Kristensen, Soren Christensen, and Kurt Jensen, "The practitioner's guide to colouredPetri nets," *International Journal on Software Tools for Technology Transfer*, vol. 2, pp. 98-132, 1998.

[20] Alexander H. Levis, "Measureing The Effectiveness of C4I Architectures," in 1997 International Symposium on Defense Information, Seoul, Republic of Korea, 1997.

[21] Insub Shin and Alexander Levis, "Performance prediction of networked information systems via Petri nets and queuing nets," vol. 6, no. 1, pp. 1 - 18, 2003.

# Appendix A: ATIS DoDAF products

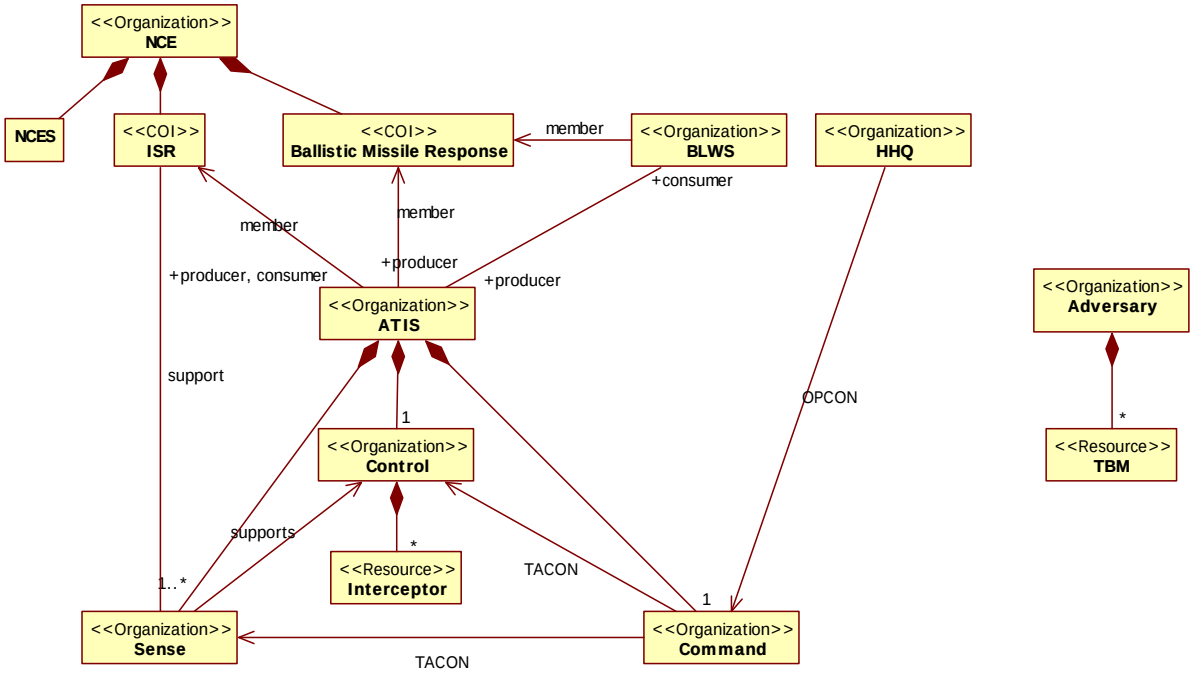


**Figure A 1: Organizational Relationships Chart (OV-4)**

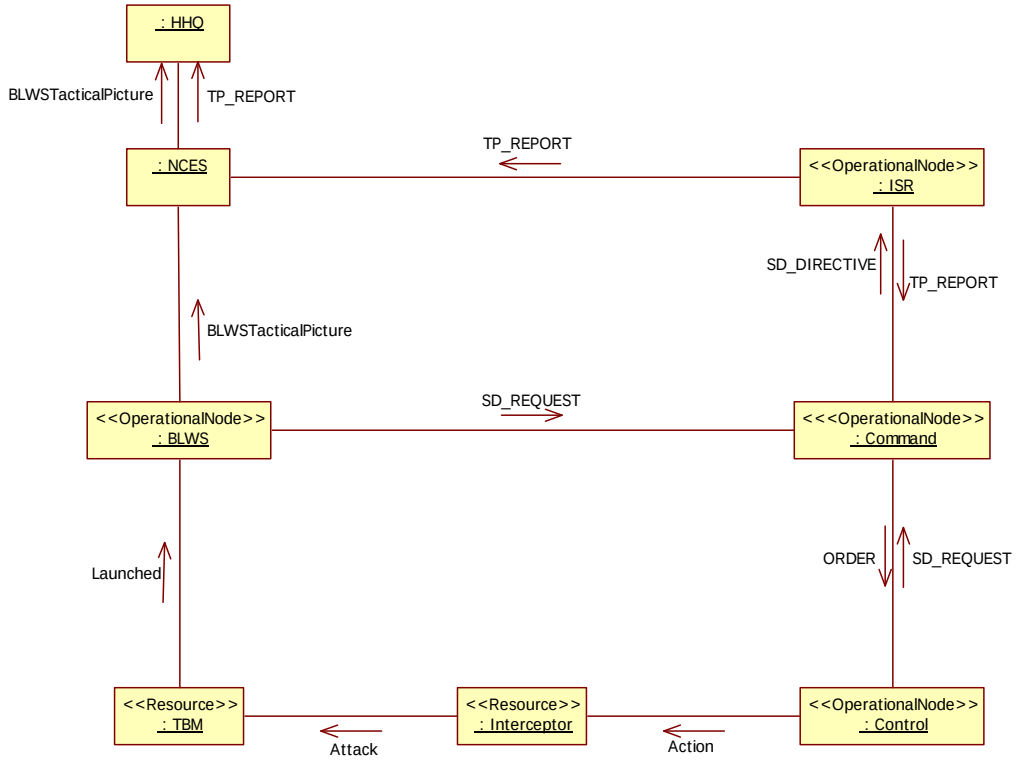


**Figure A 2: Operational Node Connectivity Description (OV-2)**

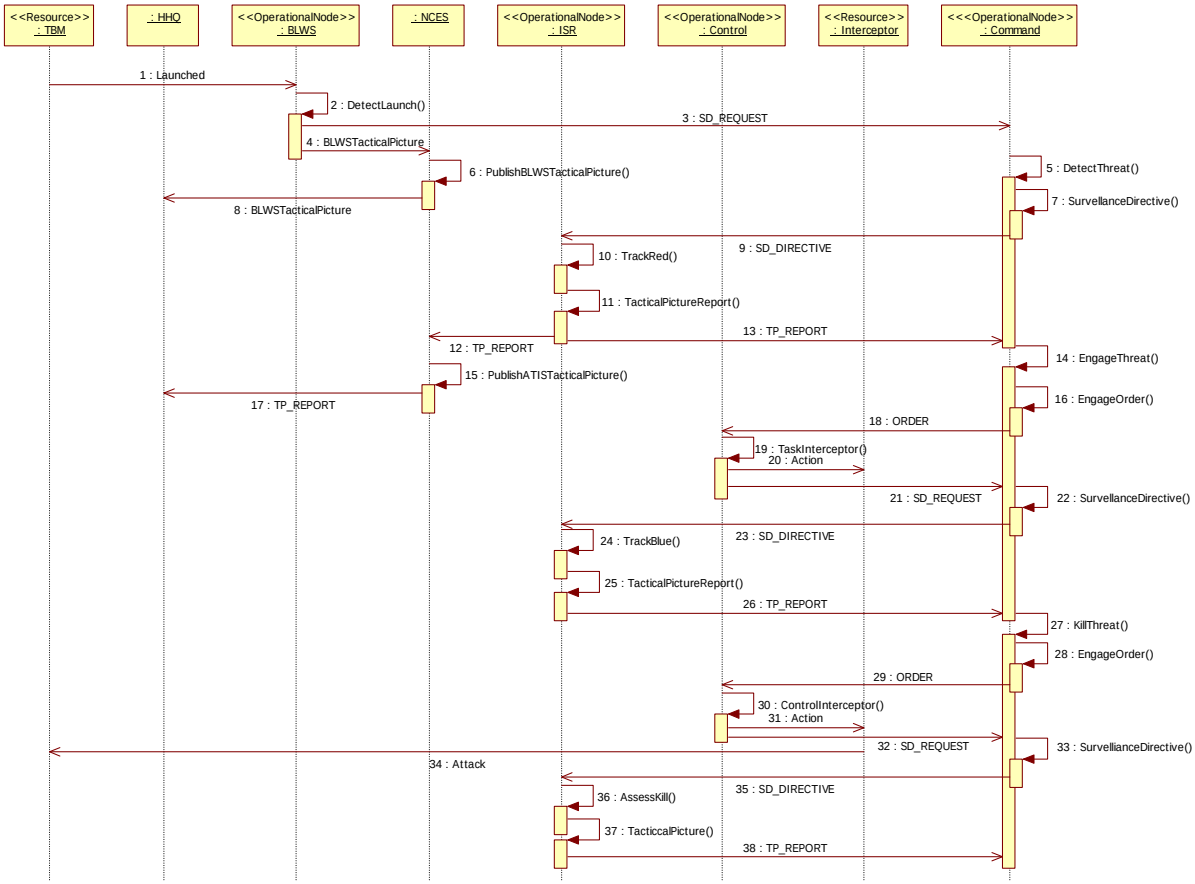


**Figure A 3: Operational Event-Trace Description (OV-6c)**

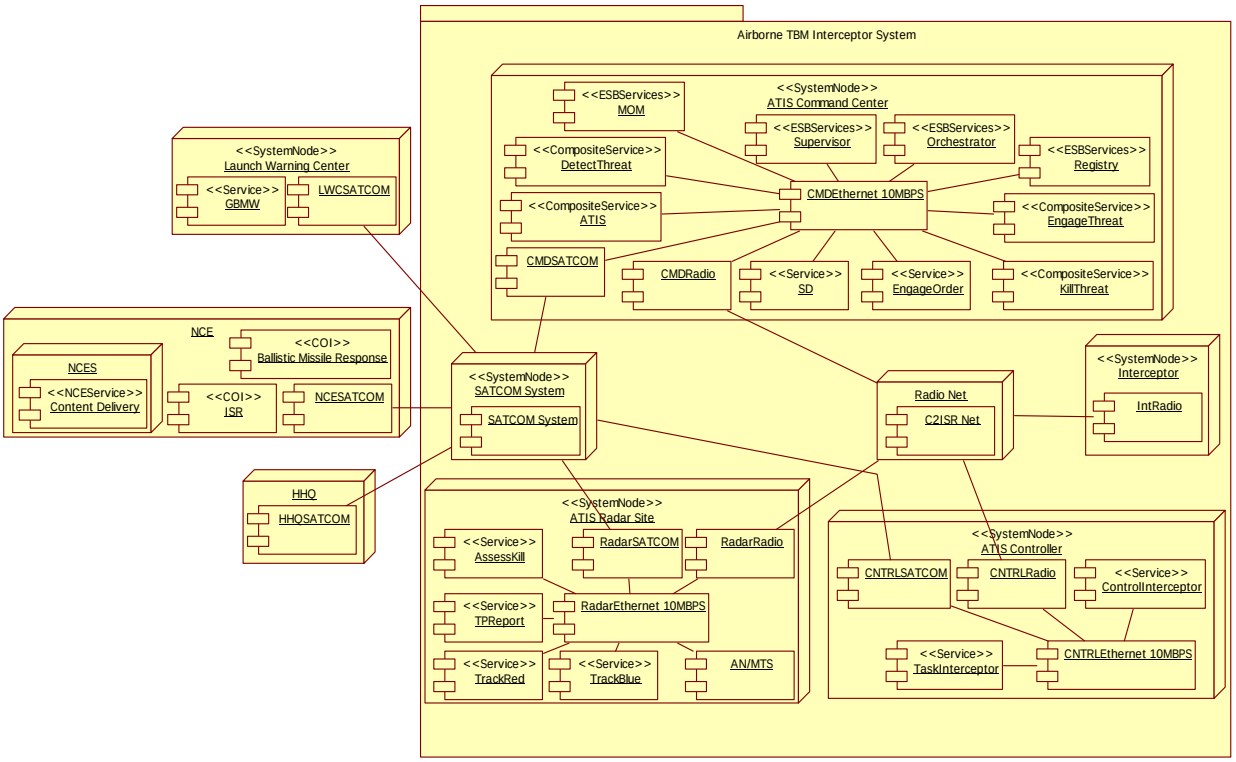


**Figure A 4: Systems and Services Communication Description (SV-2)**

**Table A 1: Operational Activity to Services Traceability Matrix (SV-5c)**

| **Services** | **Operational Activities** | | | | | | | | **Operational Capabilities** | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | Surveillance Directive | Engage Order | TrackRed | Track Blue | Assess Kill | TacticalPicture Report | Control Interceptor | Task Interceptor | Detect Threat | Engage Threat | Kill Threat |
| | | | | | | | | | | | |
| ATIS | | | | | | | | | | | |
| DetectThreat | | | | | | | | | X | | |
| EngageThreat | | | | | | | | | | X | |
| KillThreat | | | | | | | | | | | X |
| SD | X | | | | | | | | X | X | X |
| EngageOrder | | X | | | | | | | | X | X |
| TrackRed | | | X | | | | | | X | | |
| TrackBlue | | | | X | | | | | | X | |
| AssessKill | | | | | X | | | | | | X |
| TPReport | | | | | | X | | | X | X | X |
| ControlInterceptor | | | | | | | X | | | | X |
| TaskInterceptor | | | | | | | | X | | X | |

