Flexible Modeling Support Environments 13

The introduction laid out the theme of this book—it is about modeling and simulation to support "virtual build and test" of Systems of Systems (SoS). Constructing a computer model and testing the design of a configuration of components before implementing it in reality is increasingly the only workable approach to creating a SoS. The focus of Parts I and II was to elucidate the concepts underlying the approach to "virtual build and test" based on DEVS methodology in the context of the MS4 Modeling Environment. As we begin Part III, we now take a broader look at DEVS methodology and support environments from a number of different perspectives.

In this chapter, we discuss a design environment that supports implementation of a novel architecture for systems of fractionated satellites—these are satellites that are composed from modular components. Although framed as applying to fractionated satellites, the considerations equally apply to many other types of SoS. The environment includes a comprehensive user input interface intended to elicit stakeholder objectives, values, and service requirements and a Modeling and Simulation Support Environment (MSE). As a core component of this environment, the MSE should be capable of providing simulations that evaluate spacecraft system architectures in response to the requirements of diverse stakeholders such as satellite designers, communications specialists, and space experimenters.

13.1 Supporting Multiple Paths Through Development Process

The goal of responding to the requirements of diverse stakeholders is contrasted with typical modeling and simulation (M&S) workflows conceptualized in Fig. 13.1. The workflow in Fig. 13.1(a) is that of a sequential waterfall process like that discussed in Chap. 3. It has phases such as conceptualization, design, implementation and testing together with the possibility of iterations that return the flow through earlier phases of the process. In contrast, the concept illustrated in Fig. 13.1(b) envisions a flexible system architecture that supports a wide variety of stakeholders who may be taking different paths through the environment. Depending on diverse interests, objectives, and values, different modeling, simulation, and analysis services as well

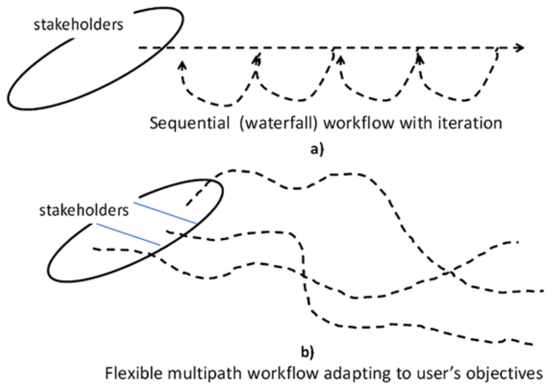

stakeholders
Sequential (waterfall) workflow with iteration
a)
stakeholders
b)
Flexible multipath workflow adapting to user's objectives

Fig. 13.1 Accommodating diverse stakeholders in a flexible MSE

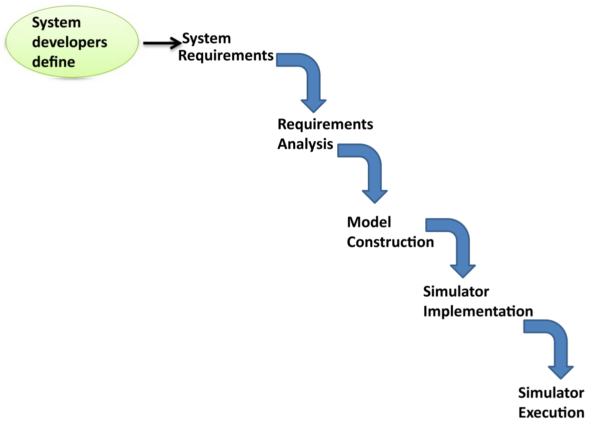


Fig. 13.2 Another formulation relating to developing training simulators

as different pipelines of services may be appropriate. This expanded approach conforms better to a theory-based methodology for developing simulation models of complex systems (Aumann 2007). The key to achieving this flexibility is to provide an appropriate classification of stakeholders that places users with similar paths in the same equivalence class.

In the following, we evolve from a structured workflow such as in Fig. 13.1(a) to flexible environment such as Fig. 13.1(b) in a series a steps that show the advantages and utility of the latter concept as well as the requirements for control of activities and flow of information that need to be met to make it work.

Figure 13.2 depicts an example workflow process that starts with a formulation of requirements for a training system that a simulator is to support, e.g., flying a newly developed jet airplane, controlling a nuclear power plant, or a team collaboration in a war game (Kim et al. 2011). Such requirements state the kinds of behaviors that the training system should display and the kinds of interactions that users (trainees) can have with it. Analysis of these requirements then results in the kinds of objects that should be included, their attributes and behaviors, and the types of measurements that can be made of trainee performance. Next, a DEVS model is constructed

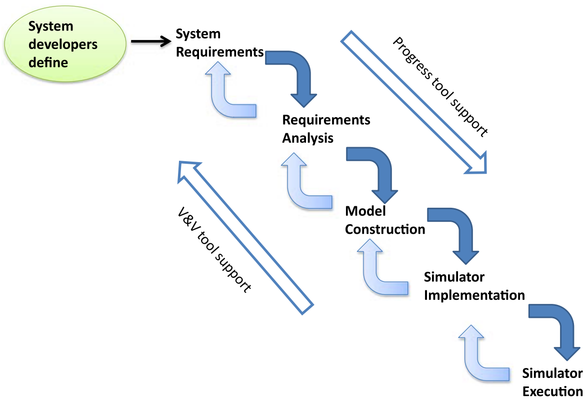


Fig. 13.3 Simulator development process with reverse verification steps

that contains the objects and attributes as well as realizing the specified behaviors and measurement indexes. The model is implemented in a DEVS simulator which typically will be a distributed platform with multiple simulation nodes. Execution of the simulator by trainees (possibly training as a group in a distributed simulation) completes the process.

Sequential step-by-step depictions such as those in Fig. 13.2 are good at showing a normal workflow or idealized progress through M&S development processes.

A workflow depiction is a good starting point but fails to provide a sufficient basis for understanding how tools and services can support such processes.

- One defect is that the idealized nature of the waterfall sequence fails to portray how activities in real world development processes proceed where departures from this pattern are more common than not.
- A second defect is the fact that the arrows signify flow of control without explicitly showing the information artifacts produced and consumed at each step.

Figure 13.3 addresses the first defect by showing one pattern in which the process can return to earlier steps in an iterative manner. We will refer to software applications or concepts that help you check your work at each step and enable you to return to this or earlier steps as *verification and validation* (V&V) tools. These tools can be contrasted to those that enable you to advance from one step to the next in the workflow, which we will call *progress* tools.

Figure 13.4 addresses the second defect by showing process steps as information processing modules with input and output data objects. In this diagram, the information flow follows along the lines of the sequence of processing steps so that whenever a step (or module) completes its work, the products are shown as outputs that are sent to the next module as inputs. This kind of information flow diagram can be enhanced to show data objects being sent forward to modules in the chain beyond the immediate next step, as well as in the reverse direction.

