SYSTEMS ENGINEERING
Research Center

Communications Effects Server (CES) Model for Systems Engineering Research

Final Technical Report SERC-2012-TR-025

Principal Investigator: Dr. Robert Cloutier, Stevens Institute of Technology

Team Members

Peter Korfiatis, Research Assistant, Stevens Institute of Technology
Kyle Thompson-Bass, Research Assistant, Stevens Institute of Technology

Contract Number: H98230-08-D-0171 DO1, TTO2, RT0025
Report No. SERC-2012-TR-025
January 31, 2012

UNCLASSIFIED

Report Documentation Page	*Form Approved* *OMB No. 0704-0188*

1. REPORT DATE **31 JAN 2012**	2. REPORT TYPE	3. DATES COVERED **00-00-2012 to 00-00-2012**

4. TITLE AND SUBTITLE **Communications Effects Server (CES) Model for Systems Engineering Research**	5a. CONTRACT NUMBER
	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **Systems Engineering Research Center,Stevens Institute of Technology ,1 Castle Point on Hudson,Hoboken,NJ,07030**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR’S ACRONYM(S)
	11. SPONSOR/MONITOR’S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES

14. ABSTRACT
All systems and system of systems (SoS) have an architecture ? either explicit or implicit. The architecture of the system impacts many downstream design decisions and the longer lifecycle issues that impact the cost of system maintenance, security and integration with other systems. As systems become more interconnected into system of systems, they must operate in a network centric environment. These networks may be a collection of disparate networks, wireless networks, networks of people, and some unknown future form of connectivity. This is such a daunting concern that industry competitors have created the Network Centric Operations Industry Consortium (NCOIC) for the purpose of working together to improve the probability that individually architected systems from different companies will interoperate with one another. One of the many challenges in this environment is to predict system performance as the system(s) architecture is being designed, and then to validate that performance when the system is prototyped and later instantiated. The good viable approach to this is through model-based systems architecting.

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**	**Same as Report (SAR)**	**74**	

This page intentionally left blank

ABSTRACT

All systems and system of systems (SoS) have an architecture – either explicit or implicit. The architecture of the system impacts many downstream design decisions, and the longer lifecycle issues that impact the cost of system maintenance, security, and integration with other systems. As systems become more interconnected into system of systems, they must operate in a network centric environment. These networks may be a collection of disparate networks, wireless networks, networks of people, and some unknown future form of connectivity. This is such a daunting concern that industry competitors have created the Network Centric Operations Industry Consortium (NCOIC) for the purpose of working together to improve the probability that individually architected systems from different companies will interoperate with one another.

One of the many challenges in this environment is to predict system performance as the system(s) architecture is being designed, and then to validate that performance when the system is prototyped and later instantiated. The good viable approach to this is through model-based systems architecting.

This page intentionally left blank

TABLE OF CONTENTS

LIST OF FIGURES

1 SUMMARY

The goal of this task was to determine the feasibility of establishing a systems architecture modeling and assessment (SAMA) environment within the SERC, with the Army's CES (Communications Effects Server) modeling tool as a key component. While CES was originally intended to assess network centric system performance, this research investigated applying the tool to architecture assessment prior to putting the system in the field. In Phase 1 the research team delivered an architecture description document which modeled the CES software architecture to further understand the ability to extend CES to the more abstract application of architecting.

What was found was that the CES is coded at a very low level, and will not easily be used for abstract architecting. The amount of information needed to run a simulation is too detailed and unknown during early product development.

Additionally, the high cost of the core license ($50,000) makes this a very expensive product for system architecting. It is recommended that the research associate with the CES product not continue.

2 INTRODUCTION

The CES is a network centric architecture tool, meant to model wired and wireless networks and provide performance characteristics for communications networks based on real world communications equipment data.

The ADD was developed as a deliverable of an ongoing research task between the Systems Engineering Research Center (SERC) and the US Army Research, Development and Engineering Command (RDECOM) Communications-Electronics Research, Development and Engineering Center (CERDEC) Space & Terrestrial Communications Directorate (S&TCD) (Research Topic ID DO2/TTO2/0023).

The main objectives of this research include: [SERC 2010]

1. Research, review and assess the applicability of the CES modeling environment as a research platform for model based architecture assessment. This work should also include the extensibility of the CES for other classes of systems beyond those currently modeled.
2. Its applicability to a model based architecting environment, to allow rapid architectural assessment from an end to end systems performance perspective.
3. Make recommendations for further verification and validation of the network centric architecture (performance) modeling environment, CES.
4. Understand the quality of the models and the modeling approach to allow extensibility to assessing the PEO-I BCT-M (or similar) architecture from a number of additional perspectives such as network reliability and dependability, network resilience, and security.

The Architecture Description Document was created to provide the researchers with a full understanding of the CES in order to make informed recommendations and assessments based on these research objectives.

3 WORK PERFORMED

The team utilized an HP Elitebook 8540W with an Intel i7 processor, and 8 GB RAM for this research. They acquired the necessary loaner CES licenses from the research sponsor, and began the process of learning and understanding the functionality and capability of the CES. As part of the process, an Architecture Description Document (ADD) was created. The ADD specifies the architecture for the Communications Effects Server (CES) software package.

Considerable effort was spent on understanding the CES. There was very little end user documentation, and the bulk of the provided information documented the code in the form of programmer manuals for CES. The research team captured issues uncovered as well as a high level model of the CES design. One researcher (Kyle Thompson-Bass visited Ft Monmouth twice a week to work with the CES team for most of 2010.

The first IPR occurred on Dec. 7 at Fort Monmouth. Status updates were provided by the team to Mr. McKeon. A SysML workshop was also scheduled to occur, but participants were called away, and the workshop was postponed (and in fact, it was never rescheduled by the sponsor due to the continued uncertainty of personnel transfers to Aberdeen Proving Ground). However, work continued on the modeling of the CES high level architecture. Once that was understood, the team began system decomposition and activities/use case development. Modeling then evolved to include logical architecture and it's traceability to previously established use cases and activities.

ARCHITECTURE DESCRIPTION DOCUMENT

CES was operational and in the field for almost a decade, and many of the original architecture artifacts were not available. Therefore, the developed ADD should be classified as "Architecture of Existing Systems" according to IEEE 1471 (commonly referred to as "as built" in industry):

> "The built system will have an architecture (since every system has an architecture, whether known or not) but it will lack an architectural description. In this case, an ADD can be created through a reverse-engineering effort."

As an ADD of an Existing System, the reverse engineering effort performed was based on available artifacts, including user and programmer manuals, conversations with CERDEC stakeholders and knowledge gained while experimenting with CES. This delivered ADD was intended to serve as the baseline architecture that might be expanded in the future.

The format of this ADD is largely influenced by the Software Engineering Institute's Software Architecture Documentation Template [SEI 2004], as well as the IEEE Standard 1471-2000 [IEEE 1471].

This ADD was organized into the following sections:

- **Section 1 – Documentation Roadmap.** Provides information about this document and its intended audience. Provides the roadmap and document overview. Section 1 also provides information about the views that are used by this ADD to communicate the software architecture.
- **Section 2 – Architecture Background.** Explains why the architecture is what it is. It provides a system overview, establishing the context and goals for the development. It describes the background and rationale for the software architecture. It explains the constraints and influences that led to the current architecture, and it describes the major architectural approaches that have been utilized in the architecture. It includes information about evaluation or validation performed on the architecture to provide assurance it meets its goals.
- **Section 3 – Views.** Specifies the architecture. Views specify elements of software and the relationships between them in varying perspectives.
- **Section 4 – Architecture Traceability.** Describes the relationship between the views presented in Section 3.
- **Section 5 – Conclusions.** Contains a summary of finding, as well as Recommendations for Extensions of CES and a Roadmap for RT23 Phase 2 research efforts.
- **Appendix A – Directory.** Provides look-up information for documents that are cited elsewhere in this ADD and provides a glossary and acronym list.
- **Appendix B – Modeling Notation.** Explains some of the symbols utilized in this ADD

The team modeled the architecture using the Systems Modeling Language (SysML) which is managed by the Object Management Group (OMG). The tool used was Sparx Enterprise Architect, with the SysML plug-in. The CES domain diagram is shown in Figure 1.

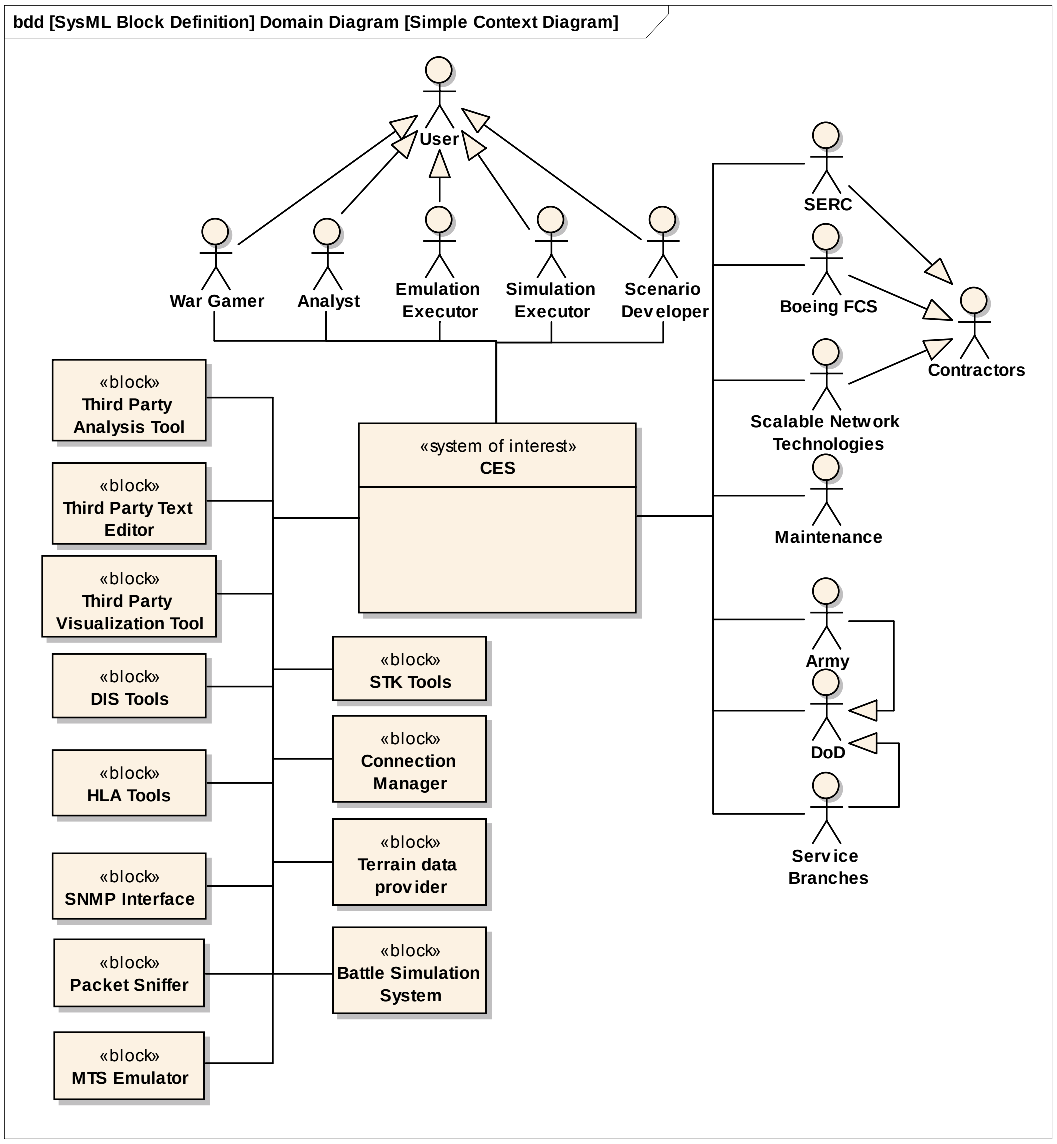

bdd [SysML Block Definition] Domain Diagram [Simple Context Diagram]
User
War Gamer
Analyst
Emulation Executor
Simulation Executor
Scenario Developer
SERC
Boeing FCS
Contractors
Scalable Network Technologies
Maintenance
Army
DoD
Service Branches
«system of interest» CES
«block» Third Party Analysis Tool
«block» Third Party Text Editor
«block» Third Party Visualization Tool
«block» DIS Tools
«block» HLA Tools
«block» SNMP Interface
«block» Packet Sniffer
«block» MTS Emulator
«block» STK Tools
«block» Connection Manager
«block» Terrain data provider
«block» Battle Simulation System

Figure 1 - CES Domain Diagram

The system of interest, CES, is placed in the center of the diagram, and surrounded by external elements that interact with it. These elements include systems that interface with CES, as well as stakeholders who use CES to accomplish their specific goals. This diagram only shows that a relationship exists between these elements and CES, and does not specify the directionality of these relationships or objects being exchanged between elements. A more detailed context diagram for the CES is found in Figure 2, showing the interfaces to external systems and operators.

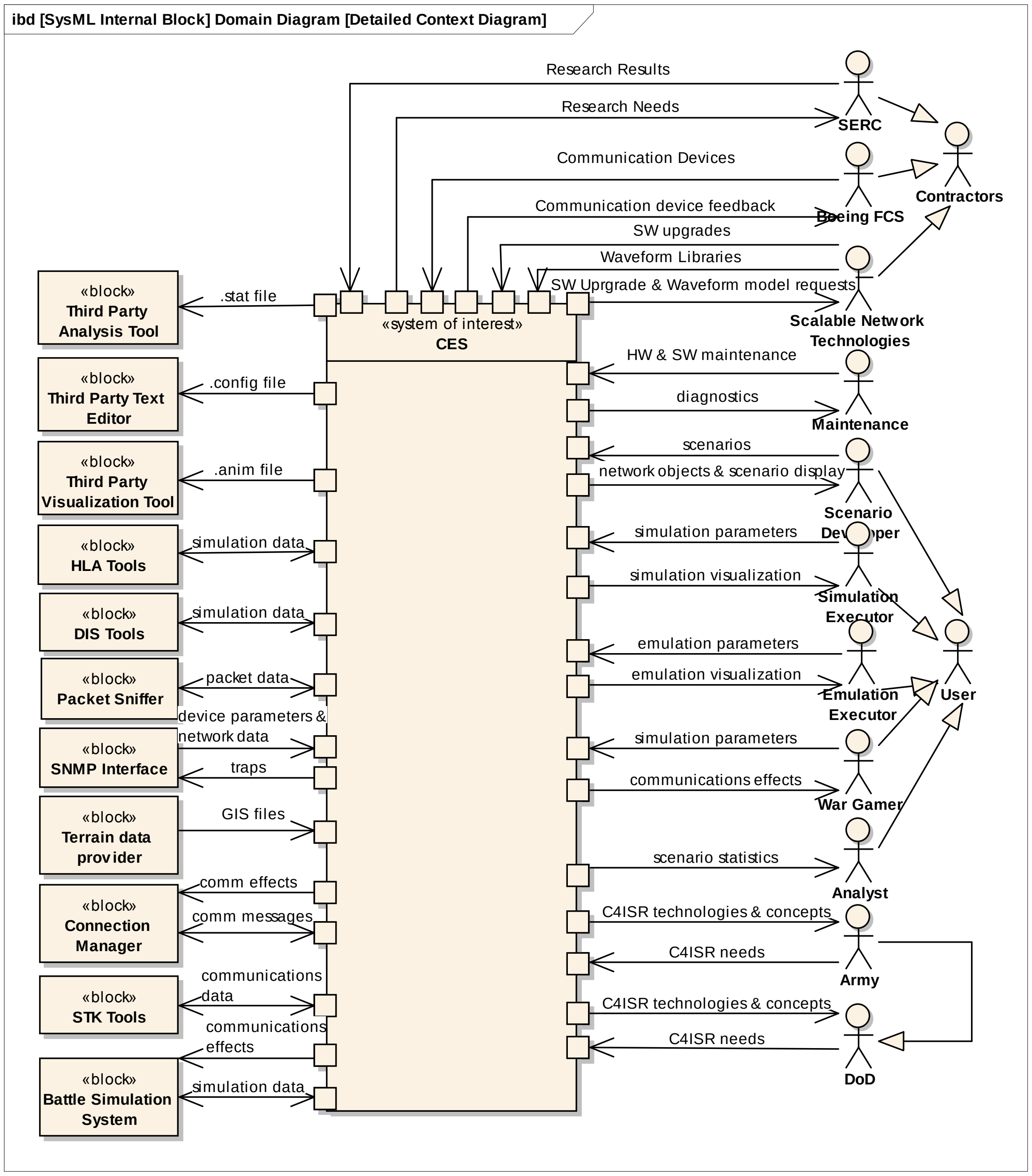


Figure 2 - Detailed CES Context Diagram

The ADD is included as an Appendix to this document.

RESEARCH RE-DIRECTION

There was a meeting conducted on March 11, 2011 at ARDEC, with the sponsors for RT 23 (Mr. Doug McKeon, CERDEC, Ft. Monmouth) and RT31 (Mr. Matthew Cilli, ARDEC, Picatinny) to discuss the potential of bringing these two research efforts closer together. It was determined that RT23 (this research effort) could prove to play an important role within RT31's CONOPS Navigator, which would also expand CES's applicability as a model based architecture tool. Since RT31 is sponsored by the ARDEC, combination of CES with RT31 research provides a unique opportunity to integrate two separate Army products to provide better capabilities to the warfighter and systems development personnel.

One major outcome of the meeting was an agreement of the two research sponsors that phase 2 of RT23 would be to combine the effort of RT23 and RT31. While RT23 and RT31 would not be fully integrated, it was agreed upon that deliverables of RT23 and RT31 will be developed together, to investigate whether or not CES could be integrated as another tool for use in conjunction with RT31's CONOPS Navigator. This agreement was documented in the March status report for RT23.

Another outcome of the March 21 meeting was for Stevens to initiate the necessary paperwork to obtain a Contractor's License to OneSAF. It was suggested that this tool might also enable a link between CES and CONOPS Navigator. This too changed the direction of RT23.

ONESAF INTEGRATION

In addition to providing an established interface to CES, OneSAF was chosen as an integration tool due to its widespread use at ARDEC (RT31 sponsor). Based on the ARDEC sponsor's experiences with OneSAF, three projects were discussed for validation of RT23/RT31 developments. Of the three, the Homeland Security project was chosen. The project involved development of a OneSAF simulation to assess the preparedness dimension of emergency management in response to terrorist attack scenarios.

Working with emergency response agencies, the sponsor developed a OneSAF model of a local train station and local emergency response assets. Simulations were then executed, using a number of scenarios to assess emergency response to potential terrorist attacks.

The Homeland Security project was chosen for a number of reasons, benefitting both RT 23 and RT31 research goals:

- The project was mature, providing a full model that had been fully developed and validated by stakeholders.
- The project material was unclassified, and could be used freely during research and development.
- The scenarios modeled were applicable to the operational context under which CES is used. In the Homeland Security project scenarios,

communication would be a crucial factor. The developed OneSAF model neglected to account for communications at a low level of fidelity; therefore application of CES to this scenario would enhance the existing OneSAF simulation.

- The project was directly relevant to RT23 research goals. An established goal of RT23 was to explore the possibility of CES integration with other modeling and simulation tools across different domains. Application of CES to the Homeland Security project would investigate the feasibility of this tool to provide information to decision makers in a Homeland Defense scenario.

Given the decision to explore OneSAF integration with CES, a license request was made to OneSAF Program Office on March 25, 2011. Because OneSAF provided the most immediate value to the RT23 effort, the RT23 sponsor authorized release of OneSAF to the research team. Since OneSAF also provided value to RT31 research, when the contract period for RT23 elapsed, the OneSAF license release was transferred to the RT31 sponsor, on October 11, 2011.

Work was also performed to examine options for direct integration between CES and Unity development environment and CES/OneSAF/Unity integration.

4 Research Challenges

There were some significant challenges that made this research extremely challenging.

Scalable Network Technologies

The architecture of the CES was built on a Scalable Network Technologies product. This product required the purchase of a $50,000 license to run CES. There was never intent for the SERC research team to acquire a license for this research. Therefore, every 30 days CERDEC would request another temporary license to enable the SERC research to continue.

Over time, it appears it became increasingly difficult for the research sponsor to secure temporary licenses, and the team would go months between new license activations.

CERDEC Personnel Changes

There were a number of research sponsor personnel changes over the course of this research which resulted in discontinuity. The project began with Mr. Oral Walker as the Program Manager, and Mr. Doug McKeon as the Technical Lead. By the time the research was completed, a new Program Manager had been assigned, and the Technical Lead had retired, with no replacement named.

5 DELIVERABLES

There were three deliverables defined for this research. They are:

1) CES Architecture Description Document
2) Assessment of "As-is" architecture for extendibility for new capability
3) Phase 2 Research Roadmap

Deliverable 1 was discussed in Section 3, and is included as Appendix B of this report.

PHASE 2 RESEARCH ROADMAP

One of the deliverables for this research was a research roadmap. Since the team is recommending the research not be continued, there is questionable value in producing the roadmap. However, to that end, the following recommendations are provided.