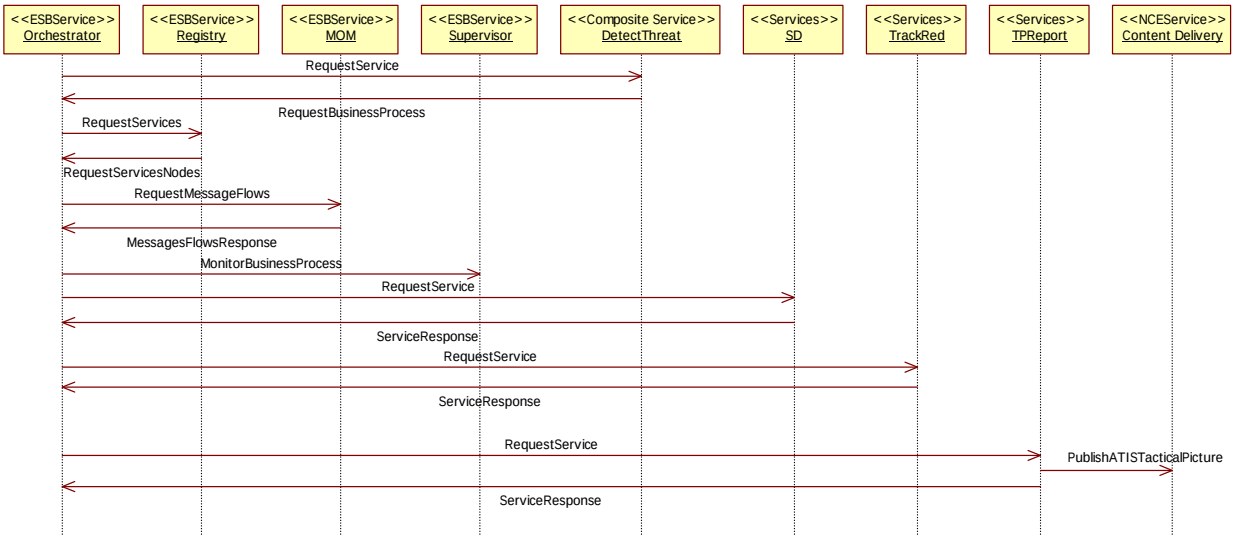


**Figure A 5: *DetectThreat* Business Process (SV-10c)**

# Appendix B: ATIS CPN Model

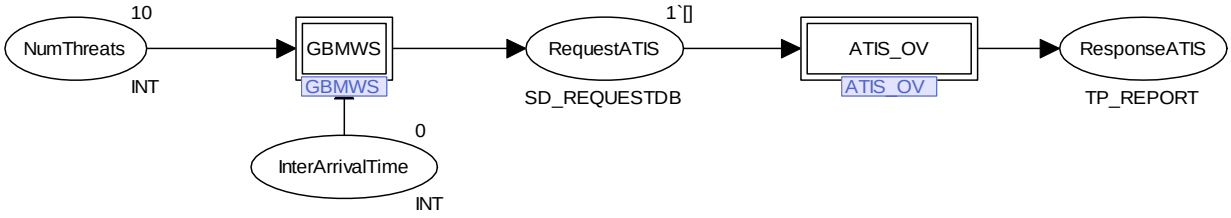