However, we can get some better insight by taking another perspective, which focuses on the data and model objects produced and consumed by the progress and V&V tools. Following the approach of Kim et al. (2011), for M&S simulator development, we display in Fig. 13.5, some of the tools they developed. These tools

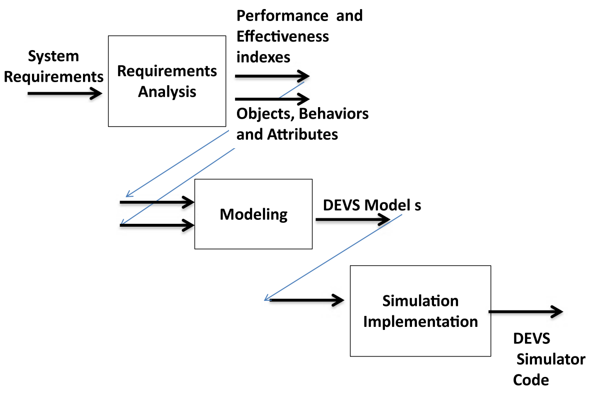

System Requirements
Requirements Analysis
Performance and Effectiveness indexes
Objects, Behaviors and Attributes
Modeling
DEVS Model s
Simulation Implementation
DEVS Simulator Code

Fig. 13.4 Simulator development process showing information flow

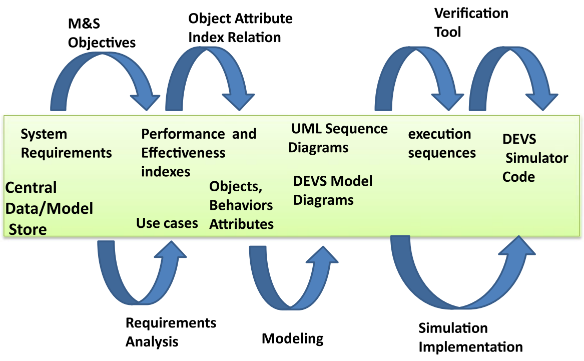


Fig. 13.5 Simulation development supported by tools interacting with a central store

include progress tools such as the concept of employing M&S Objectives to derive measures of performance and effectiveness from the given system requirements in the requirements analysis phase (step). Also included is the Object-Attribute-Index Matrix, which is a relation among objects, attributes, and performance indexes. This can be viewed as a relational database in which you can enter the performance or effectiveness indexes that were identified as needed during requirements analysis and get suggested objects and attributes to support work in the following model development phase. Requirements analysis also identifies behaviors of objects needed to realize the M&S Objectives. These behaviors are then expressed in UML sequence diagrams that capture the interaction of objects over time in various use cases that are also products of the requirements analysis. The V&V tools can do things like generate correct execution sequences that the implemented simulator must display. Any departures from such sequences signal that you should regress to the model development phase to redefine some of the DEVS models that may be the source of the errors. Not shown are also a host of progress and V&V tools that relate to implementation of the simulator in a distributed simulation environment. See Kim et al. (2010) for details.

You can see that the more tools that are available *and* the more that they are effective and easy to use, the faster progress will be in achieving your objectives and requirements. A good part of the effectiveness of tools depends on their getting the right data produced by other tools. The workflow approach in Fig. 13.4 offers one way to do this by having tools send their outputs directly to tools that need their products as inputs. However, it is a rigid approach that can overly constrain development work as we shall see later. Figure 13.5 suggests a more flexible approach in which tools can deposit their output products in a persistent form and draw upon the products of others as needed from the common data store. This approach is not as straightforward to implement and brings up a number of interesting issues that we will address.

13.2 M&S Tools as Services in a Service Oriented Architecture

So far we have talked about formulating the activities of M&S in terms of processes and carrying out the phases of these processes with the help of tools. A sequential process such as in Fig. 13.1 provides a "baseline" to formulate the succession of activities that must be undertaken for a successful result. Real M&S activities depart from this baseline but it still serves as way to support the M&S process with what we have called progress and V&V tools (Fig. 13.3). In such an extended process, the information exchange in which tools produce and consume data objects can be supported by peer-to-peer message flow (Fig. 13.4) or by reference to a common data store (Fig. 13.5). With this as background, we can make the leap to a service oriented architecture (SOA) environment as a framework for supporting M&S activities. Tools are encapsulated into web services that are hosted on web servers. Some instances of such services are shown in Fig. 13.6. The operations of web services need the right interface descriptions to properly share data, whether directly or as mediated by the common model and data store. Moreover, service operations must be orchestrated, i.e., invoked in the proper order to execute the baseline M&S process or extended versions of it. Such interfacing and orchestration will be discussed later.

A SOA environment provides a number of important benefits:

- flexibility to support a diversity of users taking a multitude of different variations from the baseline process
- discoverability of the data and models developed and stored in the common data repository
- reusability of discovered data and models in new compositions
- learning over time based on mining the data in the common repository.

13.3 Case Study: Fractionated Satellite Systems

The Frontier design environment is a research product of the DARPA System F6 Program (Future, Fast, Flexible, Fractionated, Free-Flying Spacecraft United by Information Exchange). This program envisions the evolution of spacecraft architectures from the point of view of the future with its potential for radically new satellite design and launch technologies that can support large numbers of smaller, modular,

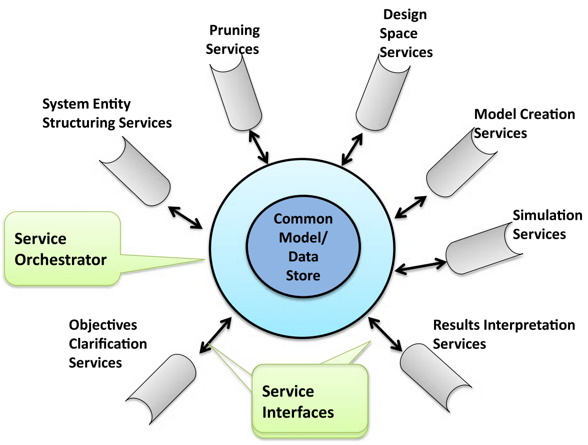


Fig. 13.6 Tools as services to support M&S in a SOA

satellites organized by networks of communication and control. A central feature in the design of the Frontier MSE, is to address the focus of the F6 program, which is to assess whether there can be fractionated architectures that can outperform current monolithic satellite systems. The following text is based on the article published by Zeigler et al. (2012). Phrased more broadly this question asks whether a proposed system of system can outperform the current system in operation.

We acknowledge that the term "outperform" will depend on the interests and objectives of stakeholders—there may not be a one-size-fits-all solution acceptable to every one. Accordingly, we created a characterization of stakeholder interests that encapsulates the diversity of such interests within a space spanned by Strategic/Tactical and Supply/Demand dimensions. The Strategic/Tactical axis refers to the horizon (long versus short) and level (high versus low) of planning. The Supply/Demand axis refers to the characterization of stakeholders' requirements for services provided by the network of satellites (Demand) versus spacecraft assets and resources available to provide those services (Supply).

We briefly present this characterization and discuss its suitability to the problem. The interests of stakeholders for evaluations of potential fractionated spacecraft architectures available in the future are characterized in the Strategic/Tactical axis as either:

- *Strategic*. This reflects an interest in evaluations of how well architectures do over extended time spans (e.g., 20 years), assessing system architectures for their *ability to adapt, evolve, and survive* in the face of changing patterns of demand for services supplied by such architectures. This particularly emphasizes long term market-oriented financial comparison of monolithic and fractionated cluster architectures.

or

- *Tactical* which typically evaluates proposed architectures over relatively short time spans (e.g., 1 year) with focus on system behavior and physical constraints (e.g., imposed by orbital mechanics) to address engineering and technological issues at the system level.

Refining this bipolar axis, we further breakdown stakeholder's interests into the four main categories:

- *Strategic/Supply* focuses on system developers, enabling them to explore the long term financial performance of their proposed architectures or technologies.
- *Strategic/Demand* focuses on system user communities, enabling them to explore the long term financial viability of their proposed profiles of demands e.g., the range of experiments of interest to a NASA space exploration community.
- *Tactical/Supply* focuses on system developers, enabling them to explore the technical performance of their proposed architectures or technologies (e.g., the effect of spacing of satellites within a cluster on its ability to meet demands for services). This category includes engineers and scientists working on the technical aspects of the F6 program.
- *Tactical/Demand* focuses on individual system users, enabling them to explore the technical feasibility of their proposed profiles of demands (e.g., a set of experiments of interest to a particular mission developer).