As documented in the Architecture Description Document, the research team recommends the following extensions to CES, if the Army decides to extend the capability.

RECOMMENDATION 1:

Create functionality to allow for users to develop scenarios in a Systems Modeling Language (SysML) tool (such as Sparx Enterprise Architect) and convert them into input (.config files) for CES. This would allow for a more abstract beginning to the scenario development, which focuses CES more on the designing of new systems instead of the validation of completed ones. It also allows for the user to utilize artifacts created by model based system engineering efforts within CES, and vice versa.

Implementation: Import/export of SysML data is typically managed through XML files, therefore to develop this functionality an XML parser would be needed to take XML data and convert it into a .config file. The .config file could then be read by CES as well as translated back to SysML notation. Likely the SysML model will not configure all parameters of the .config file, but rather create an outline which would then be filled in with the CES GUI or by manually editing the configuration files.

Potential Obstacles:

- Requires new 3rd party software to run.

- Changes to the syntax of the .config file during versioning by Scalable Technologies could cause mayhem with the parser. Would likely have to be modified for each new version of CES.
- Requires knowledge of SysML
- Complexity of configurable parameters in CES possibly beyond the scope of SysML.

RECOMMENDATION 2:

Extend the analysis capabilities of CES by creating a rules based analysis of the simulation statistics. This would work by monitoring the statistics output of a scenario and comparing it to allowed value ranges; if a metric is outside the acceptable range an alert is created. This could also be extended to provide real-time alerts in an emulation scenario that could let a user know that there has been a communications failure.

Implementation: Using the MySQL database interface, key feedback statistics could be parsed and presented in a variety of tools. Real-time emulation implementation would be more complex. Currently CES writes out statistics only during the finalization process. In order to do real time monitoring you would need to access metric data as it was being created.

Potential Obstacles:

- CES provides extensive statistics. Determining what an acceptable range for each metric may not be easy. It may be best to start with the most important and basic metrics, such as packet loss or throughput.
- Changes in versioning by Scalable Technologies could potentially cause issues for the rules analysis.

RECOMMENDATION 3:

Further integration of CES with other Army Architecting and Analysis Tools should be conducted. One popular software package in use by the Army for battle space simulation is OneSAF. CES has a low level of integration with tools like OneSAF, which could be expanded to provide additional functionality of the CES.

Implementation: Exploration of current Army tools would highlight the most relevant existing tools in use today, and further analysis would yield the value and specific applications of CES integration with these tools.

Obstacles:

- Creating useful feedback from CES. Could combine with intelligent analysis to provide more informative feedback for battle space simulation software.

ASSESSMENT OF "AS-IS" ARCHITECTURE FOR EXTENDABILITY FOR NEW CAPABILITY

As documented in the Architecture Description Document, Figure 3 provides recommendations where the CES Architecture might be extended with new capability.

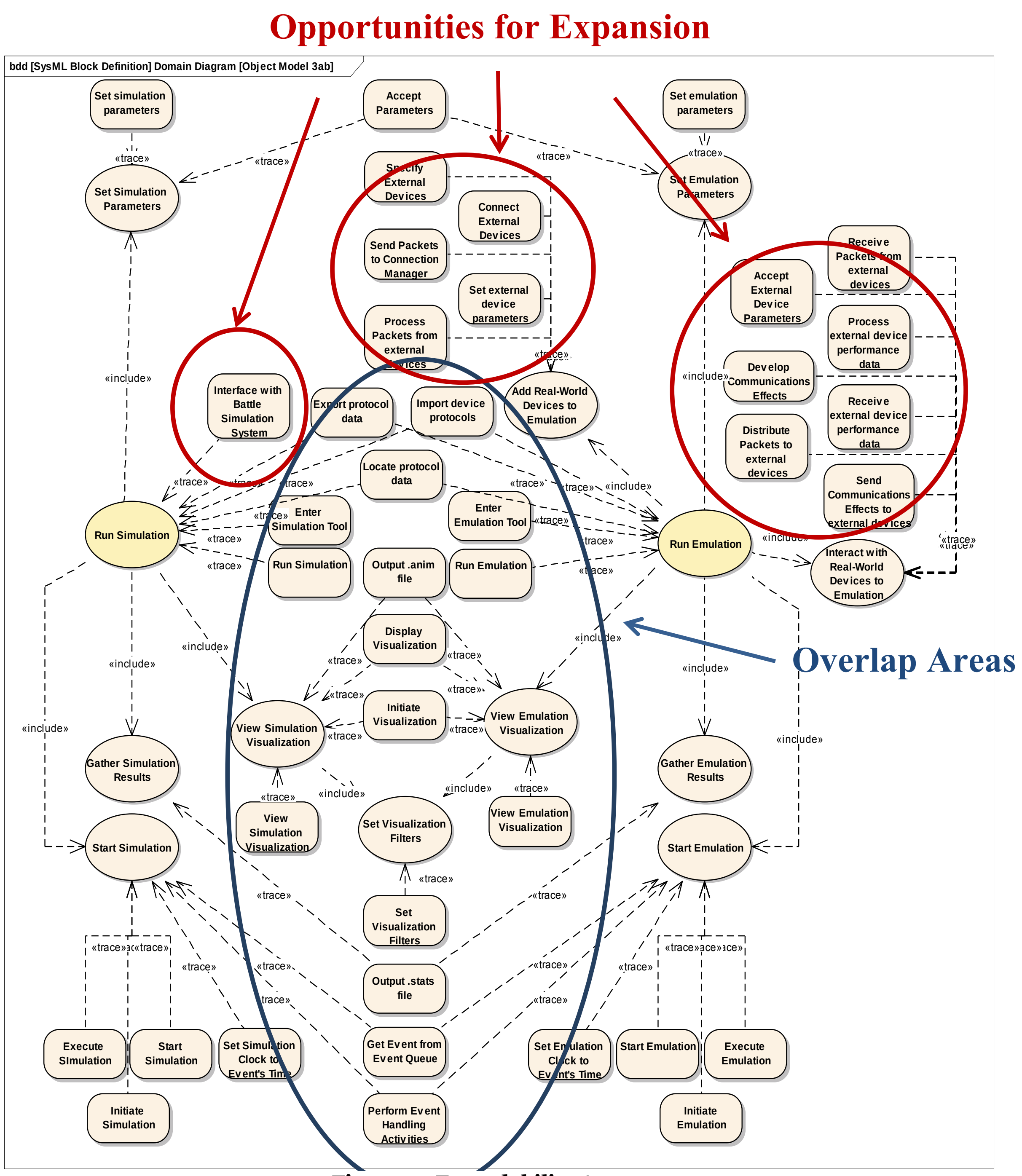

Opportunities for Expansion
bdd [SysML Block Definition] Domain Diagram [Object Model 3ab]
Set simulation parameters
Accept Parameters
Set emulation parameters
Set Simulation Parameters
Specify External Devices
Connect External Devices
Set Emulation Parameters
Send Packets to Connection Manager
Set external device parameters
Receive Packets from external devices
Accept External Device Parameters
Process Packets from external devices
Process external device performance data
Develop Communications Effects
Interface with Battle Simulation System
Export protocol data
Import device protocols
Add Real-World Devices to Emulation
Receive external device performance data
Distribute Packets to external devices
Locate protocol data
Send Communications Effects to external devices
Enter Simulation Tool
Enter Emulation Tool
Run Simulation
Run Emulation
Interact with Real-World Devices to Emulation
Run Simulation
Output .anim file
Run Emulation
Display Visualization
Overlap Areas
Initiate Visualization
View Simulation Visualization
View Emulation Visualization
Gather Simulation Results
Gather Emulation Results
View Simulation Visualization
Set Visualization Filters
View Emulation Visualization
Start Simulation
Start Emulation
Set Visualization Filters
Output .stats file
Get Event from Event Queue
Execute Simulation
Start Simulation
Set Simulation Clock to Event's Time
Set Emulation Clock to Event's Time
Start Emulation
Execute Emulation
Initiate Simulation
Perform Event Handling Activities
Initiate Emulation
«trace»
«include»

Figure 3 - Extendability Assessment

6 CONCLUSIONS

Based on the research performed, the team has found this to be a highly specific product that does not lend itself to higher level abstractions. The documentation is at the code level of detail (programmers manuals) and it would take an experienced programmer months to learn the codebase to extend this product for communication system architecting. Each of the waveforms requires a high level of detail that would not be available when architecting the system.

That aspect, coupled with the high cost of the core license ($50,000) makes this a very expensive product for system architecting. It is recommended that the research associate with the CES product not continue.

As documented in the delivered Architecture Description Document, Figure 3 captures some of the potential areas for expansion for CES. However, it is the recommendation of the research team that this work be done by the developer of CES rather than as a research project.

In regards to the work with OneSAF and VBS2 which came out of the March meeting, that work will continue with ARDEC and the CONOPS Navigator.

APPENDICES

APPENDIX A: REFERENCES

The following documents were used during the course of this research:

- Future Combat Systems Software Version Description (SVD) for Communication Effects Server (CES) 5.3.1; Document # 786-0000087192; Revision: G; Release Date: August 26, 2009
- User's Guide for Communications Effects Server (CES) 5.3.1 with EXata 2.0.1, August 2009, Scalable Network Technologies Inc.
- EXata CTDB Model Library for CES 5.3.1, August 2009, Scalable Network Technologies Inc.
- EXata Developer Model Library for CES 5.3.1, August 2009, Scalable Network Technologies Inc.
- EXata Distributed Reference Guide for CES 5.3.1, August 2009, Scalable Network Technologies Inc.
- EXata Model Library Index for CES 5.3.1, August 2009, Scalable Network Technologies Inc.
- EXata Multimedia and Enterprise Model Library for CES 5.3.1, August 2009, Scalable Network Technologies Inc.
- EXata OTF Model Library for CES 5.3.1, August 2009, Scalable Network Technologies Inc.
- EXata Programmers Guide for CES 5.3.1, August 2009, Scalable Network Technologies Inc.
- EXata Standard Interfaces Model Library for CES 5.3.1, August 2009, Scalable Network Technologies Inc.
- EXata Urban Propagation Model Library for CES 5.3.1, August 2009, Scalable Network Technologies Inc.
- EXata Wireless Model Library for CES 5.3.1, August 2009, Scalable Network Technologies Inc.
- EXata Users Guide for CES 5.3.1, August 2009, Scalable Network Technologies Inc.
- EXata 2.0.1 Connection Manager Users Guide, April 2009, Scalable Network Technologies Inc.
- EXata Network Security Model Library for CES 5.3.1, August 2009, Scalable Network Technologies Inc.

APPENDIX B: ARCHITECTURE DESCRIPTION DOCUMENT

Systems Engineering Research Center (SERC)

Communications Effects Server (CES)
Architecture Description Document (ADD)

Version 1.0

Dr Robert Cloutier
Peter Korfiatis
Kyle Thompson-Bass
February 20, 2011

Table of Contents

Referenced Materials

List of Figures

List of Tables

1 Documentation Roadmap

1.1 Document Management

Rev. #	Revision Date	Purpose of Revision	Scope of Revision	Editor
0.1	01/16/11	CES ADD creation		PGK
0.2	01/23/11	Sec 1 Documented	Filled in Sec 1. Documentation roadmap	PGK
0.3	01/25/11	Sec 3 Documented	Started working on Sec 3	PGK
0.4	01/27/11	Sec 3 Documented	Completed Sec 3.3-3.4	PGK
0.5	01/29/11	Sec 3 Documented	Completed Sec 3.1- 3.2	PGK
0.6	02/01/11	Sec 3 Edits, Sec 4 Documented	Altered Sec 3 to reflect feedback, Started Sec 4	PGK
0.7	02/03/11	Sec 3 & 4 Edits Appendices Documented	Further altered Sec 3 & 4 to integrate feedback and keep sections consistent Added content to Appendices	PGK KTB
1.0	2/15/11	Final Approval by PI	Minor adjustments	RJC

1.2 Purpose and Scope of the ADD

This Architecture Description Document (ADD) specifies the architecture for the Communications Effects Server (CES) software package. The CES is a network centric architecture tool, meant to model wired and wireless networks and provide performance characteristics for communications networks based on real world communications equipment data.

This ADD is being developed as a deliverable of an ongoing research task between the Systems Engineering Research Center (SERC) and the US Army Research, Development and Engineering Command (RDECOM) Communications-Electronics Research, Development and Engineering Center (CERDEC) Space & Terrestrial Communications Directorate (S&TCD) (Research Topic ID DO2/TTO2/0023).

The main objectives of this research include: [SERC 2010]

1. Research, review and assess the applicability of the CES modeling environment as a research platform for model based architecture assessment. This work should also include the extensibility of the CES for other classes of systems beyond those currently modeled.

2. Its applicability to a model based architecting environment, to allow rapid architectural assessment from an end to end systems performance perspective.

3. Make recommendations for further verification and validation of the network centric architecture (performance) modeling environment, CES.

4. Understand the quality of the models and the modeling approach to allow extensibility to assessing the PEO-I BCT-M (or similar) architecture from a number of additional

perspectives such as network reliability and dependability, network resilience, and security.

The ADD presented here seeks to provide researchers with a full understanding of the CES in order to make informed recommendations and assessments based on these research objectives. With this ADD's purpose in mind, the scope and architecture views, components and decisions defined below may deviate from common software engineering practices to fulfill the objectives above. A more complete and standardized ADD would extend the information below to cover the full breadth and depth of the CES.

Since the CES has been operational and in the field for almost a decade, many of the original architecture artifacts are not available, so this ADD effort can be classified as "Architecture of Existing Systems" according to IEEE 1471 (commonly referred to as "as built" in industry):

> "The built system will have an architecture (since every system has an architecture, whether known or not) but it will lack an architectural description. In this case, an ADD can be created through a reverse-engineering effort."

As an ADD of an Existing System the reverse engineering effort here is based on available artifacts, including user and programmer manuals, conversations with CERDEC stakeholders and knowledge gained while experimenting with CES. This document will serve as the baseline for architectural changes moving forward.

1.3 How the ADD Is Organized

The format of this ADD is largely influenced by the Software Engineering Institute's Software Architecture Documentation Template [SEI 2004], as well as the IEEE Standard 1471-2000 [IEEE 1471].

This ADD is organized into the following sections:

- **Section 1 - Documentation Roadmap** - Provides information about this document and its intended audience. Provides the roadmap and document overview. Section 1 also provides information about the views that are used by this ADD to communicate the software architecture.
- **Section 2 - Architecture Background -** Explains why the architecture is what it is. It provides a system overview, establishing the context and goals for the development. It describes the background and rationale for the software architecture. It explains the constraints and influences that led to the current architecture, and it describes the major architectural approaches that have been utilized in the architecture. It includes information about evaluation or validation performed on the architecture to provide assurance it meets its goals.
- **Section 3 - Views -** Specifies the architecture. Views specify elements of software and the relationships between them in varying perspectives.
- **Section 4 – Architecture Traceability** - Describes the relationship between the views presented in Section 3.

- **Section 5 - Conclusions** – Contains a summary of finding, as well as Recommendations for Extensions of CES and a Roadmap for RT23 Phase 2 research efforts.
- **Appendix A - Directory** - Provides look-up information for documents that are cited elsewhere in this ADD and provides a glossary and acronym list.
- **Appendix B - Modeling Notation**– Explains some of the symbols utilized in this ADD

1.4 Stakeholder Representation

This section provides a list of the stakeholder roles related to the architecture described by this ADD. For each, the section lists the concerns that the stakeholder has that can be addressed by the information in this ADD.

Stakeholder Name	Concerns
User	Super class of stakeholder, concerned with aspects of using CES to accomplish its goals
Scenario Developer	The Scenario developer uses CES to architect the communication network scenario. His concern is proper development of Scenarios for Simulation and Emulation
Simulation Executor	Simulation personnel are responsible for running simulations of communications networks.
Emulation Executor	The emulation personnel is responsible for running emulations of communications networks, using virtual communication devices as well as hardware and software that exists in the field.
Analyst	Access to statistics resulting from CES Simulation and Emulation executions, in a coherent, easy to read and concise manner
War Gamer	The war gamer is a user who will use CES simulation capabilities to introduce communication effects into war gaming simulations.
Maintenance Personnel	Maintenance personnel are concerned with upkeep of CES software and hardware components, as well as providing assistance to CES users.
Department of Defense	DoD refers to the enterprise level DoD personnel, as well as other organizations and service branches. These entities are concerned with using the CES (as it is, with extensions or with modifications) to accomplish specific goals in each of their mission profiles.
Army (Other)	The Army (other than CERDEC) is concerned with using CES to aid in the development and analysis of C4ISR technologies and concepts as they exist and as they hope to be expanded.
CERDEC	CERDEC is the main sponsor for CES research. CERDEC is concerned with using CES to gain insight into building and integration of communications networks. A matter of concern is also expanding the CES to increase its applicability to other areas of science, technology and engineering within the DoD and private industry.
Contractors	Contractors are those who provide development and research associated with CES and the communication networks CES is to analyze. Their specific concerns are listed below.
Boeing FCS	Boeing FCS acts as the primary integrator of Future Combat System (FCS) components. The responsibilities of Boeing FCS include developing and modeling waveforms for the communication devices created to interact with the FCS.
Scalable Network Technologies	Scalable network Technologies is the developer of CES and its simulation and emulation engines. They are responsible for upgrading CES and its components as well as updating waveform libraries as Boeing FCS develops new communications devices.
SERC	Members of SERC are the authors of this ADD. Their concerns lie in understanding the CES architecture, and providing suggestions for possible improvements and expansions to the tool

Table 1: Stakeholders and Concerns

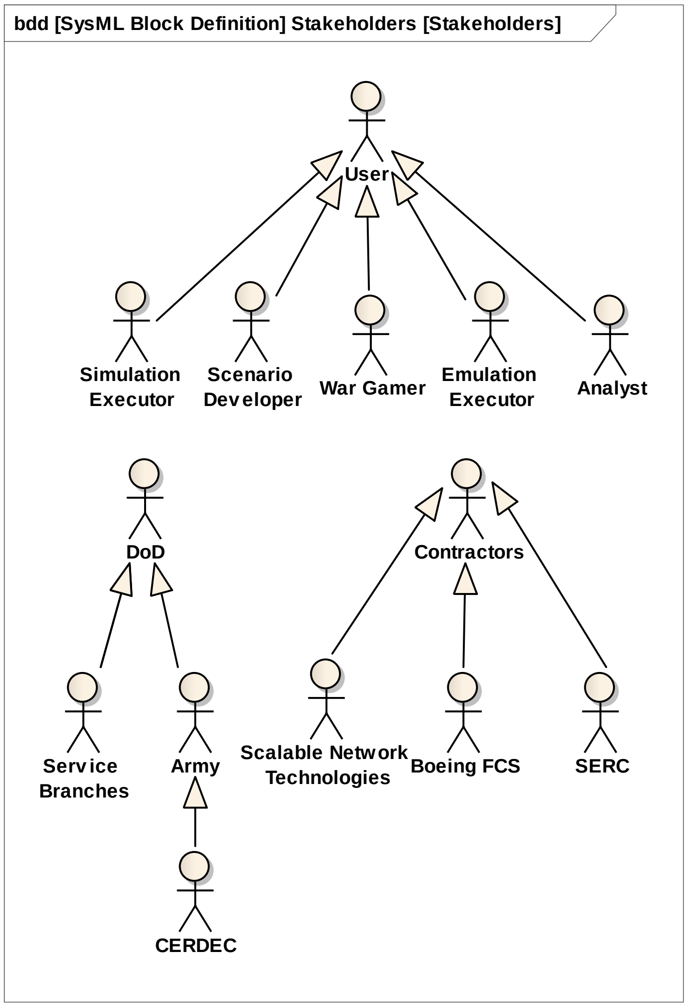


Figure 1: Select CES Stakeholders

1.5 View Definition

The ADD employs a stakeholder-focused, multiple view approach to architecture documentation, as recommended by IEEE 1471-2000. For those unfamiliar with IEEE 1471, the standard approaches architecture descriptions based on views and viewpoint. The concept of architectural views and viewpoint stems from the definition of software architecture as "the structure or structures of a system, which comprise software elements, the externally-visible properties of those elements, and the relationships among them" [Bass 2003]. In this ADD, a *view* is defined as the specification of one or more of these structures from the perspective of a related set of concerns. [IEEE 1471] Consistent with this definition, a *viewpoint* is a "pattern or template from which to develop individual views by establishing the purposes and audience for a view, and the techniques for its creation and analysis." [IEEE 1471]

Documenting an architecture, then, is a matter of documenting the relevant views and then documenting information that applies to more than one view [Clements 2002]. This approach means that the structure of the CES as described in this ADD is based on the specific views considered relevant in this environment, and the understanding of the authors.