**Figure B 1: Scenario Generator**

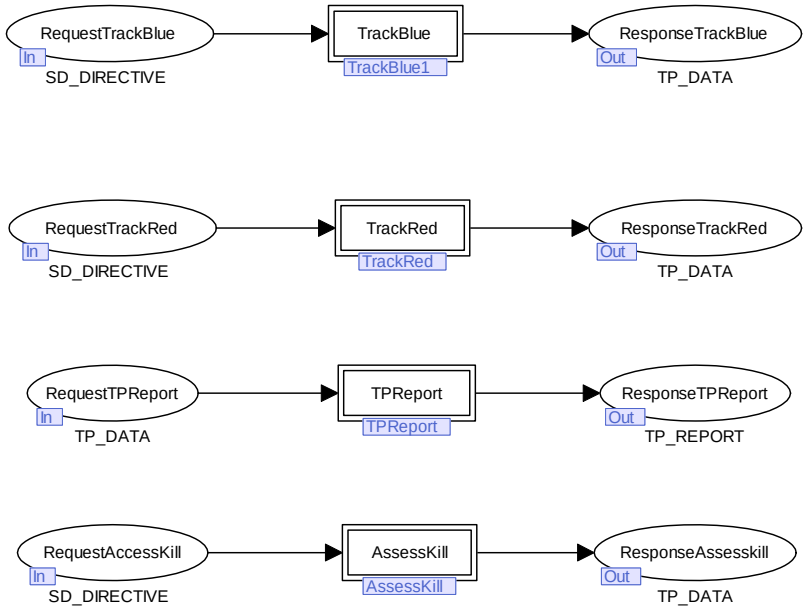


**Figure B 2: ISR Node**

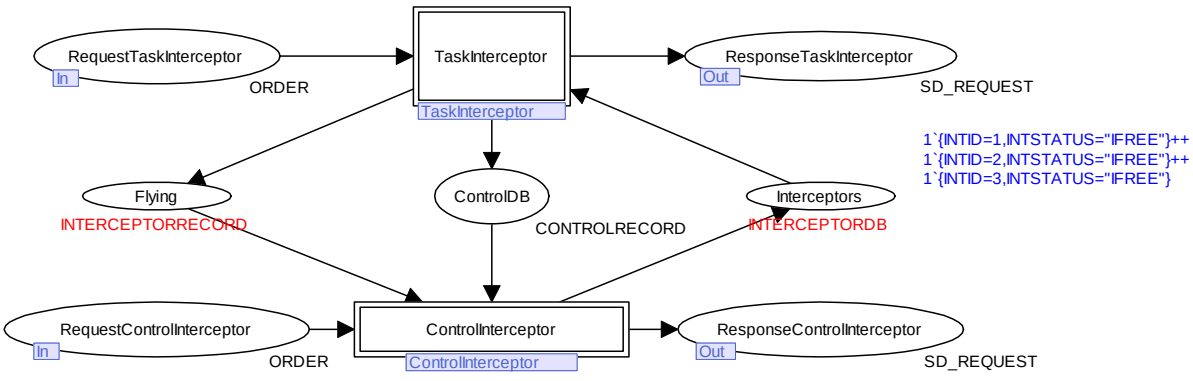