Stakeholders are not limited to a single characterization; the same user may adopt different characterizations to gain insights available from different perspectives—e.g., switching between Strategic and Tactical to understand both economic prospects and technical feasibility of a potential solution.

The MSE is built upon a configurable framework that adapts to each of the four stakeholder types spanned by the above dimensions. Given such a stakeholder type, the MSE configures a simulation that outputs a set of architectures that are evaluated and ranked according to their ability to meet the user's requirements. The simulation is based on various levels of abstraction that are designed to support the stakeholder's questions and objectives. This allows a given set of architectures to be evaluated, on traditional attributes (cost, weight, etc.) as well as engineering: "iities" (adaptability, reliability, etc.).

13.3.1 How the MSE Adapts to Types of Stakeholders

The MSE is built on a set of harmonized components implemented as web services that can be orchestrated to fit the stakeholder's requirements. The services are briefly enumerated and outlined:

- *Pre-Simulation Service (PSS):* interprets user inputs in the form of documents that specify demand, and evaluation metrics (for experimental frames) and architectures (for models) needed by the downstream simulation.
- *Development and Pruning of Alternatives Service (DPAS):* is the core component that applies input pruning scripts to master characterization architectures to generate a subset of potential satellite cluster architectures to be explored.
- *Simulation Service:* includes both the Strategic and Tactical Simulation Services and simulates the architectures in response to the generated demands and supplies results for evaluation and ranking. The simulation is configured to either strategic or tactical forms depending on stakeholder interest. The Simulator

Service module takes on two fundamental simulator configurations. The simulator configurations are based on models that embody abstractions of the investigated system architecture and its environment. These abstractions are tuned to the stakeholder types: Strategic (Supply or Demand) and Tactical (Supply or Demand). The simulators also tune the breadth and depth of the solution space and fidelity of the simulations (low and high) so support the user type.

  - *The Market Model Simulator (Strategic Level)* encapsulates an Experimental Frame (see Chap. 18) (generating demand inputs and collecting cluster outputs) that interacts with a Market Model to perform financial evaluation over a long time span composed of successive Simulation Analysis Intervals (e.g., quarterly).
  - *The Physical Model Simulator (Tactical Level)* encapsulates an Experimental Frame that interacts with a simulation model of the proposed architecture over a specified interval to evaluate system performance at the physical behavior level.

- *Results Analysis Service (RAS):* transforms data generated by the simulation runs into information that can be presented to the user. The input to the RAS comes from the MSE and the PSS. The input from the MSE is the data from individual simulation runs for all value metrics aggregated by Simulation Analysis Interval (SAI). Among other items, the RAS generates an overall performance score for each cluster configuration instance from a weighted average of all value metrics and an overall operational risk score for each cluster configuration. These outputs provide the basis for ranking cluster configurations instances by performance score.
- *Evaluation Service (ES):* provides the user with an interactive tool to perform decision analysis at increasing levels of detail and sophistication using the data compiled by the RAS. The functions performed by the ES include aggregating information from the RAS and generating evaluation scores to support advanced decision making features, providing information to the user in form of decision analysis tools, to support advanced interaction and analysis by the user, and collecting refined user criteria from the GUI to reassess the ranking order and current results, as well as to support sensitivity analysis of those results.

The MSE, as composed of harmonized web services, is intended to be to meet a multitude of stakeholder requirements by configuring itself to enable each user to pursue multiple paths through the system. It is convenient, however, to begin with a more constrained view of user interaction with the system in which there are two main paths corresponding to the Strategic and Tactical user classification. Such paths can be viewed as the normal workflows visualized in Fig. 13.7. Both types of users start entering inputs at the Adaptable User Interface. Such elicited information is mapped into elements of experimental frames (EF) for later interaction with the models. After all available data have been entered, processing proceeds to the PSS and then the DPAS modules. At this point the paths bifurcate according to the stakeholder characterization, where either the Market (Strategic) or Physical (Tactical) simulator service is invoked. Simulation results are then feed back to the user interface through the analysis and evaluation services.

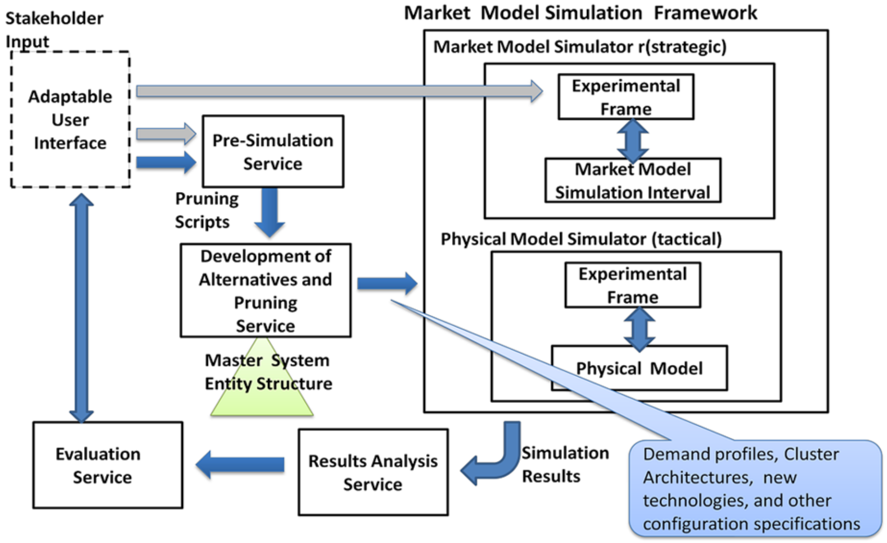

Stakeholder Input
Adaptable User Interface
Pre-Simulation Service
Pruning Scripts
Development of Alternatives and Pruning Service
Master System Entity Structure
Market Model Simulation Framework
Market Model Simulator r(strategic)
Experimental Frame
Market Model Simulation Interval
Physical Model Simulator (tactical)
Experimental Frame
Physical Model
Simulation Results
Demand profiles, Cluster Architectures, new technologies, and other configuration specifications
Evaluation Service
Results Analysis Service

Fig. 13.7 MSE Workflow showing typical service sequencing (*purple*) and information flow (*grey*)

This will typically initiate an iterative process in which the user's processing cycles several times through the corresponding work flow until arriving at a satisfactory outcome.

Although convenient as a first conception, such constrained workflows do not offer the flexibility we seek. Accordingly, we exploit the reconfigurability inherent in a Service Oriented Architecture with orchestration of the offered services to provide much greater flexibility. Such flexibility allows users to flow through the processing steps at will bypassing intervening steps if appropriate. For example, users may iterate between the input GUI and the ES in order to try out different weights on their value attributes. Or a user may alternate between the Strategic and Tactical stances to gain the perspectives of both views on the feasibility of his/her technology proposal. In another case, especially after the system has accumulated simulation experience, the user may bypass simulation and employ estimates of cluster worth offered up the built-in learning mechanisms. Moreover, such flexibility allows users to interact intermittently with the system over time, building up individual profiles of work that provide a basis for starting from accumulated experience rather than from a clean slate at any time.