Section 3 of this ADD contains multiple views of CES. The goal of the SERC CES research task is to understand CES as it exists today, and use this understanding to guide further expansions to current capabilities. To meet this goal, the ADD has been developed using the Kruchten 4+1 Model as a base set of architecture views [Kruchten 1995]. 4+1 Model views were adapted, and additional views were added, as deemed necessary to form a full understanding of CES. Views that will be utilized in this ADD include:

- System Context View – Describes the systems, applications and users the system of interest will interact with in order to accomplish its goals. The system context view will be modeled using a SysML Context Diagram, presented in Section 3.1. [Mitra 2008]
- Scenario View – Describes sequences of interactions between objects and processes. Kruchten specified that the most critical functions (highest frequency of use or presenting significant risk) be modeled using SysML Use Case diagrams, presented in Section 3.2 [Kruchten 1995]
- Process View – Describes the runtime behavior of the system, representing the major activities that take place, and their sequence. The process view will be modeled using SysML Activity Diagrams, which will be presented in Section 3.3. [Kontio 2005]
- Logical Architecture View – Describes the system decomposed into its main logical components and their relationships. The logical architecture view will be modeled using SysML block definition diagrams, which will be presented in Section 3.4 [Friedenthal 2008]

2 Architecture Background

2.1 Problem Background

2.1.1 System Overview

CES is a network simulation and emulation tool which allows the user to model a communications network in order to examine functionality in a variety of situations. It is useful for validating network design, testing network deployment, and providing communications effects for third party programs such as war game simulators. CES contains a network emulator that allows evaluation of on-the-move communication networks faster and with more realism than any other emulator. It creates a digital network replica that interfaces with real networks and applications.

2.1.2 Goals and Context

Quality of wireless communication is seriously impacted by a variety of channel and environmental factors. These may include weather, non-line of sight, interference, terrain and other factors that may be difficult to account for when designing, testing and integration new communications equipment into an existing network.

When considering networks with few devices, designing for interoperability of the network devices across a wide variety of environments and conditions is relatively easy. However, when increasing the number of devices to match the hundreds or thousands of components of today's net-centric DoD deployed and support forces, management of network design, testing and integration becomes increasingly complex. Frequently, communication networks come together in incremental fashion, with some components reaching maturity and fielding well before other components, adding to the complexity of testing and integration. Finally, adding in the variety of communication devices used by the DoD, and their message formats and priority levels, as well as the variety of environments in which such devices are deployed, a manual plug in/integrate/test paradigm is insufficient to account for the needs of communication device and network developers.

The goal of CES is to provide network developers and integrators with a tool to enable virtualization of whole or portions of communications networks for the purposes of simulating and emulating the impact of various communications effects on a proposed or real life communication network. CES can be utilized as a design, testing and integration tool through its main interface, as well as a participant in a variety of war games and battle simulations through the numerous external interfaces it supports.

2.2 Solution Background

2.2.1 Architectural Approaches

CES provides services to design and analysis across a full spectrum of network development. This is accomplished through its use of both a simulation and an emulation engine, allowing CES to be utilized for design, testing and integration of network devices.

Emulation and Simulation

A network emulator mimics the functions of a real network so that it appears, interacts, and behaves like the real network. The emulator provides an exact, high quality, reproduction of external behavior so that the emulated system is indistinguishable from the real system. An emulator provides a cost-effective method of evaluating new network technologies before actual systems or networks are built.

A network simulator duplicates the behavior of a real network, but cannot interact with real networks. A simulator uses lower quality reproduction or abstraction of the real system and focuses on simply replicating the real network's behavior. A network simulation is a very low cost method for developing the early stages of network centric systems. You can evaluate the basic behavior of a network and test combinations of network features that are likely to work.

Network emulation helps in developing a net-centric system by providing an environment in which design decisions can be easily changed and their impact evaluated. Customers of the net-centric system can use the emulated network and see how their applications (such as VoIP, situational awareness, sensor data, and streaming video) will perform when the real system is built. The emulated network can also be integrated with legacy systems to test interoperability and be used to train users on the next generation networks. By evaluating what works best early in the design cycle, the cost of modifying a system can be greatly reduced. This also sets realistic expectations of what the communications network will deliver, i.e., it provides predictability.

The key features of the CES emulation engine that enable creating a virtual network environment are:

- Speed

  Can support real-time speed to enable software-in-the-loop, network emulation, and hardware-in-the-loop modeling. Faster speed enables model developers and network designers to run multiple "what-if" analyses by varying model, network, and traffic parameters in a short time.

- Scalability

Can model thousands of nodes by taking advantage of the latest hardware and parallel computing techniques. EXata can run on cluster, multi-core, and multi-processor systems to model large networks with high fidelity.

- Model Fidelity

  Uses highly detailed standards-based implementation of protocol models. It also includes advanced models for the wireless environment to enable more accurate modeling of real-world networks.

- Portability

  Runs on a vast array of platforms, including Windows XP, Mac OS X, and Linux operating systems, distributed and cluster parallel architectures, and both 32- and 64-bitcomputing platforms. Users can now develop a protocol model or design a network on their desktop or laptop Windows XP computer and then transfer it to a powerful multi-processor Linux server to run capacity, performance, and scalability analyses.

- Extensibility

  Can connect to other hardware and software applications, such as OTB, real networks, and third party visualization software, to greatly enhancing the value of the network model.

3 Views

This section contains the views of the system architecture. A view is a representation of a whole system from the perspective of a related set of concerns [IEEE 1471]. Concretely, a view shows a particular type of architectural elements that occur in a system, their properties, and the relations among them.

3.1 System Context View

3.1.1 System Context View Description

The System Context View describes the CES system as a black box. All external elements, relevant stakeholders and interfaces are displayed. Included in the context diagram are the most important and relevant exchanges between these entities and CES. The System Context View is modeled here using SysML block definition and internal block diagrams.

3.1.2 System Context View Diagrams and Narrative

This ADD uses two diagrams to represent the System Context View, a block definition diagram (bdd) and an internal block diagram (ibd). The bbd in Figure 2 shows a simplified view of a context diagram. The system of interest, CES, is placed in the center of the diagram, and surrounded by external elements that interact with it. These elements include systems that interface with CES, as well as stakeholders who use CES to accomplish their specific goals. This diagram only shows that a relationship exists between these elements and CES, and does not specify the directionality of these relationships or objects being exchanged between elements. Figure 2 allows the viewer to see what CES, as a black box, interfaces with, and the personnel involved in its operation in a quick simple format. The elements contained within the Simple System Complex Diagram will be explained in detail in Sections 3.4.2.2 and 3.4.2.3.

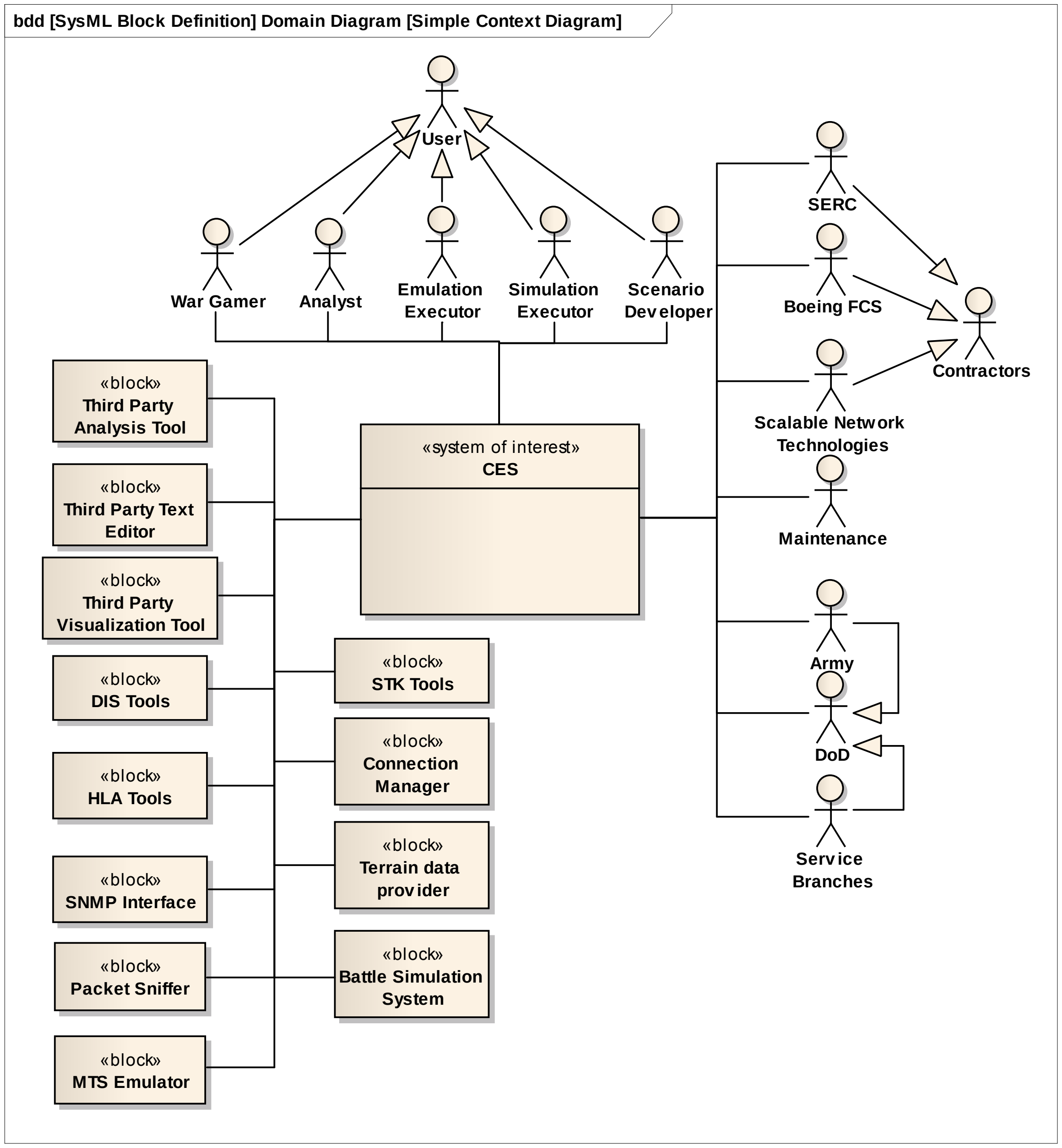

bdd [SysML Block Definition] Domain Diagram [Simple Context Diagram]
User
War Gamer
Analyst
Emulation Executor
Simulation Executor
Scenario Developer
SERC
Boeing FCS
Contractors
Scalable Network Technologies
Maintenance
Army
DoD
Service Branches
«system of interest» CES
«block» Third Party Analysis Tool
«block» Third Party Text Editor
«block» Third Party Visualization Tool
«block» DIS Tools
«block» HLA Tools
«block» SNMP Interface
«block» Packet Sniffer
«block» MTS Emulator
«block» STK Tools
«block» Connection Manager
«block» Terrain data provider
«block» Battle Simulation System

Figure 2: Simple Context Diagram

While Figure 2 is very simple and easy to read, it does not provide the modeler, viewer or system developer with any information regarding what specific data is being exchanged between elements, and which direction it is travelling in. To display this information, an internal block diagram was constructed, as seen in Figure 3. This more detailed diagram uses ports attached to the CES system of interest, and labeled directional connectors, to display information related to data and object exchange between CES and its external interfaces and stakeholders.

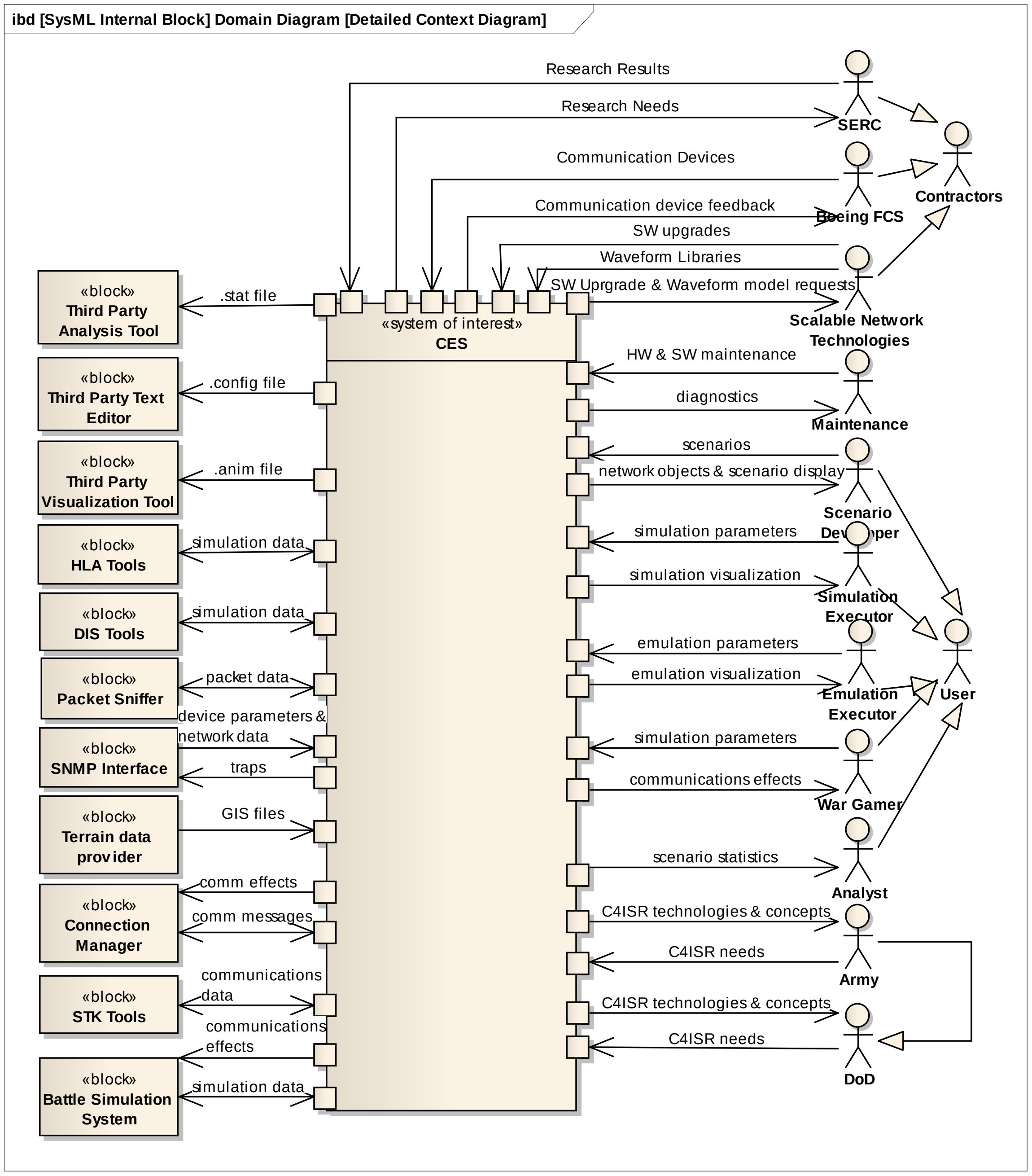


Figure 3: Detailed System Context View

As with Figure 2, the elements represented in Figure 3, as well as the data they import or export, will be specified in Sections 3.4.2.2 and 3.4.2.3. The ibd detailed System Context diagram provides viewers with information about objects transferred to and from CES, allowing a more in depth understanding of how CES interacts with its surroundings. Of particular importance is the relationship with the Connection Manager. This interface, transferring communications messages and effects, is the key component to facilitate CES's ability to execute emulations, allowing CES to model complex communications networks including both virtual and real world components. An understanding of Figure 3 also promotes consideration of internal CES components necessary to handle the external interfaces of CES. Using the Connection Manager as an example, it is clear that since the Manager is used for emulations only, it is likely to interface with only components of CES that are involved in emulation activities. On the other handle, HLA and DIS interfaces only communicate simulation data with CES, therefore they will most likely not interact with and CES internal emulation components.

The System Context View is an important view to understand how CES will interoperate with its external interfaces. By looking at the system from the outside-in, readers of this ADD are able to form a mental model of CES's interoperability characteristics, and the main components, both internal and external, that will facilitate this interoperation.

3.2 Scenario View

3.2.1 Scenario View Description

The Scenario View describes how the stakeholders interact with the CES to accomplish a set of goals. The Scenario View is represented as a collection of Use Case diagrams. These diagrams place elements of functionality within the CES system and trace each of these functions to the associated users and/or external systems.

3.2.2 Scenario View Diagrams and Narrative

Use Case diagrams are developed using multiple levels and maintaining inheritance among the elements at different levels. Figure 4 presents the major top level functionality of the CES system.

It is important to note that while CES users have been specified to the nature of their interaction with CES (Scenario Developers develop scenarios), it is likely that the personnel types will overlap. It is probable that a Scenario Developer will then analyze the statistics of a CES simulation or emulation. Conversely, simulation and emulation personnel serve two different mission profiles; simulation is most beneficial prior to design, perhaps as an Analysis of Alternatives, whereas emulation is crucial for communication device integration. Therefore, distinguishing the Use Cases related to Simulation and Emulation Executors in this view is useful when considering the possible adaptation of CES for different missions and objectives.

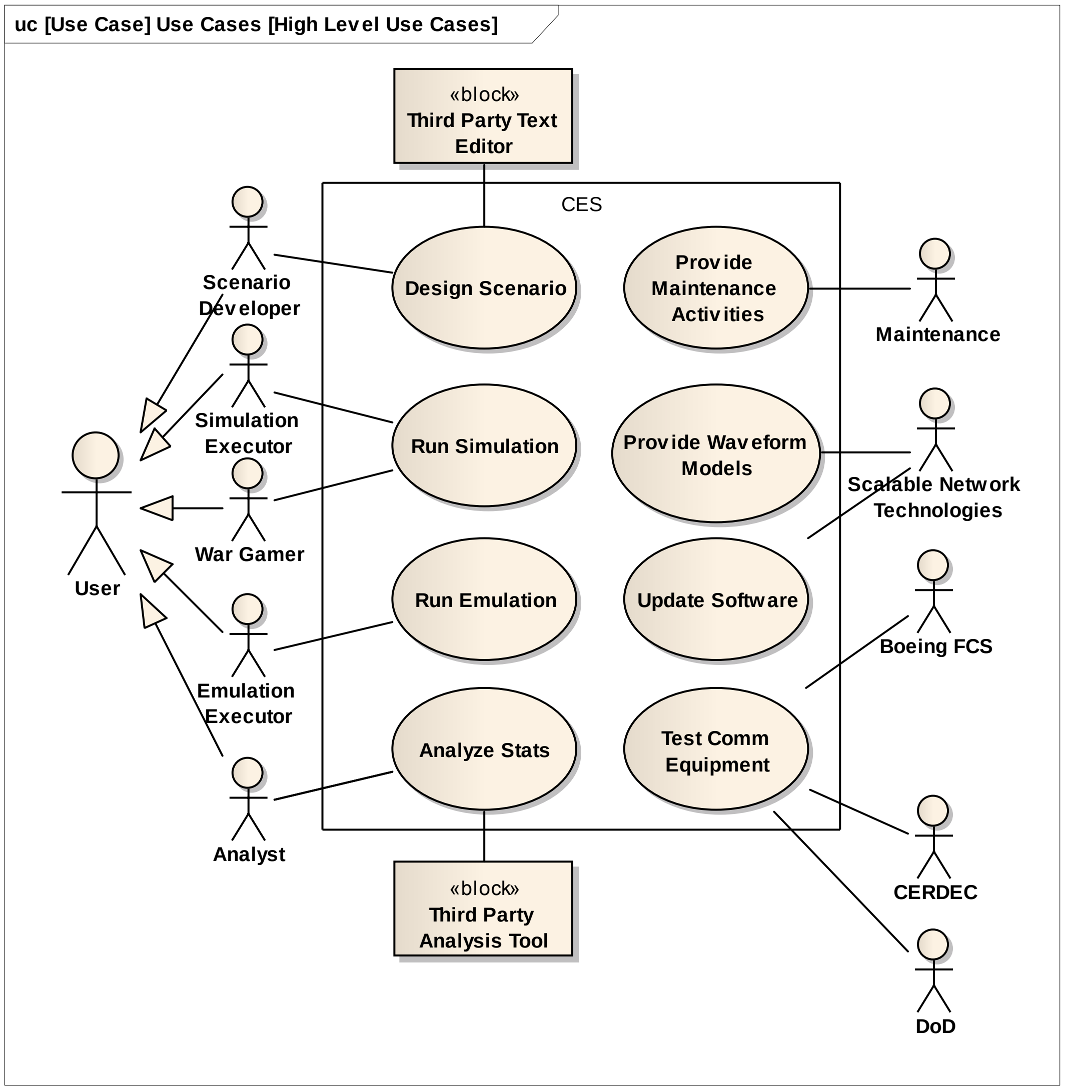


Figure 4: High Level Use Cases

In Figure 4, the major interactions of the various users are defined, as well as some additional stakeholder interactions representing significant impact on system operation. This diagram shows how the users of CES can be decomposed based on their primary interactions with the system, including designing scenarios, running simulations and emulations, as well as analyzing simulation and emulation statistics. Each of these interactions will be modeled at a lower level of abstraction in subsequent Use Case diagrams.