**Figure B 3: Control Node for 3 Interceptors**

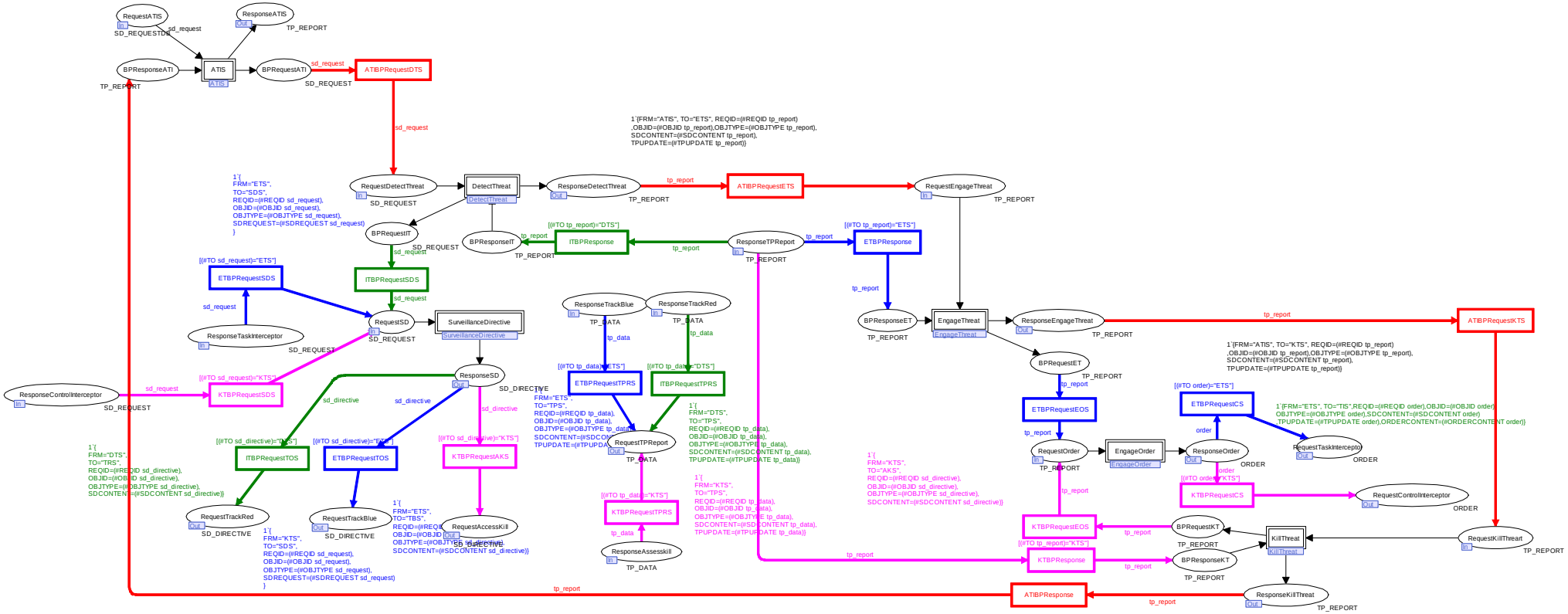


**Figure B 4: Command Node**

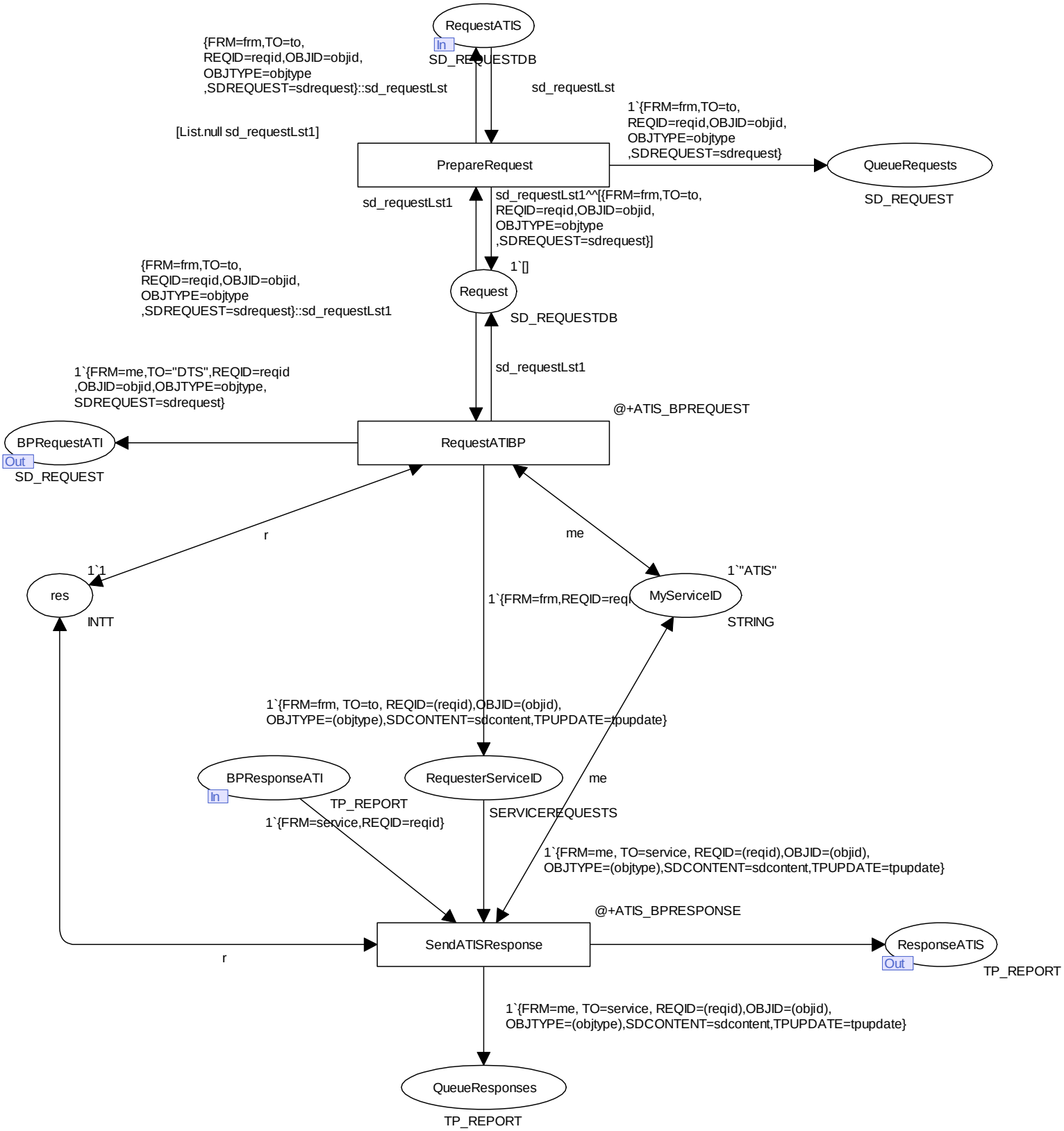


**Figure B 5: ATIS Service**