13.3.2 System Entity Structure (SES): Key Support for MSE Flexibility

The core component of the MSE necessary for its user-adaptive flexibility is the Development and Pruning of Alternative Service (DPAS). As indicated earlier, the DPAS generates instances of clusters that can potentially meet the user's require-

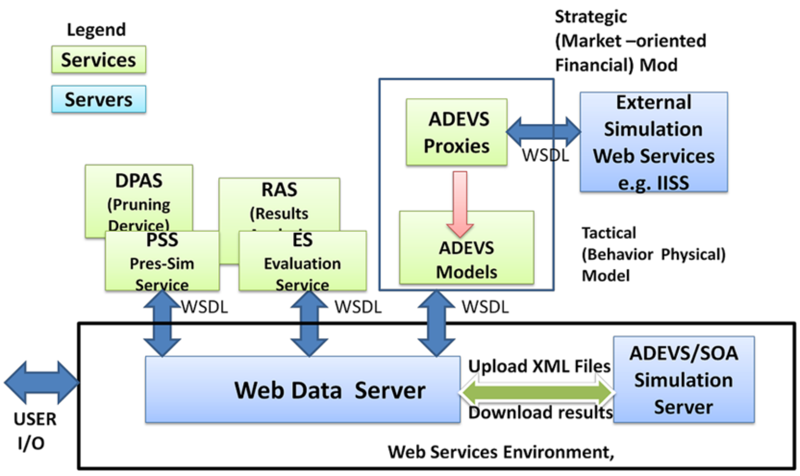


Fig. 13.8 Overall implementation of MSE

ments. The key enabler of such generation is a Master System Entity Structure (SES) (Chap. 3, Hagendorf and Pawletta 2010) which is the overall specification of all possible components and their relationships. In previous chapters, we have discussed tools to support specification of SES's in constrained natural language format, to prune SESs so as to result in well-defined model specifications, and to transform such pruned entity structures (PES) to executable simulation models. Further, a constrained natural language approach to pruning not only allows easy manual pruning but also enables automated specification through input pruning scripts. This capability provides a key element in achieving the flexibility to adapt to user requirements. In the standard workflow of Fig. 13.7, the PSS outputs a pruning script to the DPAS which prunes the Master SES to constrain the model's structure (viz., the cluster architectures) to be evaluated via simulation. The family of PESs generated by the DPAS from a single pruning script constitutes the solution space. These PESs are encoded in XML, and with the help of an XMLToOWL converter, stored in the Common Data Service (OWL-S 2004). In this form, they are available on demand to the simulators, whether Strategic or Tactical, as well as to other Frontier services (such as the RAS).

13.3.3 MSE Implementation: Service Orient Architecture

Figure 13.8 illustrates the implementation of the MSE as a set of web service components that can be configured to adapt to stakeholder requirements. In the initial phase of the Frontier project, we implemented the constrained workflow of Fig. 13.2, all information exchanges between services are mediated by the common data service supported by the Web Data Server. This is envisioned within a Web Services Environment that supports a "Semantic Bus" to be described shortly.

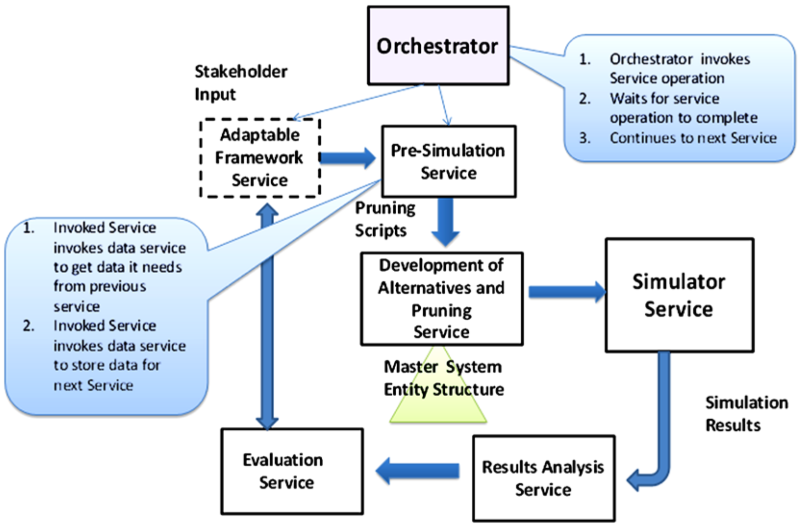

Orchestrator
1. Orchestrator invokes Service operation
2. Waits for service operation to complete
3. Continues to next Service
Stakeholder Input
Adaptable Framework Service
Pre-Simulation Service
Pruning Scripts
1. Invoked Service invokes data service to get data it needs from previous service
2. Invoked Service invokes data service to store data for next Service
Development of Alternatives and Pruning Service
Simulator Service
Master System Entity Structure
Simulation Results
Evaluation Service
Results Analysis Service

Fig. 13.9 Constrained orchestration of the MSE workflow

In the constrained workflow, an ad hoc function serves as orchestrator to move the processing steps along the paths illustrated in Fig. 13.9. However, a more flexible orchestration is required to implement automatic invocation of modeling services in such a way as to maximize value for the stakeholder. Depending on interests, objectives, and values, the stakeholder is categorized into one of the four basic categories discussed earlier, and different modeling services and different pipelines of services may be appropriate. Eventually an intelligent learning system can make such decisions. However, initially the Orchestrator must be seeded with some criteria for selecting services and invoking them in a particular order, with outputs from some services passed as input to others.

There are two overall methods for seeding such information.

- Hard coded representation in the orchestrator source code
- Representation in a process (service orchestration) specification.

We want to avoid hard coding such knowledge because it will be difficult to change as the experience with the system accumulates. There are well-established precedents for representing service orchestration in high-level specification languages, e.g., the Business Process Execution Language (2012), which is XML-based, and OWL-S (2004) in the semantic technology world. Because Frontier is a semantics-based system, an OWL-based representation is most appropriate. This representation will reside in the Common Data Service and implemented in TripleStore (2012), thus providing support for orchestrating services.

The next level up in flexibility from a static OWL representation is to provide intelligence to perform simple matching between the declared capabilities of the basic MSE services (outlined in Fig. 13.8) and the declared needs (including values) of the user. We have argued above that the categorization into 4 types of users (along Strategic/Tactical and Supply/Demand dimensions) provides a good initial basis for such matching, particularly with respect to choice of model abstraction for simulation.

Eventually, an intelligent learning system can make better matches, i.e., those that maximize value for the stakeholder. The easiest and most simple-minded form of matching will look for explicit matches, i.e., equal values, between corresponding attributes (OWL properties) of the Frontier services and the user requiring the services. Such matching will, at first, be crude, but it will be more than syntactic signature matching because the properties represent semantic information about both the demand and supply side. More advanced matching will involve dynamic orchestration decisions, such as examining the results from one service and inferring—at that point—whether they should be passed to another service, or whether perhaps the previous inputs should be passed to another service, etc. Whether such learned rules and/or choices can themselves be represented in OWL or must be kept in a sub-symbolic neural representation is not yet clear; but in either case, the goal is to provide the ability to adapt services to stakeholder needs and different notions of stakeholder value. We can envision that the orchestrator will maintain models for the users which represent their beliefs, goals, and intentions in relation to using the tools and executing the steps toward getting the results they expect. The concept of agents, as represented in DEVS, discussed in Chap. 10, is appropriate here.