This diagram shows not only the subclasses of CES users and how they interact with the system, but also displays some important stakeholders who do not operate CES, but have significant impact on its functionality and capabilities. Among these types of stakeholders, Boeing FCS and SNT play the most important role, developing and updating the CES software, as well as providing CES with the protocol data for both commercial and military waveforms that enable CES's communications network analysis functionality. Maintenance personnel is responsible for providing maintenance activities, which in this context is defined as necessary software and hardware repairs, as well as assistance provided to CES users.

The final Use Case presented in Figure 4 reflects the primary goal of CES at its highest level, to provide a method for analyzing communication devices being developed by Boeing FCS (and other FCS suppliers), CERDEC and other DoD entities.

The major Use Cases along the left hand side of Figure 4, those describing direct interaction between the user and CES to accomplish its mission, will be further decomposed into more lower-level Use Cases. Figure 5 represents the sub-Use Cases associated with Designing a Scenario. These include functionality such as managing the .config file, importing terrain, setting parameters, placing nodes (network objects) and setting movement profiles, all major contributors to the development of scenarios.

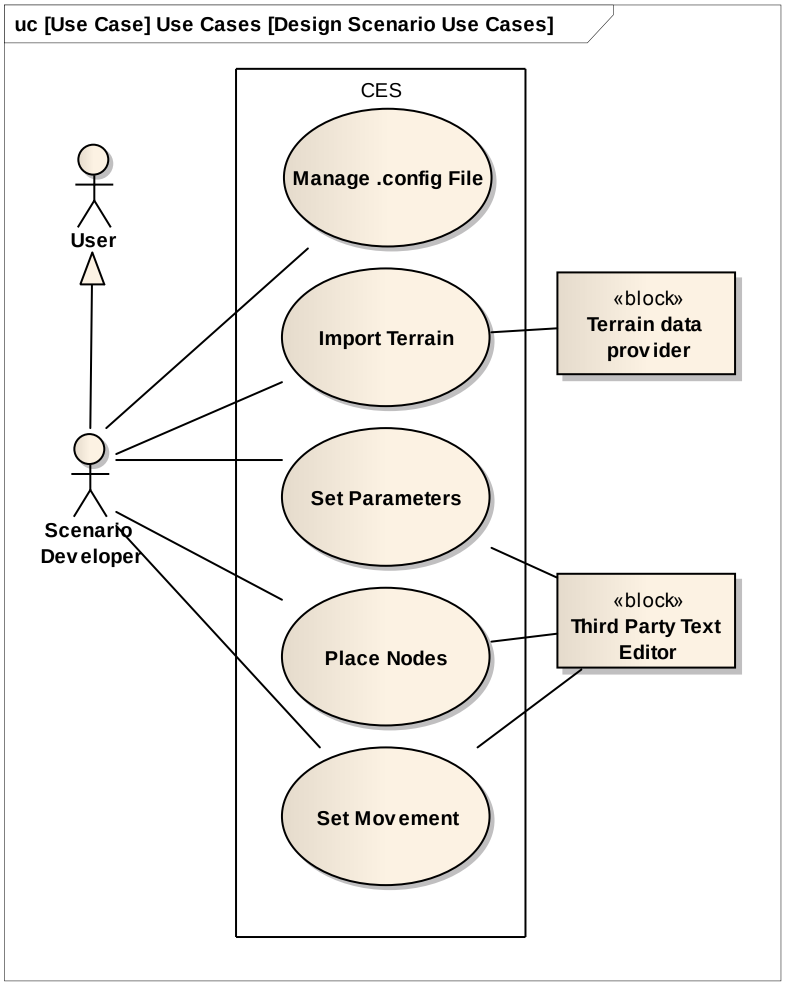
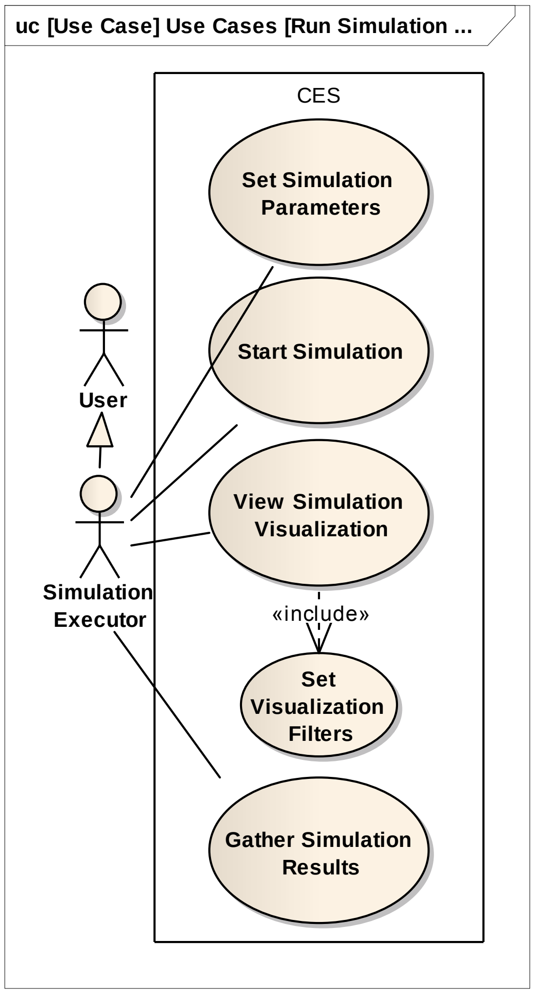

uc [Use Case] Use Cases [Design Scenario Use Cases]
CES
User
Scenario Developer
Manage .config File
Import Terrain
Set Parameters
Place Nodes
Set Movement
«block» Terrain data provider
«block» Third Party Text Editor

Figure 5: Design Scenario Use Cases

Here, we see the Use Case diagram also includes external systems that may be used to enable the user to make use of CES capabilities. While these lower level Use Case diagrams may seem rather simplistic, they are important in defining the major functionality required of CES. Any capabilities involved in designing scenarios should be decomposable from this set of Use Cases.

The same applies to Figure 6 and Figure 7, which lay out Use Cases associated with Running Simulations and Emulations.


uc [Use Case] Use Cases [Run Simulation ...
CES
Set Simulation Parameters
Start Simulation
View Simulation Visualization
«include»
Set Visualization Filters
Gather Simulation Results
User
Simulation Executor

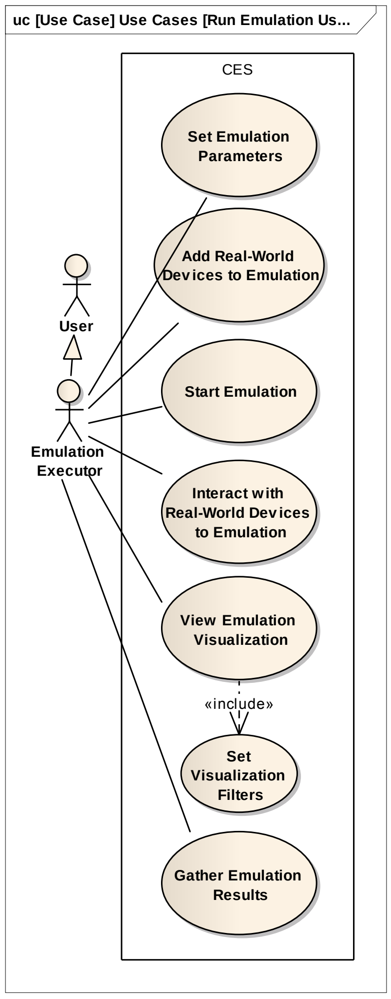

uc [Use Case] Use Cases [Run Emulation Us...
CES
Set Emulation Parameters
Add Real-World Devices to Emulation
Start Emulation
Interact with Real-World Devices to Emulation
View Emulation Visualization
«include»
Set Visualization Filters
Gather Emulation Results
User
Emulation Executor

Figure 6: Run Simulation Use Cases

Figure 7: Run Emulation Use Cases

These Use Case diagrams are similar in that that both require many of the same functions to operate. Setting filters, parameters, viewing visualization gathering results and starting simulation are emulation are basic capabilities required by the user to meet their goals with CES. The only difference is the adding of real-world communications devices to emulation, which allows the user to incorporate hardware and software in the field with virtual communications devices to provide integration analysis.

After running a simulation or emulation, the final Use Case diagram, Figure 8 represents the functions associated with Analysis of CES resulting statistics.

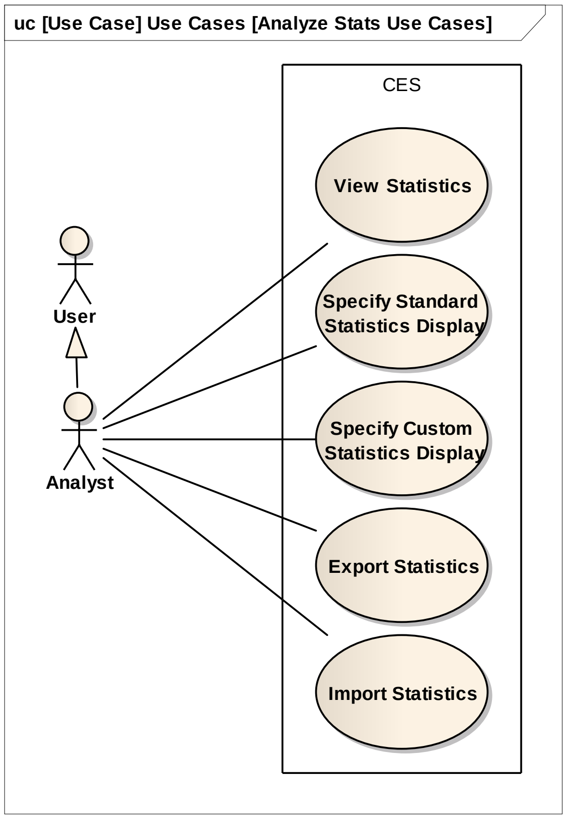

uc [Use Case] Use Cases [Analyze Stats Use Cases]
CES
View Statistics
Specify Standard Statistics Display
Specify Custom Statistics Display
Export Statistics
Import Statistics
User
Analyst

Figure 8: Analyze Statistics Use Cases

3.3 Process View

3.3.1 Process View Description

Drawing from Kruchten's 4+1 View Model [1995], "software is partitioned into a set of independent tasks. A task is a separate thread of control." Kruchten's Process View is specific to software. From the viewpoint of a system, leveraging model based systems engineering and SysML, the Process View of this ADD will present Activity Diagrams to represent a translation of Use Cases to activities taking place with and around CES. Activity diagrams consist of activities strung together to model the behavior of a system and its stakeholders. Often, swimlanes are used to show which stakeholder, system, or system component are responsible for the activities.

3.3.2 Process View Diagrams and Narrative

To develop the process View diagrams, the highest level of activities was modeled first. This High Level Activity Diagram can be seen in Figure 9.

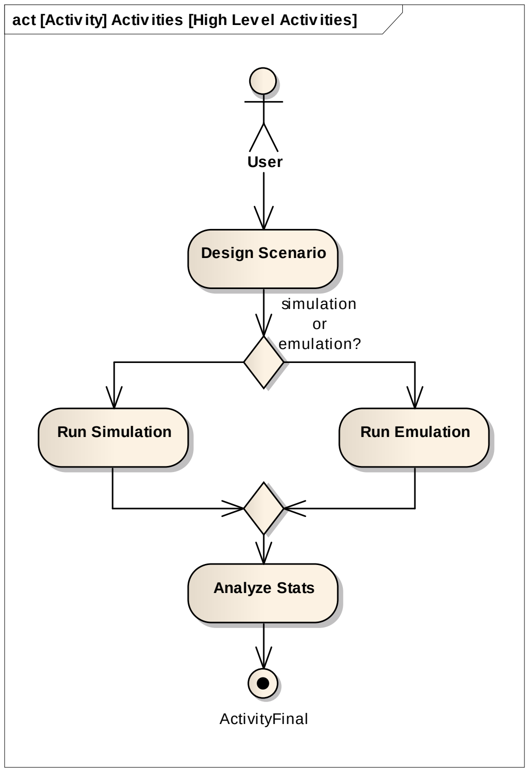


Figure 9: High Level Activity Diagram

These activities represent the most basic interactions of a user with CES. Specific activities that fall within these will be discussed below, as each of these high level activities will be decomposed into additional activity diagrams.

3.3.2.1 Design Scenario Activities

In Figure 10, the activities involved in Designing a Scenario are laid out. At the onset of a scenario design, the Scenario Designer (abstracted to User for this activity diagram) must choose which tool they will utilize for design, either CES or a third party text editor. If CES is to be used for design, there is an additional choice of using the Architect or the File Editor. Following each of these choices, the User takes an action, which is accompanied by an activity within the swim lane of the appropriate tool. In this diagram, the activities taking place within the File and Text Editor are abstracted to Edit .config File text, since they both involve reading, writing, and editing a large about of coded text. Each of these files is then saved, creating or updating a .config file that will be read by the Architect, or an external tool for simulation or emulation execution.

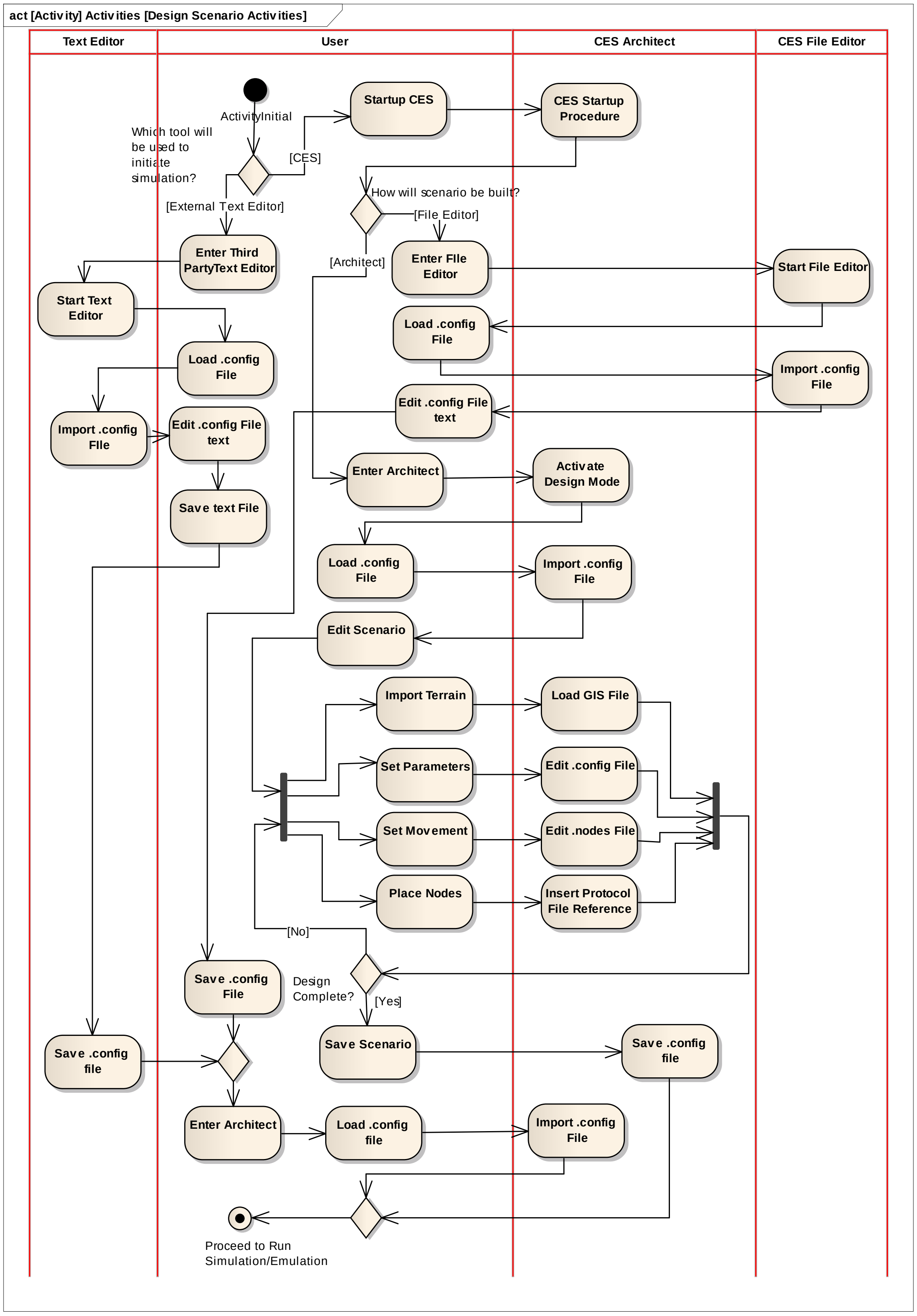


Figure 10: Design Scenario Activities

If the Architect is chosen to design the scenario, the Design Mode is activated, and the major activities carried out by the User include Importing Terrain, Setting Parameters and Movement and Placing Nodes. These activities may happen in a different order every time a scenario is designed, therefore they all stem from a Fork object, indicating an OR flow. Since it is likely that design will be an iterative process, a Decision object follows these activities, representing progression of the Process if the design is complete, or a loop if it is not complete.

Figure 10 ends with a Final Activity Node that is labeled "Proceed to Run Simulation/Emulation" since presumably this sis the next set of activities that the user will carry out. This node leads directly to the Initial Activity node of Figure 11 and Figure 13, labeled "From Design Scenario" to establish continuity of Process.

3.3.2.2 Run Simulation Activities

As in the Design Scenario activities, the initial path in the Run Simulation diagram, Figure 11, involves tool choice. Since CES is compatible with a number of simulation tools, namely through the HLA, DIS and Battlespace Simulation System interfaces to be discussed in Section 3.4, the User has a choice of whether to run simulations natively or using an external tool. External tools are accessed through the Command Line Prompt, which is used to run simulation software and load the .config file for simulation tool processing. The end result of using an external simulation tool will be the creation of a statistics file, which will allow the User to interpret and analyze simulation data, a process that will be described in Figure 15.

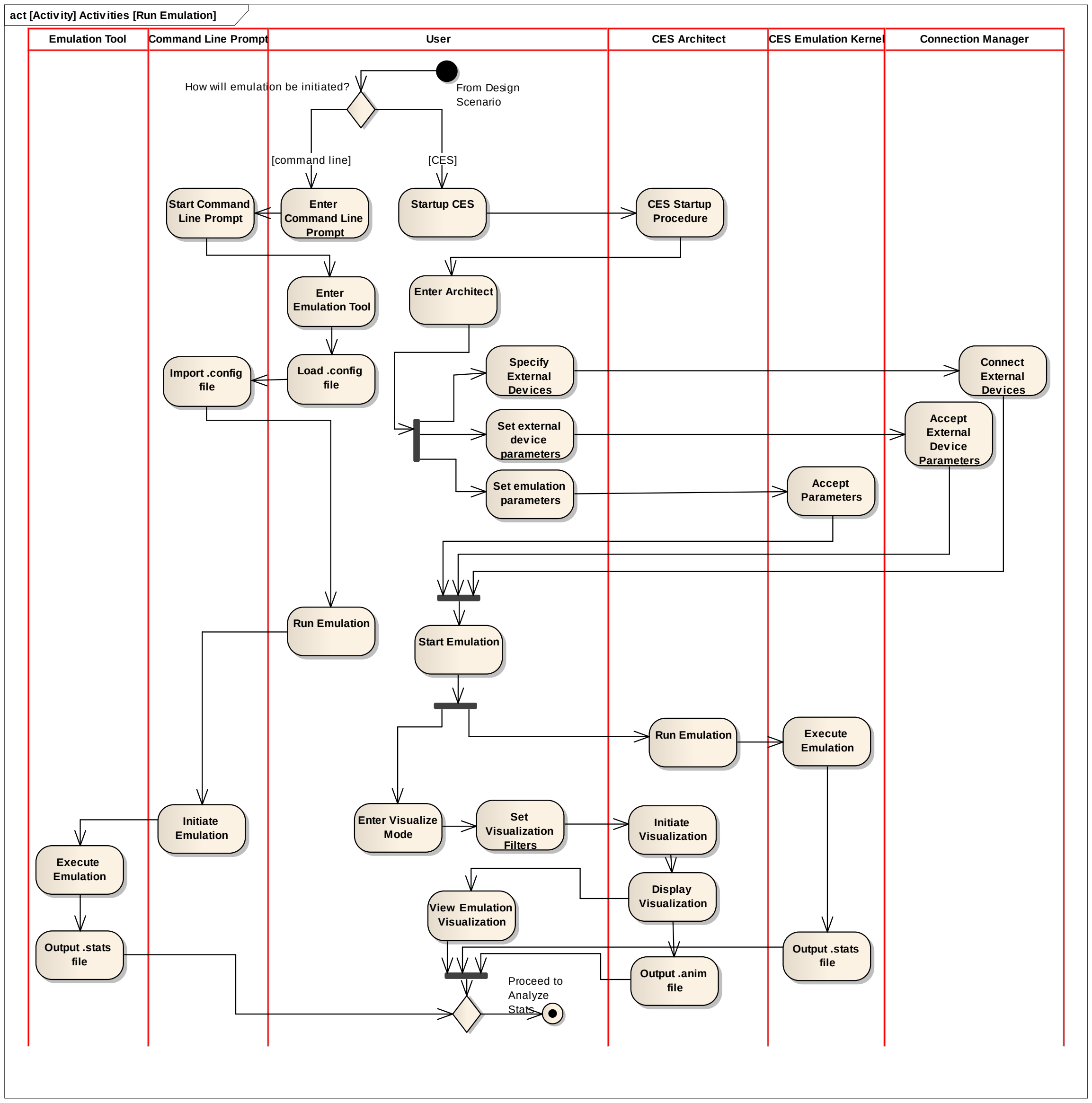


Figure 11: Simulation Activities

If the User chooses to use CES to run the simulation, he must enter the Architect to set simulation parameters, and start the simulation. Once the simulation has been started, a fork shows that Visualizing the Simulation, Executing the Simulation, and Interfacing with a Battle Simulation System (to provide communications effect and receive communications data from war games) will happen concurrently in the Architect and Simulation Kernel respectively. The output of the Simulation Execution is a .stats file, for analysis by the User, while the Visualization output is a movie and accompanying .anim file, for playback in an external visualization tool.

It is important to discuss a matter of semantics here. After the Simulation kernel has Accepted Parameters, there is a series of 3 activities that may sound similar: Start, Run and Execute Simulation. Since a different actor/component is performing each of these activities, the names are slightly different as the actions are slightly different. The User Starts the Simulation by clicking a button and sending a command to the Architect. The Architect Initiates the Simulation by passing along a command to the Simulation Kernel, which Executes the Simulation. This pattern of terminology is evident throughout the activity diagrams in this section. Of the three activities related to running a simulation, the first two (Start and Initiates) are relatively simple, taking a stimulus and translating it to a command. The final (Execute) is much more complex, as this activity is responsible for the actual carrying out of a simulation. To explore this complexity, an activity diagram has been developed to show the next level of detail for Executing a Simulation, specifically the details of how the Simulation Kernel uses Event Handling to execute a scenario. An additional activity of interest is the querying of the model libraries for protocol data by the Simulation Kernel. This activity translates the protocols referenced in the .config file to a format usable by the Simulation Kernel. Model Libraries will be discussed further in Section 3.4.2.3.

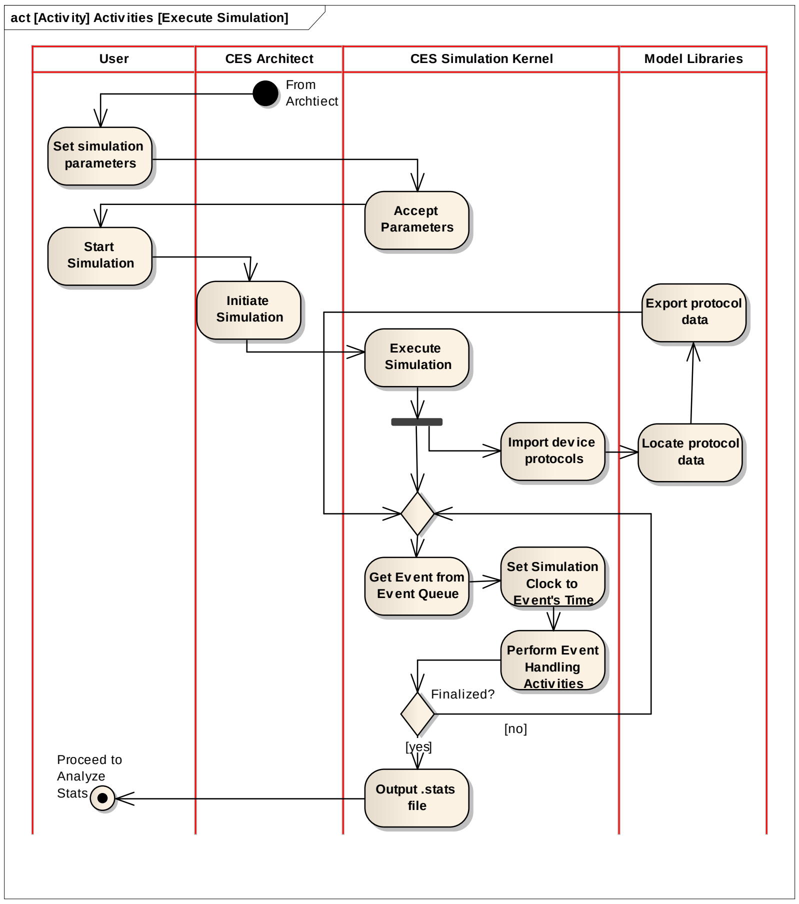


Figure 12: Execute Simulation Event Handling Details

3.3.2.3 Run Emulation Activities

The next diagram of the Process view explores the activities related to Emulation Execution. Like simulation, emulation can be executed with the Architect/Emulation Kernel combination or an external tool. Many of the activities taking place during emulation are the same as those of simulation, with the exception being the inclusion of external communications hardware and software for real time communication device analysis.

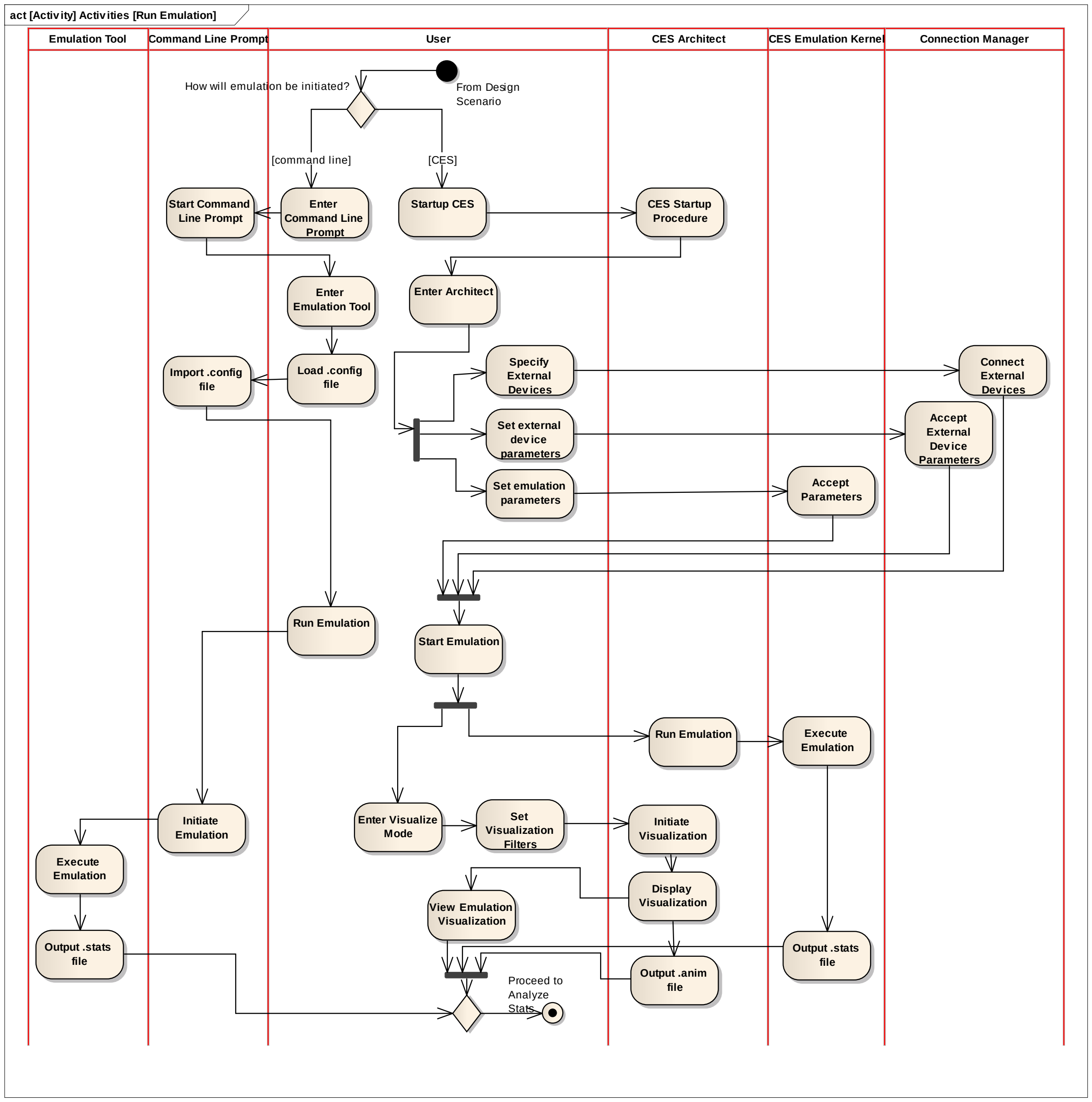


Figure 13: Emulation Activities

Once the User has chosen to use the Architect to facilitate emulation, external devices can be connected to CES, and their parameters set, through the Connection Manager Interface, discussed in Section 3.4.2.2. As seen below in Figure 14, the Connection Manager is also responsible for transmitting parameters, communications effects, packets and performance data to and from external devices during emulation execution. The remainder of Figure 14 shows details associated with the activities taking place during Emulation Execution

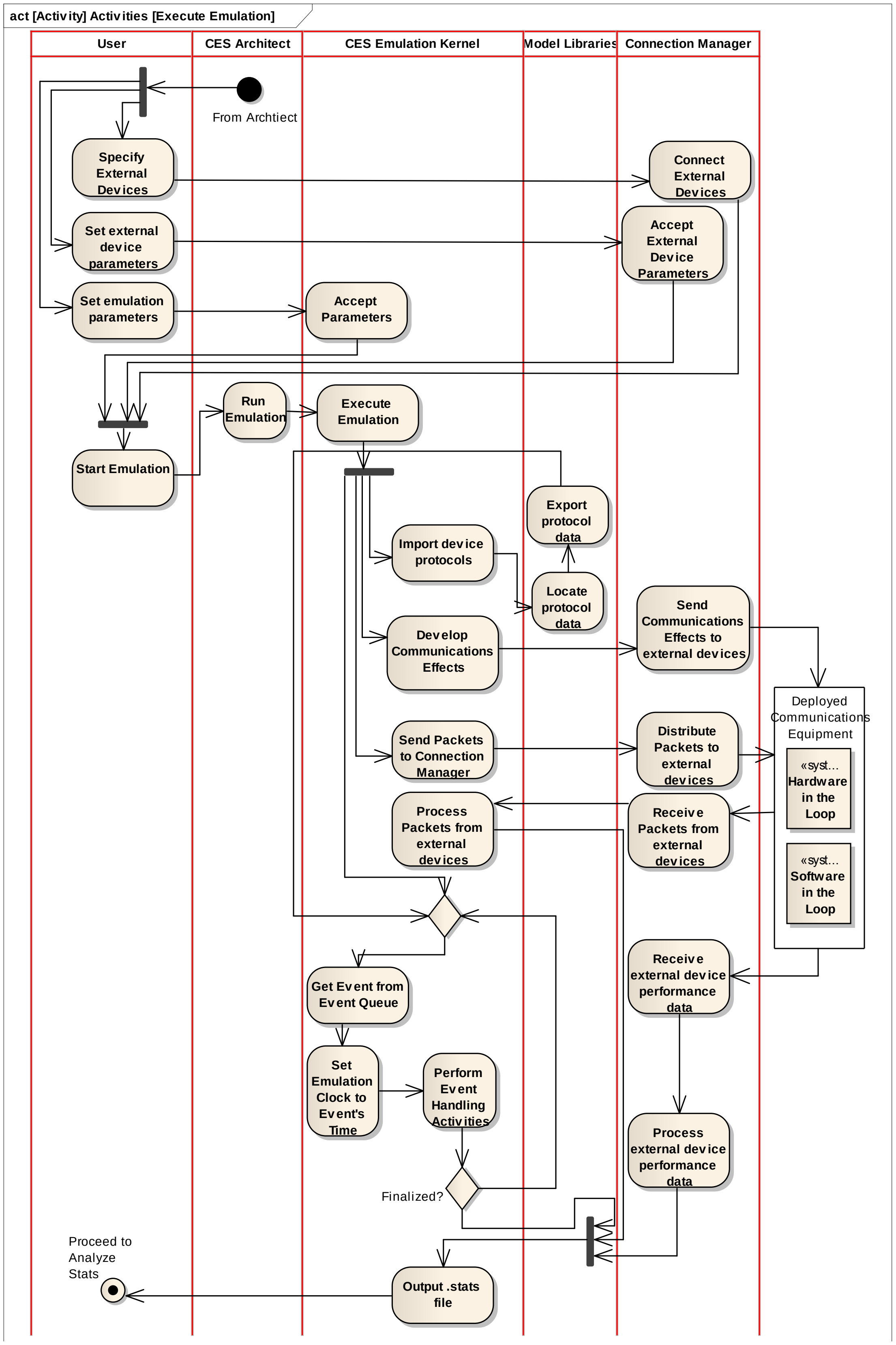

act [Activity] Activities [Execute Emulation]
User
CES Architect
CES Emulation Kernel
Model Libraries
Connection Manager
From Archtiect
Specify External Devices
Set external device parameters
Set emulation parameters
Connect External Devices
Accept External Device Parameters
Accept Parameters
Run Emulation
Execute Emulation
Start Emulation
Export protocol data
Import device protocols
Locate protocol data
Develop Communications Effects
Send Communications Effects to external devices
Deployed Communications Equipment
«syst... Hardware in the Loop
«syst... Software in the Loop
Send Packets to Connection Manager
Distribute Packets to external devices
Process Packets from external devices
Receive Packets from external devices
Receive external device performance data
Get Event from Event Queue
Set Emulation Clock to Event's Time
Perform Event Handling Activities
Process external device performance data
Finalized?
Proceed to Analyze Stats
Output .stats file

Figure 14: Execute Emulation Detailed View

3.3.2.4 Analysis Activities

The final activity diagram describing the Process View deals with analysis of simulation and emulation results. Like the other major activity diagrams, Figure 15 starts off showing the User's choice between integrated CES tool (Analyzer) and an external analysis tool. Important activities of Figure 15 include those related to specifying custom or standard statistics displays, importing .stats file for analysis and creation of a custom .stats file for export

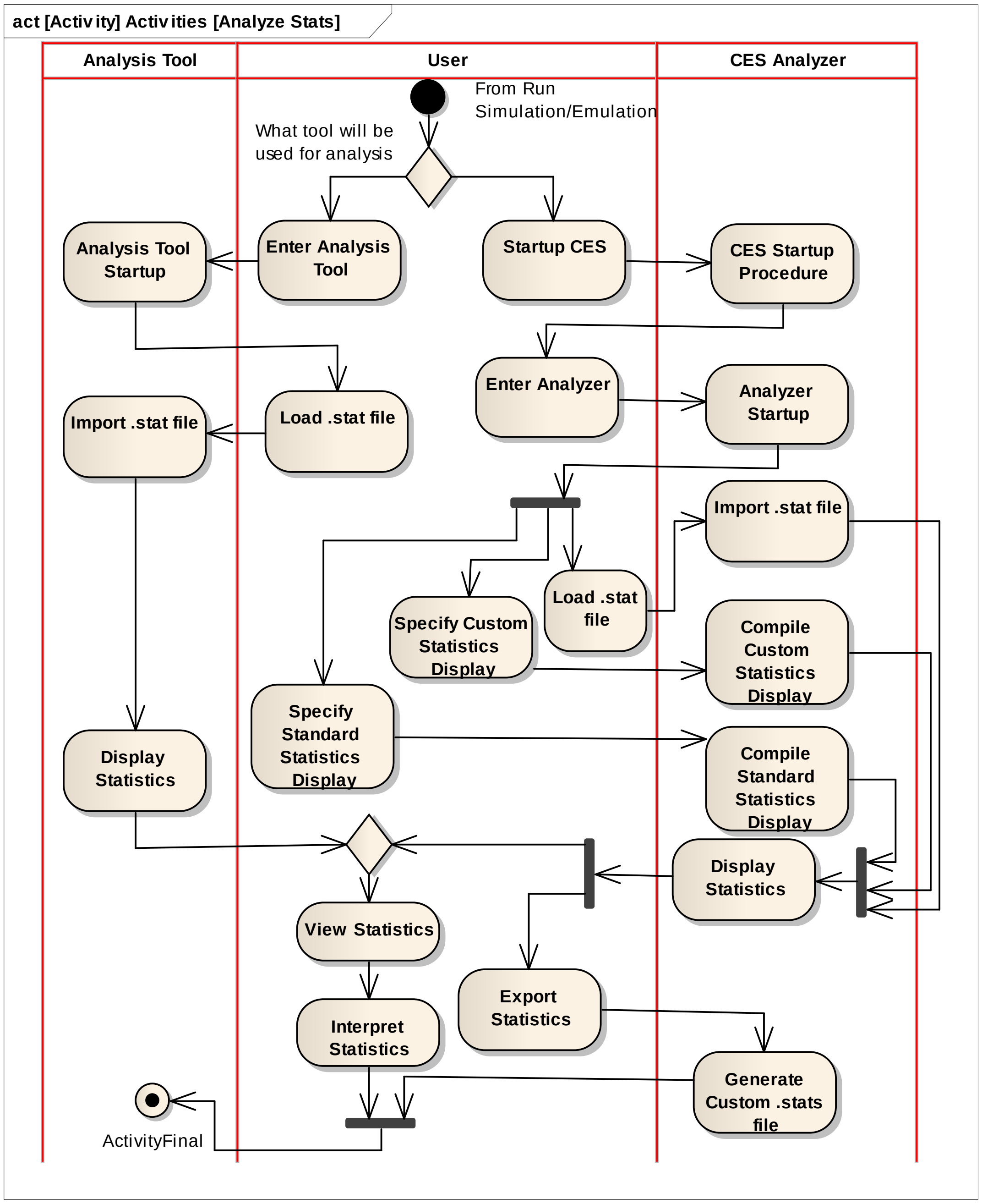


Figure 15: Analyze Stats Activities

3.4 Logical Architecture View

3.4.1 Logical Architecture View Description

Defining a Logical Architecture involves the decomposition of a system architecture into its logical components. This view requires that individual system components be abstracted to logical components that perform the system functionality without imposing implementation constraints. The logical architecture view is useful because its acts as an intermediate step between system requirements and instantiated physical architectures. This enables an organization to reduce risk when designing a system; requirements and technology will frequently change, but unless a full scale architecture redesign is required to foster these changes the logical architecture will stay relatively the same. The logical architecture of an existing system such as CES can be difficult to establish, since reverse engineering efforts will typically lead to the instantiated physical architecture. Abstracting this physical architecture will lead to a higher level of understanding of the core elements of a system.

3.4.2 Logical Architecture View Diagrams and Narrative

The first step to developing the logical architecture of an existing system is to decompose the system into its parts. The resulting decomposition for CES, seen in Figure 16, is a hybrid of physical and logical architecture views. For example, the Emulation Kernel block is an example of a logical component, while EXata, the actual software used to run emulations, is an instantiation of a logical component. While this diagram does not define a logical architecture, it is necessary to develop so that a proper understanding of the system is established.

3.4.2.1 High Level Decomposition Diagram

Figure 16 is made of a number of blocks, each representing either a component or external interacting system of CES. The dotted line marks the boundary between CES and external systems. This boundary allows the viewer to distinguish between elements that are central to this modeling effort, and those that are outside the scope of this ADD. Systems outside the dotted line will be represented in this ADD as simple black boxes.

Figure 16 is a simplistic view of CES, and is elaborated further in the diagrams below.