13.3.4 MSE Simulation Service

The Simulation Service constitutes another key component in the realization of the overall objective of the Frontier design environment. Recall that its task is to provide an environment that caters to the interests of a wide variety of stakeholders in the construction of a novel architecture of fractionated satellites. Indeed, the MSE relies on the Simulation Service to provide simulations that evaluate spacecraft system architectures in response to diverse stakeholders' requirements. The Simulation Service includes both the Strategic and Tactical Simulation Services and simulates architectural models in response to the generated demands and supplies results for evaluation and ranking. The simulation is configured to either strategic or tactical forms depending on stakeholder interest. The Tactical Simulation Service encapsulates a DEVS coupled model containing an Experimental Frame (EF) (see Chap. 18) that interacts with the Physical Model Simulator (PMS) as seen in Fig. 13.10. The EF generates space system service requests over a simulation time interval to the PMS. The parameters of the service request stream are derived from the demand profile elicited through interactions with the stakeholder. The PMS simulates the processing of these requests based on a cluster configuration that has been developed for it by the DPAS. A cluster consists of a networked group of satellites evaluated in the simulation. The cluster may be made up of specific satellites, or satellites may rotate in and out of the cluster due to line of sight considerations or module hardware/software failures. The total time encompassed by each simulation is the same for all instantiations and each run of a specific cluster instantiation results in a single series of time-ordered events that describe in specific changes in the cluster

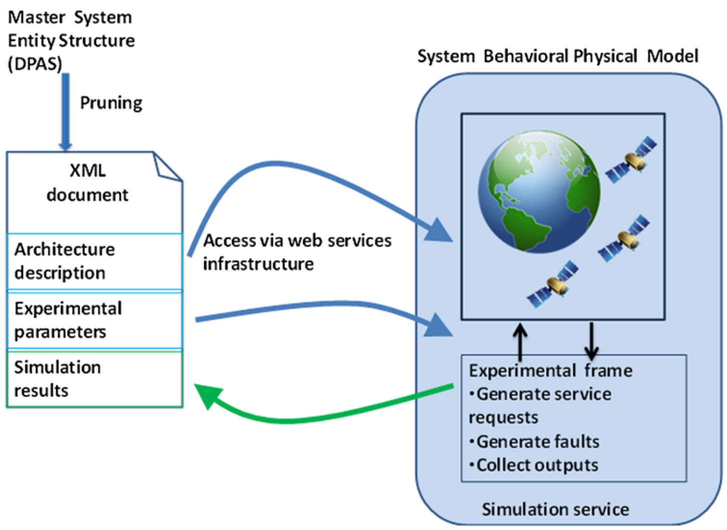


Fig. 13.10 Simulation Service with encapsulated Experimental Frame and PMS

as the simulation progresses. During the simulation, outputs of the PMS are continually sent to the EF as events occur. At the end of the simulation interval the EF provides the values for performance measures that it has developed by summarization and statistical operations performed on the received data. The performance metrics originate from the input elicited through interaction with the stakeholder.

The Market Model Simulator (Strategic Level) consists of an Experimental Frame that interacts with a Market Model (MM) to perform financial evaluation over a long time span composed of successive Simulation Analysis Intervals (e.g., quarterly). The goal of the simulation is to generate time series data that can be used in the market analysis in the RAS. As for the tactical case, the simulations use instantiations of clusters generated by the DPAS and are designed to output data that will support the evaluation of the values, goals, and metrics derived from the stakeholder input by the elicitation process. As shown in Fig. 13.11 the MMSF contains an Experimental Frame that interacts with the MM. Similarly to the interaction with the PMS, the EF receives input (via the common data service) from the DPAS in an XML format. The total time encompassed by each simulation run remains constant for all simulations of all instantiation for a given cluster. Each simulation run of a specific cluster instantiation will result in a single series of time-ordered events that describe in specific changes in the cluster as the simulation progresses.

In contrast to the PMS, in the interaction with the MM, this series of changes is divided into time periods during which the cluster configuration and atomic services do not change. These periods of stable cluster configuration are referred to as Simulation Time Periods (STPs) and consist of a varying number of Simulation Analysis Intervals (SAIs). Each STP and the cluster configuration associated with it define the data that are output to the common data service and stored in Triple-Store for market evaluation. Because a single series of events generated by s single

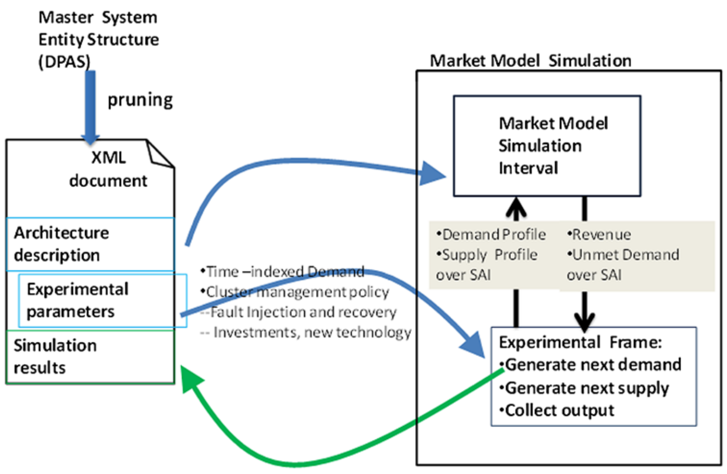


Fig. 13.11 Interaction of the Experimental Frame

simulation run cannot be considered an adequate description of the entire range of possible event strings, multiple simulation runs are conducted with the same cluster and environment. The data gathered from all the simulations runs are aggregated to generate probability distributions on the parameters of interest. These distributions describe the range of possible outcomes for the parameters of interest and provide the basis for the risk analysis component of the RAS.

13.3.5 Simulation Using Web Services

As illustrated in Fig. 13.8, the implementation of Frontier Simulation Services is based on the extension of the open source ADEVS (2012) environment to support simulation using Web Services. This work was comprised of three major steps: (1) enabling models expressed in ADEVS to be provisioned on server hosts and to be simulated in a federation employing web technologies, (2) adapting the simulation coordinator in the Main Service to execute in C++ and to exchange simulation control messages with the simulators, (3) adapting the simulators to exchange DEVS payload messages in XML format. These steps were accomplished with the help of the Apache Axis2C and Staff tools for web service development (Apache Axis2C 2012). The resulting environment, called ADEVS/SOA, allows ADEVS coupled models to be executed on an open-source Tomcat-based SOA platform.

In contrast in the DEVS/SOA (Seo and Zeigler 2012) environment based on DEVSJava (Sarjoughian and Zeigler 1998), there are limitations to the ADEVS/SOA environment that should be noted:

- ADEVS/SOA (C++) does not support dynamic instantiation of ADEVS models.
- ADEVS/SOA (C++) does not support reflection functions for Object classes' variables.
- ADEVS/SOA does not support dynamic creation of XML DEVS messages from ADEVS messages.

- ADEVS/SOA does not create ADEVS Simulator Services with uploaded ADEVS models.

These limitations, stemming from the C++ language, imply that work must be done in individually tailored, rather than automated, fashion to integrate an ADEVS model to execute on ADEVS/SOA. In particular, simulation servers must be individually provisioned with simulator services with pre-assigned atomic models. In contrast in the DEVS/SOA environment, atomic models can be downloaded to generic simulation servers and locally compiled. An ADEVS/SOA Simulation Client takes a folder containing pruned entity structure (PES) XML files and sends the selected XML file to an ADEVSMainService hosting simulation services. Such a client operates in the following sequential manner:

1. The client selects the PES XML file from a resident folder at its machine
2. The selected XML file is uploaded to the ADEVSMainService
3. The client invokes the start simulation service of the ADEVSMainService to coordinate ADEVS Simulator Services.

Once the simulation is over, the aggregated simulation logs from various servers are forwarded to the client's machine.

Also as shown in Fig. 13.8, ADEVS/SOA can reach out to external web service simulations through ADEVS proxy models that can participate in ADEVS coupled models while invoking remote web services. Due the limitations of C++ libraries for dynamic invocation of web services, creation of ADEVS proxies is not as convenient as in the ingestion process developed for DEVS/Java. However, ADEVS proxies can be generated with a Java-based program using a similar process in DevsJava because Staff tools provide libraries for dynamic invocation and XML handling functions. We conclude that despite its execution performance advantages, a C++-based environment is not as suitable as a Java-based counterpart for flexibility and extensibility such as envisioned for the Frontier environment.