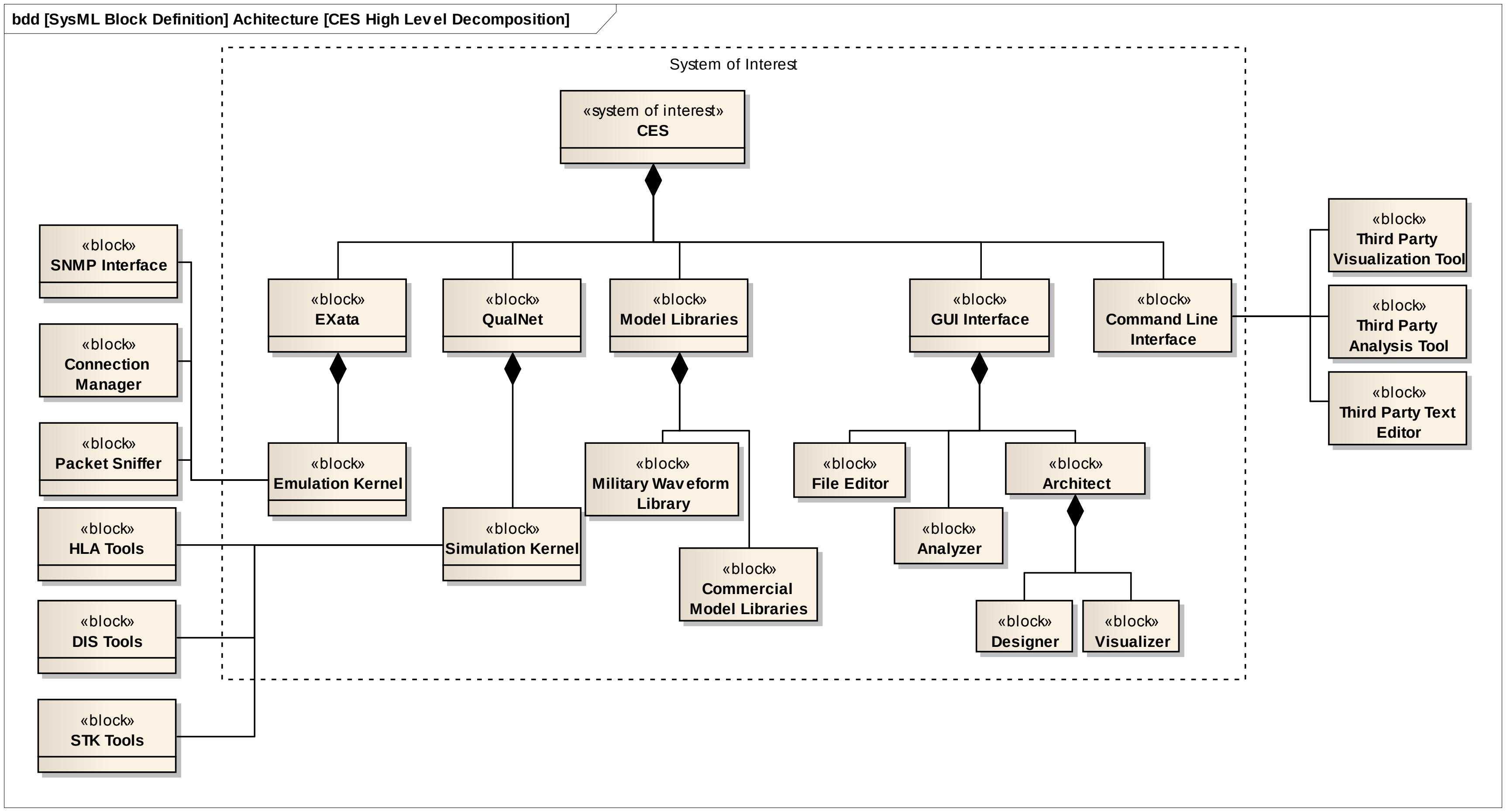

bdd [SysML Block Definition] Achitecture [CES High Level Decomposition]
System of Interest
«system of interest» CES
«block» EXata
«block» QualNet
«block» Model Libraries
«block» GUI Interface
«block» Command Line Interface
«block» SNMP Interface
«block» Connection Manager
«block» Packet Sniffer
«block» Emulation Kernel
«block» HLA Tools
«block» Simulation Kernel
«block» DIS Tools
«block» STK Tools
«block» Military Waveform Library
«block» Commercial Model Libraries
«block» File Editor
«block» Analyzer
«block» Architect
«block» Designer
«block» Visualizer
«block» Third Party Visualization Tool
«block» Third Party Analysis Tool
«block» Third Party Text Editor

Figure 16: CES High Level Decomposition

3.4.2.2 Modular Logical Architecture

Working from the High Level Decomposition Diagram, the next diagram represents a Logical Architecture separated into modules. This Logical Architecture was constructed by taking the major elements of CES, and trying to abstract them up one level to form logical components. Figure 17 contains elements such as Kernels, File Managers, Interfaces and Libraries. Modularity was established by combining elements with similar functionality and responsibilities into higher level modules. It was determined that there were 5 major high level modules:

- External Interfaces – Modeling all systems that interface with CES to accomplish its goals. Interfaces will be discussed in Sections 3.4.2.2 and 3.4.2.3.
- IO Manager – Lays out the input and output files used and created by CES. IO components will be discussed in Section 3.4.2.2.
- GUI – These components represent the tools built within CES enabling a user to create scenarios and analyze statistics resulting from emulation and simulation, and will be discussed in Section 3.4.2.3.
- Model Libraries – This is the location where protocol and waveform data is stored for use by CES simulation and emulation engines and will be discussed in Section 3.4.2.3.
- Execution Kernels – This module is where the "heavy lifting" happens, where data from other CES components and interfacing tools are processed in simulations and emulations. Components within this module are elaborated on in Section 3.4.2.3.

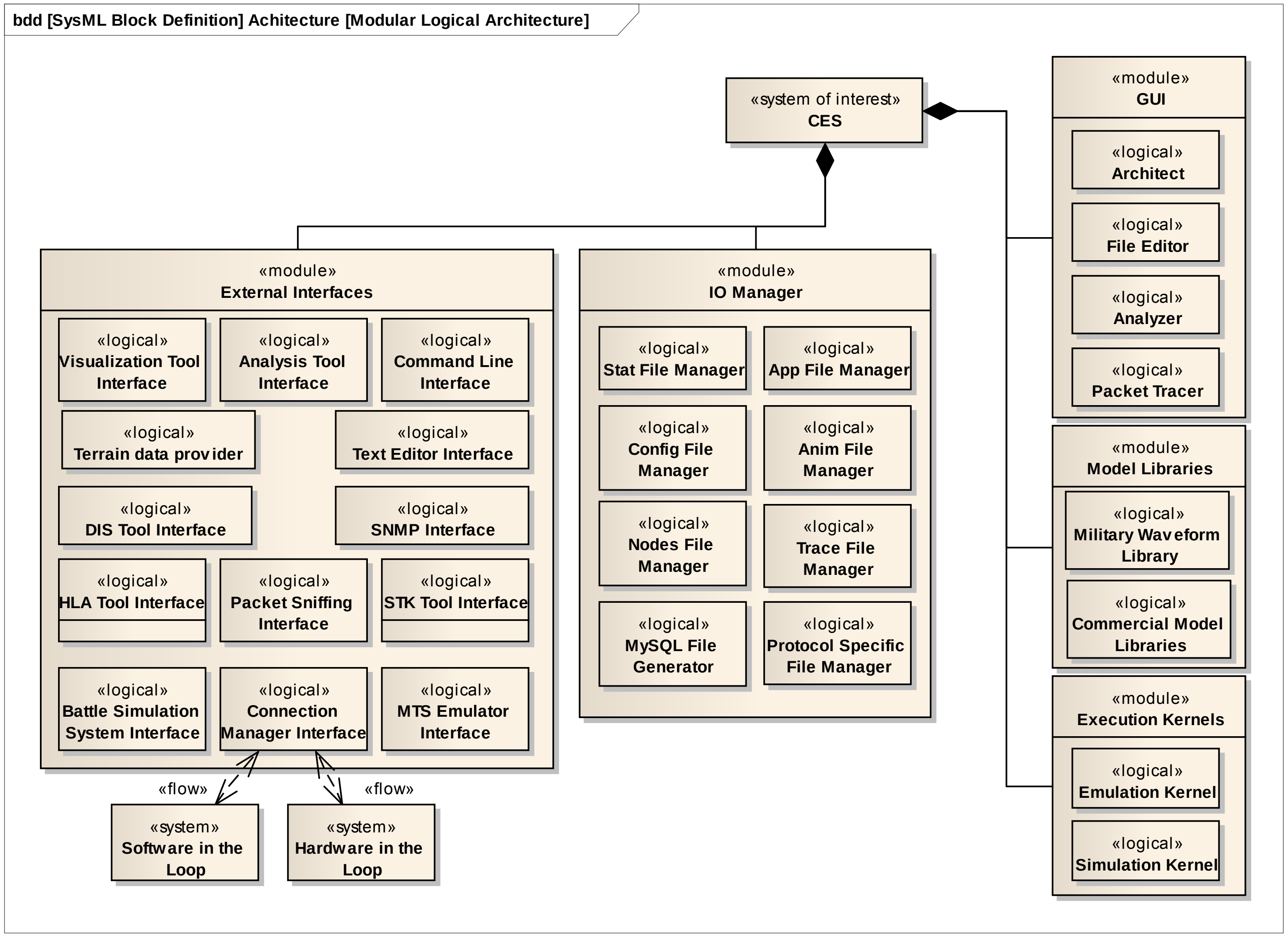

bdd [SysML Block Definition] Achitecture [Modular Logical Architecture]
«system of interest» CES
«module» External Interfaces
«logical» Visualization Tool Interface
«logical» Analysis Tool Interface
«logical» Command Line Interface
«logical» Terrain data provider
«logical» Text Editor Interface
«logical» DIS Tool Interface
«logical» SNMP Interface
«logical» HLA Tool Interface
«logical» Packet Sniffing Interface
«logical» STK Tool Interface
«logical» Battle Simulation System Interface
«logical» Connection Manager Interface
«logical» MTS Emulator Interface
«flow»
«flow»
«system» Software in the Loop
«system» Hardware in the Loop
«module» IO Manager
«logical» Stat File Manager
«logical» App File Manager
«logical» Config File Manager
«logical» Anim File Manager
«logical» Nodes File Manager
«logical» Trace File Manager
«logical» MySQL File Generator
«logical» Protocol Specific File Manager
«module» GUI
«logical» Architect
«logical» File Editor
«logical» Analyzer
«logical» Packet Tracer
«module» Model Libraries
«logical» Military Waveform Library
«logical» Commercial Model Libraries
«module» Execution Kernels
«logical» Emulation Kernel
«logical» Simulation Kernel

Figure 17: Modular Logical Architecture

In Figure 17, all tools that interact with CES are abstracted to simple interfaces. An important interface listed here is with the Connection Manager. This interface is what enables Hardware and Software that is deployed in a network to communicate with CES for running emulations. This provides a major functionality for CES, as it is the point of integration from real world and virtual world components, and allows it to act as a true test bed for studying the effects of adding new technologies and equipment to existing communications networks.

Two other important interfaces are the Battle Simulation System Interface and the Message Transceiver System (MTS) Emulator. The Battle Simulation System is essential when CES is acting in server mode, as a network communications simulation transaction server. The Battle Simulation System sends communication-based messages to the CES server, which in turn responds with an outcome of the message delivery within the simulated environment. This is important when providing simulation event processing within the FCS system, and for integration of CES with war games and other real time simulations outside of CERDEC. The MTS Emulator is a stand-alone tool which communicates with the CES server to send and receive messages to and from the CES server. These two components are especially important when interfacing external tools, software and hardware in using the CES server for distributed modeling and simulation of communications networks for simulation and emulation.

Additional interfaces include all of the external tools that can interact with the CES. This includes third party tools such as Visualization, Analysis and Text Editor Interfaces. These specific interfaces are not physical, meaning that there is no plug or coding developed to directly interact with these tools. However, through creating output files, CES is able to utilize these non-specified tools, and therefore it is important to make note of this in fully describing the functionality and capabilities of CES.

This discussion of output files leads to the development of the IO manager module. This module contains all components that deal with creating, modifying and managing the various input and output file formats associated with CES. This module includes components from processing the following eight file types:

Inputs:

- Config Files (.config) – This file is constructed using the Architect, File Editor or third party Text Editor. The .config file acts as the backbone for CES simulations and emulations. It contains information regarding scenario properties, specific references to the properties of various communications devices to be modeled, as well as the parameters for simulation and emulation. Config files can be created both internally using the Architect of the File Editor, as well as externally using a third party Text Editor.
- Nodes Files (.nodes) – This file, which is referenced within the .config file, describes the initial (and/or final) positions of nodes within each scenario, which is especially useful when the terrain surrounding and movement of communications devices is an important feature of simulation and emulation within scenarios.

- App Files (.app) – This file, which is referenced within the .config file, describes the applications that are running on the nodes defined in the .nodes file.
- Protocol Specific Files – This type of file is generic in so far as it is specific to certain types of protocols that may be referenced in the .config file. It provides the emulation and simulation engines properties and parameters associated with protocols used in a scenario.

Outputs:

- Stat Files (.stat) – This file is the primary output of CES and contains the results of simulations and emulations carried out by CES. It is used to show simulation and emulation statistics both internally with the Analyzer, and also externally in a third party Analysis Tool.
- Anim Files (.anim) – This file records the animations carried out by the Execution Kernels for visualizing scenario simulation and emulation. It is used to show simulation and emulation animations both internally with the Architect, and also externally in a third party Visualization Tool.
- Trace Files (.trace) – This file is generated by the Packet Tracer and records information and statistics related to packets as they transverse the protocol stack at each node from source to destination.
- MySQL Files – When reporting the results of simulation and emulation, CES automatically produces the .stat file. However, if enabled, CES may also report results in a database file, using the MySQL relational database management system.

3.4.2.3 Logical Architecture

This alternate format for the logical architecture describes the architecture of CES and associated components in a more linear format, while also including the interactions of users directly involved in CES operation. The alternate Logical Architecture diagram can be seen in Figure 18.

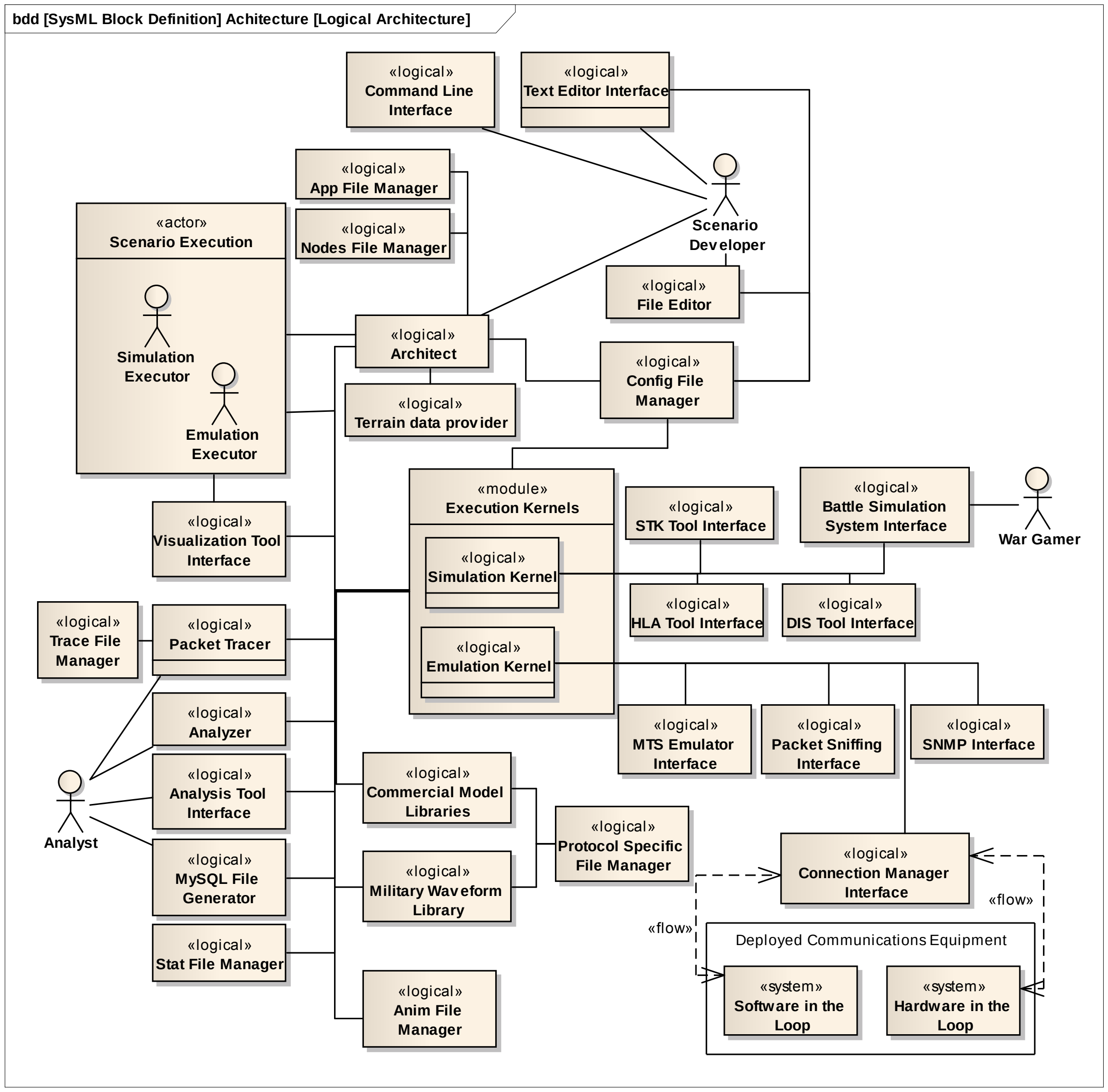

bdd [SysML Block Definition] Achitecture [Logical Architecture]
«logical» Command Line Interface
«logical» Text Editor Interface
«logical» App File Manager
«logical» Nodes File Manager
«actor» Scenario Execution
Simulation Executor
Emulation Executor
Scenario Developer
«logical» File Editor
«logical» Architect
«logical» Terrain data provider
«logical» Config File Manager
«module» Execution Kernels
«logical» Simulation Kernel
«logical» Emulation Kernel
«logical» STK Tool Interface
«logical» Battle Simulation System Interface
War Gamer
«logical» Visualization Tool Interface
«logical» HLA Tool Interface
«logical» DIS Tool Interface
«logical» Trace File Manager
«logical» Packet Tracer
«logical» Analyzer
«logical» MTS Emulator Interface
«logical» Packet Sniffing Interface
«logical» SNMP Interface
«logical» Analysis Tool Interface
«logical» Commercial Model Libraries
«logical» Protocol Specific File Manager
Analyst
«logical» MySQL File Generator
«logical» Military Waveform Library
«logical» Connection Manager Interface
«flow»
«flow»
Deployed Communications Equipment
«logical» Stat File Manager
«logical» Anim File Manager
«system» Software in the Loop
«system» Hardware in the Loop

Figure 18: Logical Architecture

Elements included in the Logical Architecture diagram that were not described in Figure 18 are discussed below:

Interfaces

- Command Line Interface – The command line interface allow users to interact with CES via a Windows DOS prompt or UNIX command window. Command line operation uses text files as input, which can be created using a text editor, without needing the CES Architect. Additionally, using the command line allows a user to be flexible with regards to choice of visualization and analysis tools.
- Terrain Data Provider – The terrain data provider is a third party source of geographical information, most likely in a GIS file format. This information is used to insert elements of elevation, shape layers, coordinate system descriptions and other information related to the

environment in which a scenario is being run, to model communication effects caused by different geographical features.

HLA Tool Interface – High Level Architecture– HLA Tool Interface is a specification used to enable inter-operation between multiple simulation software systems. This interface enables CES to interact with other HLA compatible software programs. Of particular interest in the research that surrounds this ADD, the HLA Interface allows CES to "play" with OneSAF Testbed Baseline (OTB), another Army tool that is being utilized in a parallel SERC research effort. This interface is likely to be utilize extensively in the possible expansions of CES discussed in Section 5.

- DIS Tool Interface – Distributed Interactive Simulation – DIS is an IEEE standard for interfacing multiple simulation tools into a single, real-time exercise. DIS has been replaced by HLA but remains in use today by many simulation software tools, including OTB mentioned above.
- SNMP Tool Interface – Simple Network Management Protocol – EXata implements an SNMP agent on each emulated node. This interface communicates with SNMP agents to manage nodes in an emulated network. As is presented in Figure 3, the SNMP Interface pulls device parameters and network data from a management information base and can send traps to management systems such as HP OpenView, IBM Tivoli or SolarWinds Orion.
- Packet Sniffing Interface – The Packet Sniffing Interface provides CES with the capability of making the traffic inside a scenario visible to external packet sniffing software. This software examines packets transmitted and received within emulated scenarios. This interface can link CES to third party software such as Wireshark, Observer or Microsoft Network Monitor.
- STK Tool Interface – Satellite Tool Kit – STK is a physics-based software package that allows engineers to perform complex analysis of land, sea air and space assets. It can be used with CES to handle communications analysis and battlespace management of C4ISR components. This interface allows CES to interoperate with STK when running simulations.

GUI Components

- Architect – The Architect represents the main network design and visualization components within CES. The Architect runs into two modes to provide two sets of capabilities. Design mode is where a user can utilize a drag and drop interface to specify terrain, network nodes and connections, protocol stacks, application layer traffics and associated parameters to develop a scenario. Visualize mode allows a user to observe simulations based on designed scenarios. Information including packet flow, critical performance metrics, real-time statistics and what-if analysis can be viewed in the Architect's Visualize mode.
- File Editor – The File Editor is a text editing tool that allows the user to create and modify files associates with CES operations, most notably the .config files.

- Analyzer – The Analyzer is an internal statistical analysis tool used to display the data collected during simulation and emulation. The Analyzer can be used to customize graphical display of results and can present multiple simulation and emulation results at once.
- Packet Tracer – The Packet Tracer is the CES internal component that tracks packets as they transverse the network in simulation and emulation. The Tracer is internal to CES as the Packet Sniffing Tool is external to CES.

Model Libraries

- Commercial Model Libraries – The model library is a repository for the waveforms, protocols and parameters of commercial network components. These components are referenced within the .config file, and the execution kernels draw information about the components in a scenario from this repository during simulation and emulation. These libraries are available to all users of CES, both commercial and military. These libraries may be created and compiled by any number of communication device developers.
- Military Waveform Library – The Military Waveform Library is synonymous to the Commercial Model Libraries but contains sensitive data specific to military communications devices. These libraries are available only to authorized military personnel or contractors. The specific waveforms are developed by Boeing FCS, or their subcontractors, and are described in libraries for CES by SNT.

Execution Kernels

- Simulation Kernel – The CES simulation Kernel is a parallel discrete-event scheduler. It provides the scalability and portability to run hundreds and thousands of nodes with high-fidelity models on a variety of platforms, from laptops and desktops to high performance computing systems. Users do not directly interact with the kernel, but use an API to develop protocol models. Although any simulation kernel with the correct interface can be plugged into CES, version 5.3.1 uses SNT's QualNet.
- Emulation Kernel - The CES emulation kernel is a high-fidelity, real-time interface to connect the modeled network with external real applications and hardware. The core of the emulation kernel is a real-time event scheduler that processes the internal events, as well as the events from external sources. It also provides a transparent interface to the real-world applications and hardware such that the latter can access any emulated network resource as if it exists physically. Users do not directly interact with the kernel, but use the Connection Manager or interfaces such as SNMP, Packet Sniffing interfaces to connect the real applications and hardware with the emulated network. Although any emulation kernel with the correct interface can be plugged into CES, version 5.3.1 uses SNT's EXata.

The diagram presented in Figure 18 is beneficial as it can be used to trace the activities taking place during CES operation as a combination of specific components and the flows between them. For example, in Figure 19, red lines indicate which components would be used in Designing a Scenario using the Architect, Running a Simulation, using a third party Visualization Tools to view the simulation animation and Analyzing the results using a third party Analysis Tool.

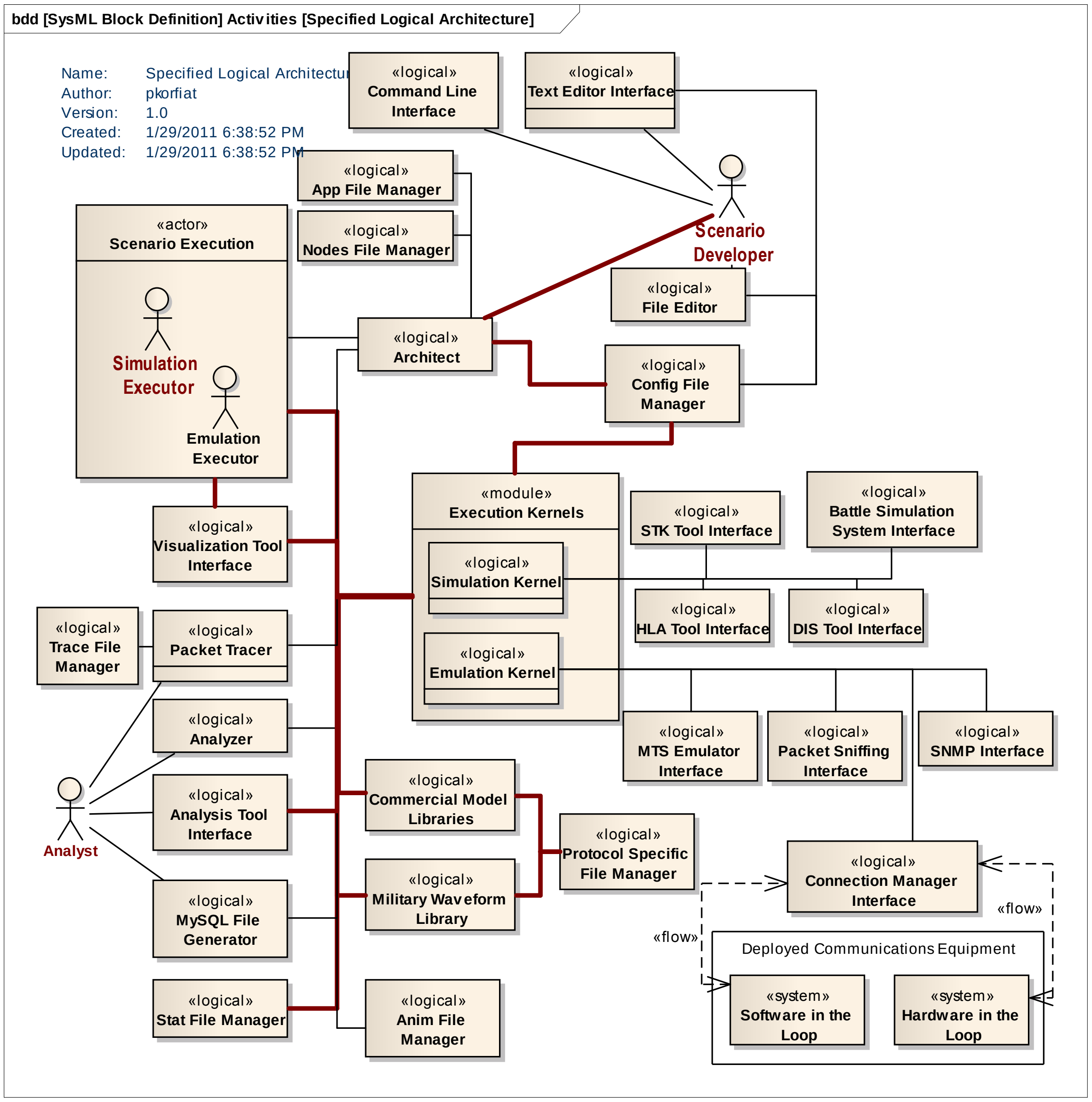

bdd [SysML Block Definition] Activities [Specified Logical Architecture]
Name: Specified Logical Architecture
Author: pkorfiat
Version: 1.0
Created: 1/29/2011 6:38:52 PM
Updated: 1/29/2011 6:38:52 PM
«logical» Command Line Interface
«logical» Text Editor Interface
«logical» App File Manager
«logical» Nodes File Manager
Scenario Developer
«actor» Scenario Execution
Simulation Executor
Emulation Executor
«logical» File Editor
«logical» Architect
«logical» Config File Manager
«logical» Visualization Tool Interface
«module» Execution Kernels
«logical» Simulation Kernel
«logical» Emulation Kernel
«logical» STK Tool Interface
«logical» Battle Simulation System Interface
«logical» HLA Tool Interface
«logical» DIS Tool Interface
«logical» Trace File Manager
«logical» Packet Tracer
«logical» Analyzer
«logical» Analysis Tool Interface
Analyst
«logical» MySQL File Generator
«logical» Stat File Manager
«logical» Commercial Model Libraries
«logical» Military Waveform Library
«logical» Anim File Manager
«logical» Protocol Specific File Manager
«logical» MTS Emulator Interface
«logical» Packet Sniffing Interface
«logical» SNMP Interface
«logical» Connection Manager Interface
«flow»
«flow»
Deployed Communications Equipment
«system» Software in the Loop
«system» Hardware in the Loop

Figure 19: Activity Traced Logical Architecture

This type of diagram provides value to modelers and other relevant stakeholders and development personnel for a number of reasons including:

- Confirms decisions made in modeling Activity Diagrams in the Process View, Section 3.3

- Allows modeler to make sure all internal and external components of the system are accounted for.
- Points out the most crucial aspects of design. For example, the diagram shows that the Execution Kernels will be the main component of CES, as most of the other components and related data flows stem from it.
- Provides guidance for where interfaces need to be, and which elements they need to connect. For example, in the Modular Logical Architecture view presented in Figure 17, all the viewer learns is that the HLA Tool Interface forms a link between an external HLA Tool and some unknown part of CES. Figure 18 further clarifies this relationship by showing that the HLA Tool will exclusively exchange data with the Simulation Kernel. This more detailed relationship aids in the development of a Physical View, as well as providing a roadmap for testing, integration, and other relevant development personnel.

4 Architecture Traceability

In each of the views described in this ADD, different notation may be used to represent similar or connected elements. This is due to the fact that each view presents a different perspective on CES and its associated components and interfaces. In order to be sure that related elements are connected and accounted for properly across different views, an object model was created to trace and confirm relationships between elements. The full object model is presented in Figure 21, however, due to magnitude of this diagram, and space limitations, many of the relationships may not be clear. Figure 20 shows an illustrative example of the object model.

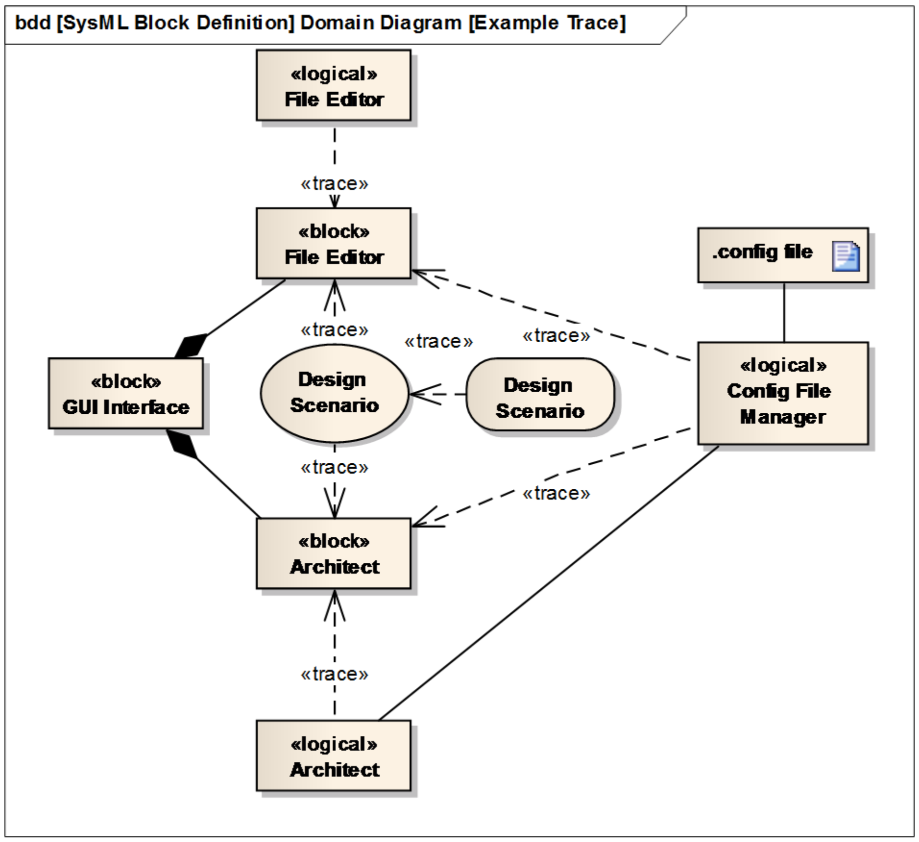


Figure 20: Object Model Trace Example

Figure 20 represents the relationships centered on the designing of a scenario. Here, the Design Scenario Use Case from the Scenarios View is linked to its corresponding activity in the Process View model. Designing a scenario can be facilitated by the Architect or the File Editor block from the High Level Decomposition in the Logical View model, which also trace to the Logical Architecture diagram. Finally, both the Architect and the File Editor make use of the Config File Manager to create and modify the .config file artifact.

Since this ADD is describing a system that is already built and fielded, and not all artifacts are readily available, there are no requirements to accompany this model. However, each of these elements, or chains of related elements, should link back to some originating or derived requirement of CES. Figure 21 represents Object Model 1, linking all block elements to their logical component counterparts. A more detailed diagram can be made available by the authors of this ADD.

Figure 21 represents Object Model 1, linking all logical components to the blocks that were used to represent them in abstract diagrams. A more detailed diagram can be made available by the authors of this ADD.

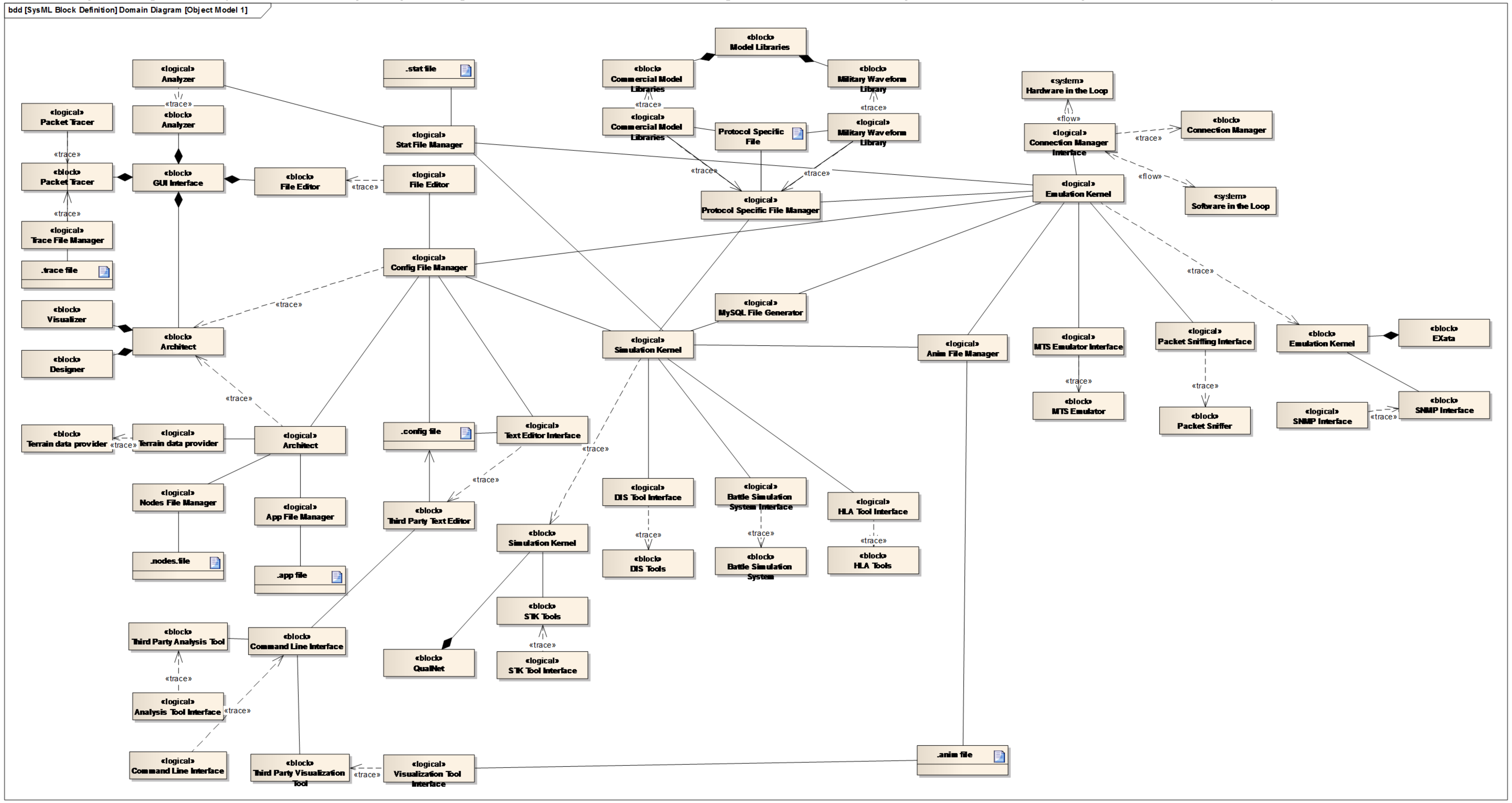

bdd [SysML Block Definition] Domain Diagram [Object Model 1]
«block» Model Libraries
«block» Commercial Model Libraries
«block» Military Waveform Library
«logical» Commercial Model Libraries
«logical» Military Waveform Library
Protocol Specific File
«logical» Protocol Specific File Manager
«system» Hardware in the Loop
«flow»
«logical» Connection Manager Interface
«block» Connection Manager
«logical» Emulation Kernel
«system» Software in the Loop
«logical» Analyzer
«block» Analyzer
.stat file
«logical» Stat File Manager
«logical» Packet Tracer
«block» Packet Tracer
«block» GUI Interface
«block» File Editor
«logical» File Editor
«logical» Trace File Manager
.trace file
«logical» Config File Manager
«block» Visualizer
«block» Designer
«block» Architect
«logical» MySQL File Generator
«logical» Simulation Kernel
«logical» Anim File Manager
«logical» MTS Emulator Interface
«block» MTS Emulator
«logical» Packet Sniffing Interface
«block» Packet Sniffer
«block» Emulation Kernel
«block» EXata
«logical» SNMP Interface
«block» SNMP Interface
«block» Terrain data provider
«logical» Terrain data provider
«logical» Architect
.config file
«logical» Text Editor Interface
«logical» Nodes File Manager
«logical» App File Manager
«block» Third Party Text Editor
«block» Simulation Kernel
«logical» DIS Tool Interface
«block» DIS Tools
«logical» Battle Simulation System Interface
«block» Battle Simulation System
«logical» HLA Tool Interface
«block» HLA Tools
.nodes.file
.app file
«block» STK Tools
«logical» STK Tool Interface
«block» QualNet
«block» Third Party Analysis Tool
«block» Command Line Interface
«logical» Analysis Tool Interface
«logical» Command Line Interface
«block» Third Party Visualization Tool
«logical» Visualization Tool Interface
.anim file
«trace»

Figure 21: Object Model 1: Tracing Blocks to Logical Componetns

Figure 22 represents Object Model 2, linking all logical components to the Use Cases they are intended to carry out. A more detailed diagram can be made available by the authors of this ADD.

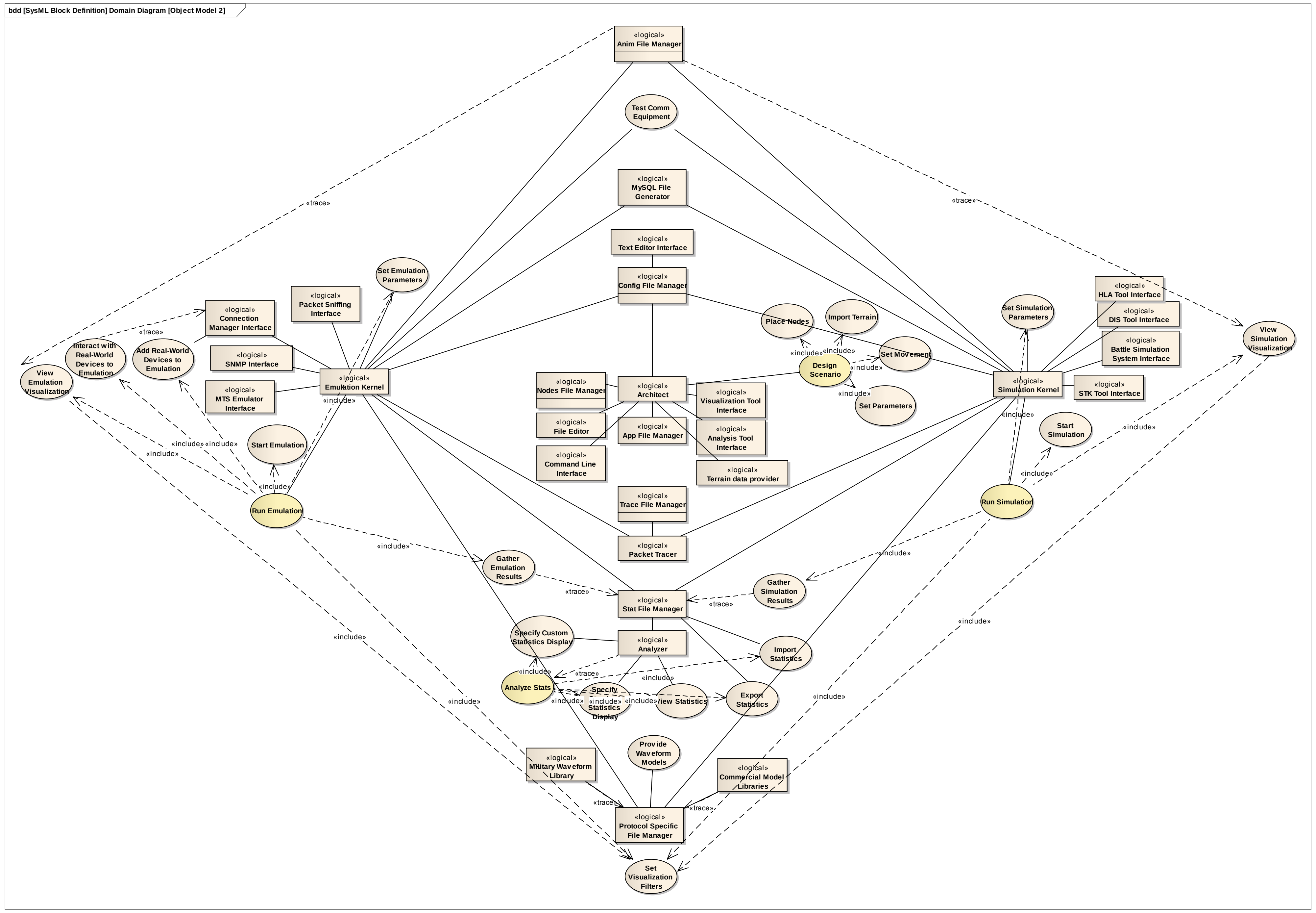

bdd [SysML Block Definition] Domain Diagram [Object Model 2]
«logical» Anim File Manager
Test Comm Equipment
«logical» MySQL File Generator
«logical» Text Editor Interface
«logical» Config File Manager
«trace»
«trace»
Set Emulation Parameters
«logical» Packet Sniffing Interface
«logical» Connection Manager Interface
«logical» SNMP Interface
«logical» MTS Emulator Interface
«logical» Emulation Kernel
Interact with Real-World Devices to Emulation
Add Real-World Devices to Emulation
View Emulation Visualization
Start Emulation
Run Emulation
«include»
Place Nodes
Import Terrain
Set Movement
Design Scenario
Set Parameters
«logical» Nodes File Manager
«logical» Architect
«logical» Visualization Tool Interface
«logical» File Editor
«logical» App File Manager
«logical» Analysis Tool Interface
«logical» Command Line Interface
«logical» Terrain data provider
«logical» Trace File Manager
«logical» Packet Tracer
Set Simulation Parameters
«logical» HLA Tool Interface
«logical» DIS Tool Interface
«logical» Battle Simulation System Interface
«logical» Simulation Kernel
«logical» STK Tool Interface
View Simulation Visualization
Start Simulation
Run Simulation
Gather Emulation Results
Gather Simulation Results
«logical» Stat File Manager
Specify Custom Statistics Display
«logical» Analyzer
Import Statistics
Analyze Stats
Specify Statistics Display
View Statistics
Export Statistics
Provide Waveform Models
«logical» Military Waveform Library
«logical» Commercial Model Libraries
«logical» Protocol Specific File Manager
Set Visualization Filters

Figure 22: Object Model 2: Tracing Logical Components to Use Cases

Figure 23 represents Object Model 3, linking all activities from the Process View to the Use Cases they are intended to carry out. A more detailed diagram can be made available by the authors of this ADD.