13.4 MSE in Operation: An Example Thread

We follow an end-to-end workflow thread in which cluster architecture is compared against a monolithic satellite for the same input service demand profile. A master cluster architecture containing satellites that can have different types of sensors and different communication capabilities is illustrated in the SES partially displayed:

```
From the sys perspective, GeneralClusterArch is made of
ExperimentalFrame and SatelliteModules!
From the mult perspective, SatelliteModules are made of more
than one SatelliteModule!
SatelliteModule can be id in index!
From the activity perspective, SatelliteModule is made of
Energizing,Propulsing, Communicating, Navigating, Controlling,
and Sensing!
From the subSensSys perspective, Sensing is made of Sensor!
From the subCommSys perspective, Communicating is made of
Communication!
From the subCDHSys perspective, Controlling is made of
```

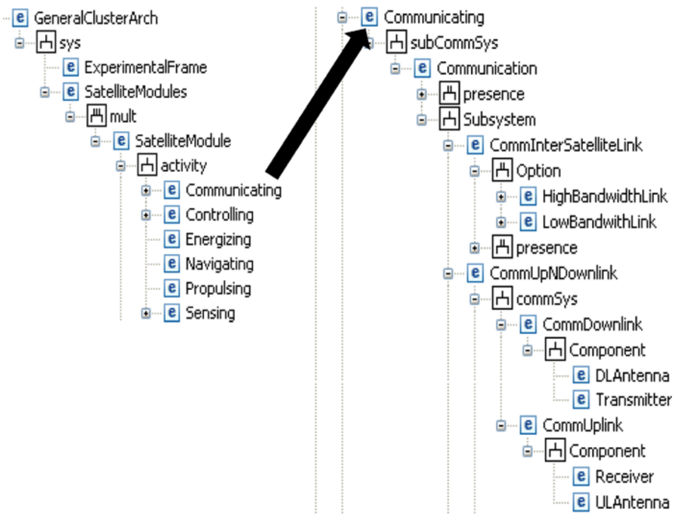


Fig. 13.12 Master SES provides a characterization of service demands and potential architectures to be explored

```
CommNDataHandling!
Sensor can be Present or NotPresent in presence!
Sensor can be Visual,InfraRed,Radar, or MultiCapability in
EMType!
...
```

Note the use of the multi-aspect concept for SatelliteModule to allow any number of modules to be used to make a Satellite system. The top level of satellite decomposition, called activity aspect, includes functions that any satellite optionally has. These functions are further broken down at the next level into decompositions with components that can implement the functions and/relevant specializations. Two such functions, Communications and Sensing are broken out in the Fig. 13.12.

The Master SES can be pruned to generate different cluster configurations, or instantiations, which are points in the solution space. Of course, the user's requirements will dictate which points count as solutions for him/her. Cluster instantiations include dimensions such as:

- Cluster size (number of satellites in the cluster)
- Satellite configuration (e.g., which satellites have which types of sensors)
- Atomic Services (modules) to be included in the cluster
- Functional dependencies
- Communications capabilities(e.g., intersatellite or down to ground)
- Module specific parameters
- Mean times to failure
- Sensor and communication footprints
- Susceptibility to external influences.

The PSS generates scripts that condition the SES to generate a more focused solution space. Some of the operations available for use are illustrated by the following:

- A pruning operation (e.g., ***select Radar from EMType for Sensor!***) selects Radar from the EMType specialization for every Sensor—a context specification can be added to restrict the selection to only those Sensors in the context.
- An SES restructuring operation (e.g., ***don't select Radar under Sensor!***) eliminates the selection of Radar from specializations that occur under Sensor everywhere Sensor occurs.
- Statements that check resulting PESs and accept only those that satisfy specified conditions:
  - set count bounds for Visual_Sensor as [0,2]!
  - set count bounds for CommUpNDownLink as [1,2]!

The distinction between general clusters and a monolithic baseline is embodied in different count bounds for the general cluster case (e.g., ***set count bounds for SatelliteModule as [3,6] !***), and for the monolithic case e.g., (***set count bounds for SatelliteModule as [1,1] !***).

A particular specialization, *presence*, has a specific connotation (e.g., ***select Present from presence for CommInterSatelliteLink!***) results in the presence of CommInterSatelliteLink in the PES vice (***select NotPresent from presence for CommInterSatelliteLink!***) in which CommInterSatelliteLink is eliminated from the PES. The first selection applies to the general clusters case while the second is appropriate to the monolithic case.

The output of the DPAS is a collection of PESs expressed as XML documents. In principle, such a PES can be interpreted by any suitable model. However in practice, models are tuned to the objectives they are built to serve. Reflecting this circumstance, an SES for Physical Model is less inclusive in scope and more detailed in physics than the Master SES. The SES for Physical Model is partially displayed:

```
From the topSys perspective, FrontierPMS is made of ExpFrame and
SatelliteModules!
From the multiSatellite perspective, SatelliteModules are made
of more than one SatelliteModule!
SatelliteModule can be id in index!
SatelliteModule has ID!
The range of SatelliteModule's ID is int!
SatelliteModule can be ImageSat or RelaySat in moduleType!
From the image perspective, ImageSat is made of
SensorModule,CommModule, and Orbit!
SensorModule has fov, viewRange, and imageBits!
...
```

As with the GeneralClusterArch SES, the use of the multi-aspect concept for SatelliteModule allows any number of modules to be used to make a Satellite system. However, in this case, SatelliteModule has a specialization in ImageSat or RelaySat so that there can be any number of image and relay satellites, respectively, as illustrated in Fig. 13.13 (also see Chap. 6).

A mapping from GeneralClusterArch (Master) SES to Physical SES relates the two abstractions as illustrated in Fig. 13.14.

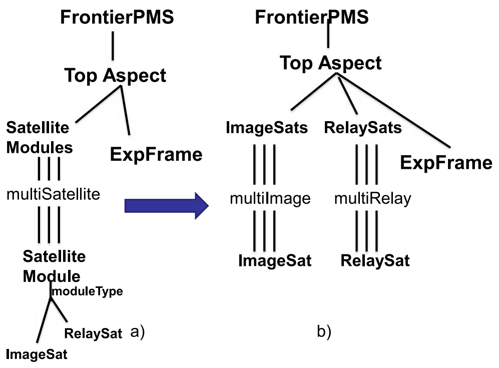


Fig. 13.13 Illustrating the restructuring of SatelliteModules into ImageSats and RelaySats

One main difference between the GeneralClusterArch SES and the Physical Model SES is that the latter distinguishes satellites as either imaging or relay while the former treats all satellites uniformly with the distinction arising from the selection of alternatives in the pruning phase. Thus, as discussed in Chap. 11, the mapping actually is at the pruned level so that a pruned GeneralClusterArch SES is mapped to a pruned Physical Model SES. Indeed, the mapping expressed at the XML level constructs an XML document configuring a Physical Model from an XML document representing configured cluster architecture. To illustrate the mapping, consider the following illustrated in Fig. 13.14:

```
For each SatelliteModule in the cluster,
if
   both an inter-satellite communication link (or any type) and
   a Sensor (of any type) are present in the substructure under
   the SatelliteModule,
then
   add an imageSat to the Physical Model cluster.
```

A similar rule to create relay satellites requires the presence of inter-Satellite communications and Up/Down Links to Earth as well as the absence of a Sensor. This kind of mapping has some very desirable attributes:

- It allows pruning of the Master SES to consistently configure multiple model abstractions—manual configuration of a single simulator model is difficult let alone consistent settings across multiple models.
- It allows the simulator to fill in missing information that it knows best (e.g., the mapping need not determine all choices in the Physical Model SES, only those that guarantee a compatible structure).
- It provides constraint criteria for improving Master pruning yield (percentage of pruned SES's that are desired instantiations). This needs further explanation. The mapping from master cluster architecture PESs to Physical Model PESs is actually not defined on the full domain of pruned entity structures, i.e., many master PESs do not yield a valid Physical Model configuration. For example, a SatelliteModule with no inter-satellite communications does not correspond to either imaging or relay assignment. To increase the probability of generating

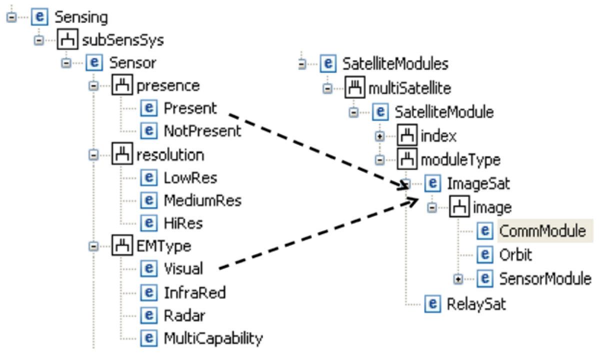


Fig. 13.14 Mapping from Master SES to SES for Physical Model

a meaningful Physical Model configuration, we should require that an inter-satellite communication link be present in every Satellite. This can be done by an appropriate pruning illustrated above. Thus, the yield of the pruning operation can be identified as the size of the range set of the mapping. This can be increased by properly constraining the domain of the mapping i.e., judiciously applied conditioning pruning rules.

- The mapping supports criteria for validation of the source SES. Assuming the mapping is correct, and then any violation must be laid at the foot of the mater SES itself.

Mapped clusters are simulated in the PMS and evaluated according to metrics of interest to the user. When invoked, the simulator service gets the given cluster instantiation and experimental frame data from the common store and starts execution. Since the latter data is the same for all candidate clusters, the performance evaluation of candidate clusters produced by the simulation allows them to be compared on equal footing. A ranking of instantiations may contain several monolithic architectures mixed in with truly fractionated clusters. If under a wide variety of conditions, the fractionated clusters dominate the monolithic cluster, the case for fractionation would be established.

13.5 Summary

In this chapter, we have discussed the Frontier Modeling Support Environment whose goal is to provide the flexibility to adapt its workflows, tools, and models, to diverse stakeholders in the DARPA F6 program. We outlined the unique features of the MSE that support its use by a wide spectrum of potential users and developers of a system of fractionated spacecraft. These features include:

- identification of user types to enable routing the user through relevant processing stages,
- automated generation of model artifacts adapted to selected pathways,
- conditioning of the solutions space to increase the opportunities to find suitable fractionated architectures,
- flexible simulation services,

- consistent configuration across multiple abstraction models,
- semantics-based orchestration of service oriented architecture.

Joint MEASURE (Zeigler et al. 1999) is a simulation system that evolved to measure utility of intelligence collection assets and strategies. The MSE was developed to abstract and re-implement Joint MEASURE's features on the SES and DEVS/SOA platform. Although the MSE is in its initial capability phase of development, the major features just stated have been demonstrated in a prototype. Much remains to be done including design and implementation of the semantics-based orchestration and an automated approach to mappings of the Master SES to incorporated abstractions. We also need to go beyond to the current pair of models (physical and market) to populating the environment with services and models to address a full range of stakeholder's objectives. As discussed above, the ingestion process for external web services is key to such extensibility, and may require adoption of Java-based, rather C++-based, proxies. Beyond the ability to exchange messages and invoke services enabled by such proxies, the perennial problem of harmonizing the data formats and operations protocols of external tools must be tackled. An ontologies-based approach is under-development consistent with the development of the semantics-based data store discussed above. The advance of Semantic Web technology and the development of pragmatics-based data engineering (Zeigler and Hammonds 2007) provide some hope that significant progress can be made, particularly within a restricted domain such as spacecraft architectures.

As is readily apparent, the approach taken in the design and development of the Frontier MSE is based on fundamental principles that have application much beyond spacecraft fractionated systems. This generic quality of the MSE concept suggests the applicability of basic design to virtual build and test of today's system of systems.

Acknowledgement This research was supported in part by the DARPA F6 Program.
Technical area 1: Design Tools for Adaptable Systems.

Appendix

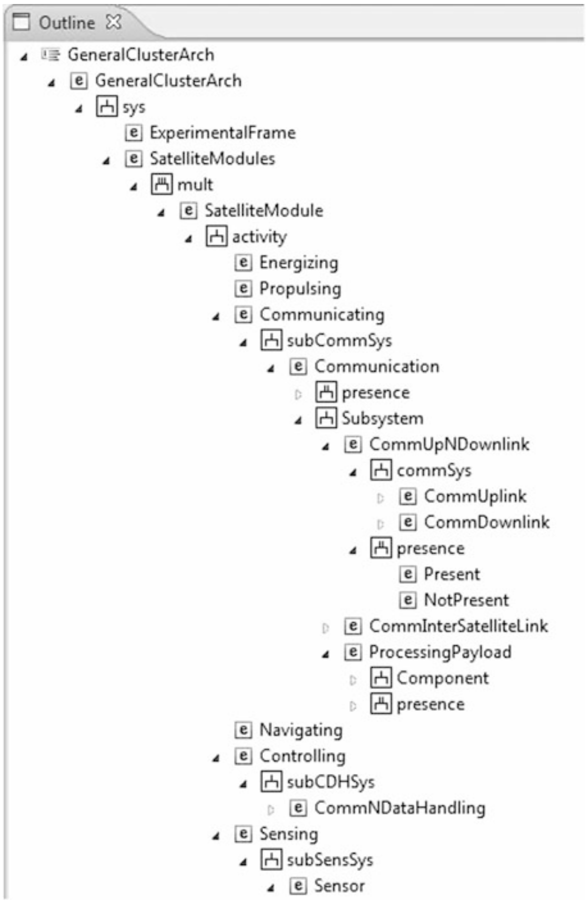
A.1 GeneralClusterArchSeS.txt

```
From the sys perspective, GeneralClusterArch is made of
ExperimentalFrame and SatelliteModules!

From the mult perspective, SatelliteModules are made of more
than one SatelliteModule!

SatelliteModule can be id in index!


From the activity perspective, SatelliteModule is made of
Energizing, Propulsing, Communicating, Navigating, Controlling,
and Sensing !


From the subSensSys perspective, Sensing is made of Sensor!

From the subCommSys perspective, Communicating is made of
Communication!

From the subCDHSys perspective, Controlling is made of
CommNDataHandling!


Sensor can be Present or NotPresent in presence!

Sensor can be Visual,InfraRed,Radar, or MultiCapability in
EMType!

Sensor can be SingleBand or MultiBand in band!

Sensor can be LowRes, MediumRes, or HiRes in resolution!

Sensor can be LowTolerant or HighTolerant in faultTolerance!

Sensor can be Stereoscopic or Monoscopic in stereo!

Sensor can be StabalizedPointing or NonPointing in
pointingCapability!

Sensor can be DataIntensive or NotDataIntensive in dataHandling!


From the SensoryFunctionA perspective, Sensor is made of
EarthObservation, and RemoteSensing!

EarthObservation can be Present, or NotPresent in presence!

RemoteSensing can be Present, or NotPresent in presence!


From the SensoryFunctionB perspective, Sensor is made of
SpaceObservation, and DataCollecting!

SpaceObservation can be Present, or NotPresent in presence!

DataCollecting can be Present, or NotPresent in presence!


Communication can be Present in presence!

From the Subsystem perspective, Communication is made of
CommUpNDownlink, CommInterSatelliteLink, and
ProcessingPayload!

From the commSys perspective, CommUpNDownlink is made of
```

```
CommUplink and CommDownlink!

From the Component perspective, CommUplink is made of ULAntenna,
and Receiver!

CommUpNDownlink can be Present, or NotPresent in presence!

CommInterSatelliteLink can be Present, or NotPresent in presence!


From the Component perspective, CommDownlink is made of
DLAntenna, and Transmitter!

CommInterSatelliteLink can be HighBandwidthLink, or
LowBandwithLink in Option!

HighBandwidthLink can be LaserLink in HBLink!

LowBandwithLink can be RadioLink in LBLink!

From the Component perspective, ProcessingPayload is made of
OnBoardProcessor, SpaceQualifiedRouter, Diplexer, Coupler,
Transciever, Amplifier, Modulator, Filter, Mixer,
Frequency_ClockGenerator, and Multiplexer!

ProcessingPayload can be Present, or NotPresent in presence!


CommNDataHandling can be Present or NotPresent in presence!

From the SubSystem perspective, CommNDataHandling is made
of CDHUplink, CDHDownlink, and TelemetryTracking!

From the component perspective, CDHUplink is made of
ReceiverAntenna, and ReceiverSystem!

From the component perspective, CDHDownlink is made of
TransmitterAntenna, and TransmitterSystem!

From the Subsystem perspective, TelemetryTracking is made of
OnBoardComputer, Processor, and IntersatelliteLink!

Sensor has weight,volume, and cost!
The range of Sensor’s weight is double !

The range of Sensor’s volume is double !

The range of Sensor’s cost is double !

Sensor has fov, viewRange, and imageBits!

if select Present from presence for CommUpNDownlink then
select Present from presence for Communication!
```

```
if select Present from presence for CommInterSatelliteLink then
select Present from presence for Communication!
```

A.2 Outline of GeneralCusterSeS



```
GeneralClusterArch
  GeneralClusterArch
    sys
      ExperimentalFrame
      SatelliteModules
        mult
          SatelliteModule
            activity
              Energizing
              Propulsing
              Communicating
                subCommSys
                  Communication
                    presence
                    Subsystem
                      CommUpNDownlink
                        commSys
                          CommUplink
                          CommDownlink
                        presence
                          Present
                          NotPresent
                      CommInterSatelliteLink
                      ProcessingPayload
                        Component
                        presence
              Navigating
              Controlling
                subCDHSys
                  CommNDataHandling
              Sensing
                subSensSys
                  Sensor
```

A.3 GeneralClusterArchBasicPrune.pes

```
don't select Radar under Sendor !
don't select MultiCapabilty under Sendor!

don't select LowRes under Sensor !

don't select Stereoscopic under Sensor !

don't select MultiBand from band under Sensor !
```

```
don't select StabilizedPointing under Sensor !

don't select DataIntensive under Sensor !

restructure multiaspects using index !

set multiplicity of index as [6] for SatelliteModule !

set count bounds for SatelliteModule as [3,6] !

select Present from presence for CommInterSatelliteLink !

set count bounds for Visual as [1,2] !

set count bounds for Infrared as [1,2] !

set count bounds for CommUnNDownLink as [1,2] !

set count bounds for ProcessingPayload as [1,2] !

set count bounds for LowTolerant_Visual_Sensor as [0,2] !

set count bounds for HighTolerant_InfraRed_Sensor as [0,1] !

set count bounds for Visual as [0,1] per SatelliteModule !

set count bounds for Infrared as [0,1] per SatelliteModule !

set count bounds for CommInterSatelliteLink as [1,2] per
SatelliteModule!

set count bounds for CommUnNDownLink as [0,1] per
SatelliteModule !

set count bounds for ProcessingPayload as [0,1] per
SatelliteModule !
```

A.4 GeneralClusterArchMonolithicPrune.pes

```
don't select Radar under Sensor !

don't select MultiCapabilty under Sensor !

don't select LowRes under Sensor !

don't select Stereoscopic under Sensor !

don't select MultiBand from band under Sensor !

don't select StabilizedPointing under Sensor !
```

```
don’t select DataIntensive under Sensor !

restructure multiaspects using index !

set multiplicity of index as [6] for SatelliteModule !

set count bounds for SatelliteModule as [1,1] !

select NotPresent from presence for CommInterSatelliteLink !

set count bounds for Visual as [1,2] !

set count bounds for Infrared as [1,2] !

set count bounds for CommUnNDownLink as [1,2] !

set count bounds for ProcessingPayload as [1,2] !

set count bounds for LowTolerant_Visual_Sensor as [0,2] !

set count bounds for HighTolerant_InfraRed_Sensor as [0,1] !
```

References

ADEVS (2012). An open source C++ DEVS simulation engine. http://www.ornl.gov/~1qn/adevs/index.html.

Apache Axis2C (2012). http://axis.apache.org/axis2/c/core/.

Aumann, G. A. (2007). A methodology for developing simulation models of complex systems. *Ecological Modelling*, *202*, 385–396.

Business Process Execution Language (2012). http://en.wikipedia.org/wiki/Business_Process_Execution_Language.

Hagendorf, O., & Pawletta, T. (2010). Framework for simulation-based structure and parameter optimization of discrete event systems. In G. A. Wainer & P. J. Mosterman (Eds.), *Discrete-event modeling and simulation: theory and applications*. Boca Raton: CRC Press.

Kim, T. G., et al. (2010). DEVSim++ toolset for defense modeling and simulation and interoperation. *Journal of Defense Modeling and Simulation*, *8*(3), 129–142.

Kim, T. G., Sung, C. H., Hong, S.-Y., Hong, J. H., Choi, C. B., Kim, J. H., Seo, K. M., & Bae, J. W. (2011). DEVSim++ toolset for defense modeling and simulation and interoperation. *The Journal of Defense Modeling and Simulation: Applications, Methodology, Technology*, *8*(3), 129–142.

OWL-S (2004). http://www.w3.org/Submission/OWL-S/.

Rubinstein, R., & Kroese, D. (2007). *Simulation and the Monte Carlo method. Wiley series in probability and statistics* (2nd ed.). New York: Wiley.

Sarjoughian, H. S., & Zeigler, B. P. (1998). DEVSJAVA: basis for a DEVS-based collaborative M&S environment. In *Proceedings of the SCS international conference on web-based modeling and simulation*, San Diego (Vol. 5, pp. 29–36).

Seo, C., & Zeigler, B. P. (2012). Simulation model standardization through web services: interoperation and federation on the DEVS/SOA platform. In *DEVS intergrative M&S symposium, proceedings of the spring simulation conference*, Orlando, FL, March 2012.

Service-Oriented Architecture (2012). http://en.wikipedia.org/wiki/Service-oriented_architecture.

Staff (2012) Open source web services framework for C++ based on Axis2/C. http://code.google.com/p/staff/.

TripleStore (2012). http://en.wikipedia.org/wiki/Triplestore.

Zeigler, B. P., & Hammonds, P. (2007). *Modeling & simulation-based data engineering: introducing pragmatics into ontologies for net-centric information exchange*. New York: Academic Press.

Zeigler, B. P., Hall, S. B., & Sarjoughian, H. (1999). Exploiting HLA and DEVS to promote interoperability and reuse in Lockheed's corporate environment. *Simulation Journal*, *73*(4), 288–295.

Zeigler, B. P., Kim, T. G., & Praehofer, H. (2000). *Theory of modeling and simulation* (2nd ed.). New York: Academic Press.

Zeigler, B. P., Nutaro, J., Seo, C., Hall, S., Clark, P., Rilee, M., Bailin, S., Speller, T., & Powell, W. (2012). Frontier modeling support environment: flexibility to adapt to diverse stakeholders. In *Symposium on theory of modeling & simulation—DEVS integrative M&S symposium*. Orlando: SpringSim.