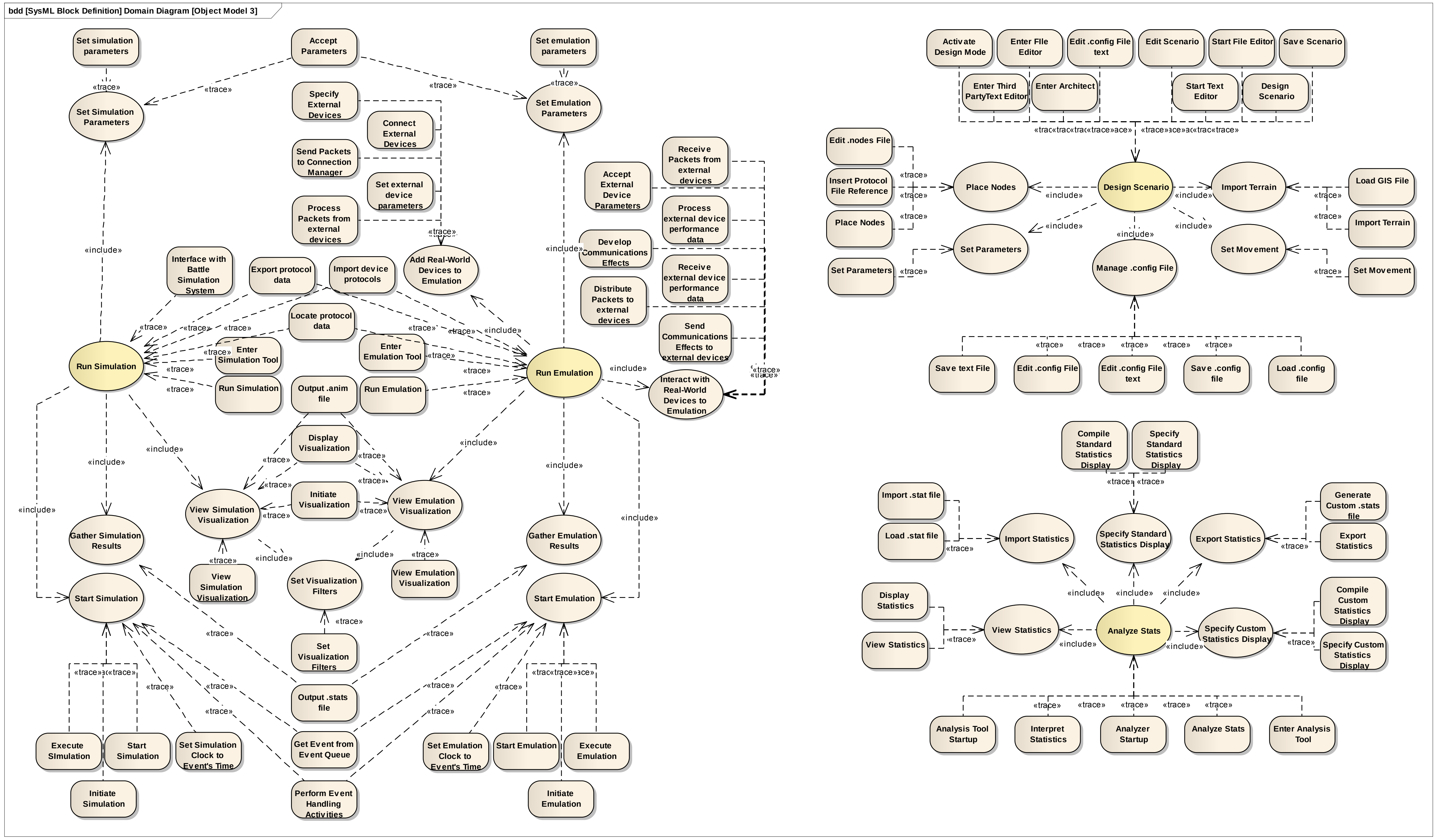

bdd [SysML Block Definition] Domain Diagram [Object Model 3]
Set simulation parameters
Accept Parameters
Set emulation parameters
Set Simulation Parameters
Set Emulation Parameters
Specify External Devices
Connect External Devices
Send Packets to Connection Manager
Set external device parameters
Process Packets from external devices
Add Real-World Devices to Emulation
Receive Packets from external devices
Accept External Device Parameters
Process external device performance data
Develop Communications Effects
Receive external device performance data
Distribute Packets to external devices
Send Communications Effects to external devices
Interact with Real-World Devices to Emulation
Interface with Battle Simulation System
Export protocol data
Import device protocols
Locate protocol data
Enter Simulation Tool
Enter Emulation Tool
Run Simulation
Run Emulation
Output .anim file
Display Visualization
Initiate Visualization
View Simulation Visualization
View Emulation Visualization
Gather Simulation Results
Gather Emulation Results
Set Visualization Filters
Start Simulation
Start Emulation
Output .stats file
Get Event from Event Queue
Perform Event Handling Activities
Execute SImulation
Start Simulation
Set Simulation Clock to Event's Time
Set Emulation Clock to Event's Time
Start Emulation
Execute Emulation
Initiate Simulation
Initiate Emulation
Activate Design Mode
Enter File Editor
Edit .config File text
Edit Scenario
Start File Editor
Save Scenario
Enter Third PartyText Editor
Enter Architect
Start Text Editor
Design Scenario
Edit .nodes File
Insert Protocol File Reference
Place Nodes
Set Parameters
Import Terrain
Load GIS File
Set Movement
Manage .config File
Save text File
Edit .config File
Save .config file
Load .config file
Compile Standard Statistics Display
Specify Standard Statistics Display
Import .stat file
Load .stat file
Import Statistics
Export Statistics
Generate Custom .stats file
Display Statistics
View Statistics
Analyze Stats
Specify Custom Statistics Display
Compile Custom Statistics Display
Analysis Tool Startup
Interpret Statistics
Analyzer Startup
Enter Analysis Tool
«trace»
«include»

Figure 23: Tracing Use Cases to Activities

5 Conclusions

Based on the diagrams described above, a few conclusions have been made concerning general aspects of the CES architecture, which have been translated to recommendations for future research. Figure 23 was particularly instrumental in developing these recommendations. Figure 23 comes from examining the intersection of two views captured in this ADD, the Scenario View and the Process View. Without development of these two views, Figure 23 would not have been created, therefore the influence of Figure 23 in the future recommendations reflects the importance of Section 3.2: Scenario View, and Section 3.3: Process View.

Along the left side of Figure 23 are the Use Cases and Activities associated with running simulations and emulations. Recreated in
Figure 24, this diagram reveals the amount of overlap between Simulation and Emulation Execution. Since the Simulation Emulation capabilities are so integral to CES, this overlap represents an area that would be difficult to expand the capabilities of CES. There are two significant places in
Figure 24 that are not within this overlap, both related to major interfaces of CES, the Use Cases associated with Real-World Devices and the Interface with Battle Simulation System Activity. These two elements represent a possible area for improvement of CES, which are reflected in Recommendation 3 below.

While altering simulation aspects of CES would integrate well with other SERC research efforts, altering emulation aspects of CES would be complex given the wide variety emulation interfaces over which the SERC has little influence, therefore Recommendation 3 is limited to the simulation portion of CES.

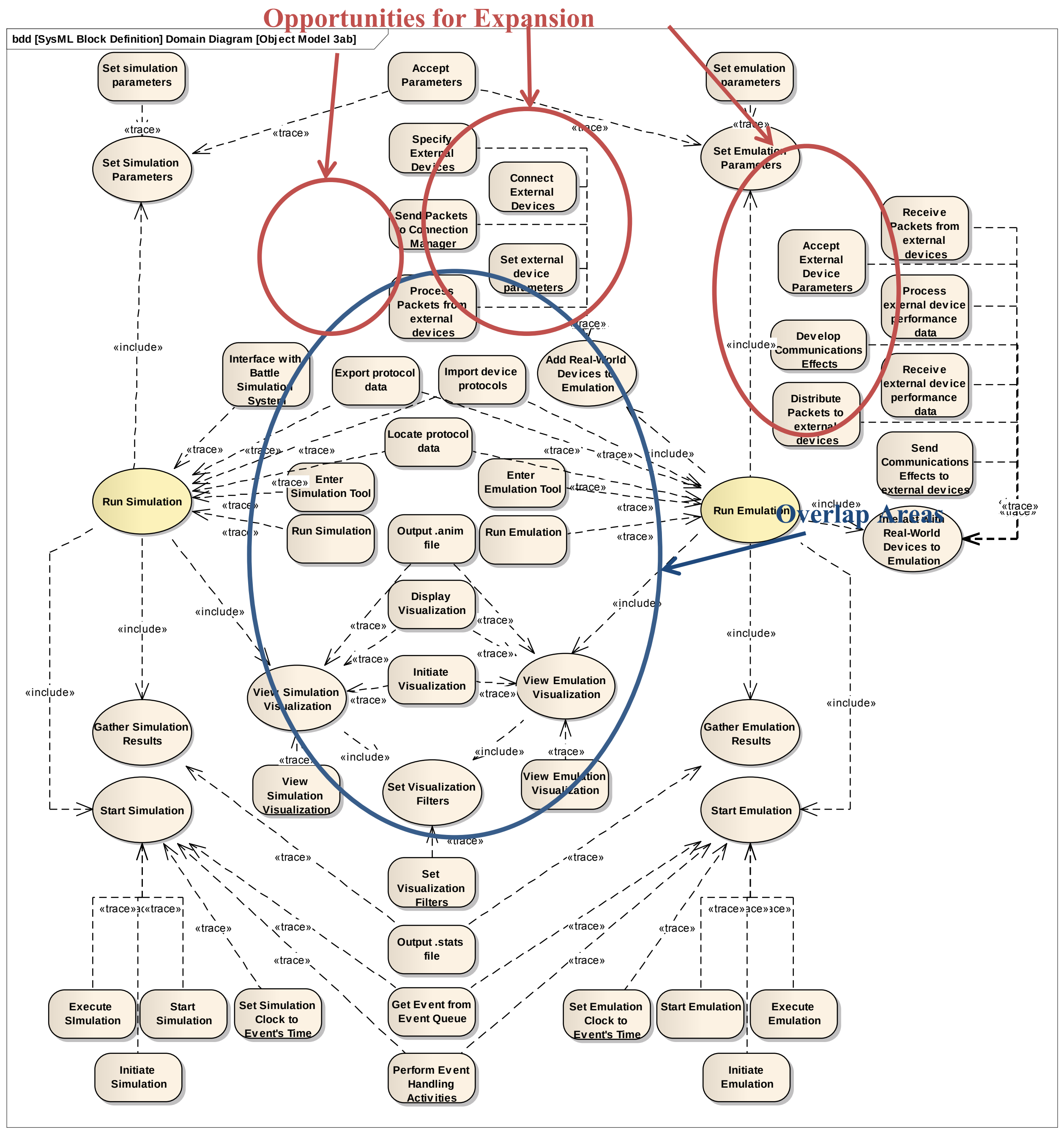

Opportunities for Expansion
bdd [SysML Block Definition] Domain Diagram [Object Model 3ab]
Set simulation parameters
Accept Parameters
Set emulation parameters
Set Simulation Parameters
Specify External Devices
Connect External Devices
Set Emulation Parameters
Send Packets to Connection Manager
Set external device parameters
Receive Packets from external devices
Accept External Device Parameters
Process Packets from external devices
Process external device performance data
Develop Communications Effects
Interface with Battle Simulation System
Export protocol data
Import device protocols
Add Real-World Devices to Emulation
Distribute Packets to external devices
Receive external device performance data
Locate protocol data
Send Communications Effects to external devices
Run Simulation
Enter Simulation Tool
Enter Emulation Tool
Run Emulation
Overlap Areas
Interface with Real-World Devices to Emulation
Run Simulation
Output .anim file
Run Emulation
Display Visualization
Initiate Visualization
View Simulation Visualization
View Emulation Visualization
Gather Simulation Results
Gather Emulation Results
View Simulation Visualization
Set Visualization Filters
View Emulation Visualization
Start Simulation
Start Emulation
Set Visualization Filters
Output .stats file
Execute Simulation
Start Simulation
Set Simulation Clock to Event's Time
Get Event from Event Queue
Set Emulation Clock to Event's Time
Start Emulation
Execute Emulation
Initiate Simulation
Perform Event Handling Activities
Initiate Emulation
«trace»
«include»

Figure 24: Object Model 3:Simulation and Emulation

Examining the right hand side of Figure 23, similar opportunities exist for expanding the capabilities of CES through its major interfaces, specifically the design mode of the Architect (Recommendation 1) and the Analysis Tool (Recommendation 2).

5.1 Recommendations for the extension of CES

1. Create functionality to allow for users to develop scenarios in a Systems Modeling Language (SysML) tool (such as Sparx Enterprise Architect) and convert them into input (.config files) for CES. This would allow for a more abstract beginning to the scenario development, which focuses CES more on the designing of new systems instead of the validation of completed ones. It also allows for the user to utilize artifacts created by model based system engineering efforts within CES, and vice versa.

   Implementation: Import/export of SysML data is typically managed through XML files, therefore to develop this functionality an XML parser would be needed to take XML data and convert it into a .config file. The .config file could then be read by CES as well as translated back to SysML notation. Likely the SysML model will not configure all parameters of the .config file, but rather create an outline which would then be filled in with the CES GUI or by manually editing the configuration files.

   Obstacles:
   - Requires new 3rd party software to run.
   - Changes to the syntax of the .config file during versioning by Scalable Technologies could cause mayhem with the parser. Would likely have to be modified for each new version of CES.
   - Requires knowledge of SysML
   - Complexity of configurable parameters in CES possibly beyond the scope of SysML.

2. Extend the analysis capabilities of CES by creating a rules based analysis of the simulation statistics. This would work by monitoring the statistics output of a scenario and comparing it to allowed value ranges; if a metric is outside the acceptable range an alert is created. This could also be extended to provide real-time alerts in an emulation scenario that could let a user know that there has been a communications failure.

   Implementation: Using the MySQL database interface, key feedback statistics could be parsed and presented in a variety of tools. Real-time emulation implementation would be more complex. Currently CES writes out statistics only during the finalization process. In order to do real time monitoring you would need to access metric data as it was being created.

   Obstacles:

- CES provides extensive statistics. Determining what is an acceptable range for each metric may not be easy. It may be best to start with the most important and basic metrics, such as packet loss or throughput.
- Changes in versioning by Scalable Technologies could potentially cause issues for the rules analysis.

3. Further integration of CES with other Army Architecting and Analysis Tools. A popular software package in use by the Army for battle space simulation is OneSAF. CES has a low level of integration with tools like OneSAF, which could be expanded to provide additional functionality of the CES.

   Implementation: Exploration of current Army tools would highlight the most relevant existing tools in use today, and further analysis would yield the value and specific applications of CES integration with these tools.

   Obstacles:
   - Creating useful feedback from CES. Could combine with intelligent analysis to provide more informative feedback for battle space simulation software.

5.2 Roadmap for RT23 Phase 2

Research for the second phase of the RT23 Research Task will include three broad areas:

- **Updating the CES Architecture Description Document**

  After a review of the ADD version 1 with CERDEC, research will be conducted to improve the ADD so that it fully reflects CES as it exists today. ADD version 2 should also include elements missing from version 1, including a Physical View, Deployment View, and other Views that were not fully captured by the ADD authors. This second iteration of the ADD will allow researchers to identify key aspects of CES architecture that can be exploited for accomplishing the following two goals.

- **Extending CES**

  Based on CERDEC feedback, research will be conducted to implement or provide guidance for implementation of the CES extensions of interest to CERDEC. These specific extensions are listed in 5

- **Integrating RT23 with RT31**

  Currently, SERC is engaged in research with ARDEC on RT31. RT31 is a research task aimed at providing system stakeholders and developers with an interactive tool for the development of a graphical Concept of Operations. RT31 will make use of Army tools OneSAF and VBS2, to aid in the development and execution of a Graphical CONOPS.

Due to the complex nature of simulating today's communications intensive systems, this graphical CONOPS tool would benefit from the introduction of CES to model communications effects throughout possible CONOPS scenarios. Through simple transformation of CES elements, it would also be possible to extend current CES functionality to manage other aspects of a graphical CONOPS development (namely Analysis of Alternatives and Concept Playback).

The goal of this research would be to introduce CES to the software currently being developed for RT31, to provide the Army with a cross-RDEC tool set for simulation and graphical CONOPS development.

Appendix A Directory

A.1 Referenced Materials

Bass 2003	Bass, Clements, Kazman, Software Architecture in Practice, second edition, Addison Wesley Longman, 2003.
Clements 2002	Clements, Bachmann, Bass, Garlan, Ivers, Little, Nord, Stafford, *Documenting Software Architectures: Views and Beyond*, Addison Wesley Longman, 2002.
Friedenthal 2008	Friedenthal, Sanford, Moore, Alan, Steiner, Rick, A Practical Guide to SysML, Elsevier Inc, 2008
IEEE 1471	ANSI/IEEE-1471-2000, *IEEE Recommended Practice for Architectural Description of Software-Intensive Systems*, 21 September 2000.
Kontio 2005	Kontio, Mikko, *Architectural Manifesto: Designing software architectures, Part 5*. IBM Developer Works, 2005.
Kruchten 1995	Kruchten, Philippe, *Architectural Blueprints—The "4+1" View Model of Software Architecture*. IEEE Software 12(6): 42-50, 1995.
Mitra 2008	Mitra, Tilak, *Documenting software architecture, Part 2: Develop the system context*. IBM Developer Works, 2008.
SEI 2004	SEI SAD Template
SERC 2010	RT paperwork for RT 23

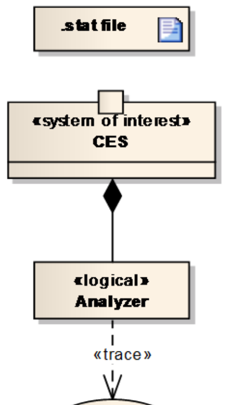
A.2 Glossary

Term	**Definition**
view	A representation of a whole system from the perspective of a related set of concerns [IEEE 1471]. A representation of a particular type of software architectural elements that occur in a system, their properties, and the relations among them. A view conforms to a defining viewpoint.
viewpoint	A specification of the conventions for constructing and using a view; a pattern or template from which to develop individual views by establishing the purposes and audience for a view, and the techniques for its creation and analysis [IEEE 1471]. Identifies the set of concerns to be addressed, and identifies the modeling techniques, evaluation techniques, consistency checking techniques, etc., used by any conforming view.

A.3 Acronym List

ADD	Architecture Description Document
API	Application Programming Interface
bdd	Block Definition Diagram
C4ISR	Command, Control, Communications, Computers, Intelligence, Surveillance and Reconnaissance
CERDEC	Communications-Electronics Research Development and Engineering Center
CES	Communications Effects Server
DIS	Distributed Interactive Simulation
DoD	Department of Defense
FCS	Future Combat System
GIS	Geographic Information System
GUI	Graphical User Interface
HLA	High Level Archititeture
ibd	Internal Block Diagram
IEEE	Institute of Electrical and Electronics Engineers
IO	Input/Output
MTS	Message Transceiver System
OTB	OneSAF Testbed Baseline
PEO-I BCT-M	Program Executive Office Integration Brigade Combat Team Modernization
RDECOM	US Army Research, Development and Engineering Command
SEI	Software Engineering Institute
SERC	Systems Engineering Research Center
SNMP	Simple Network Management Protocol
SNT	Scalable Network Technologies
STK	Satellite Tool Kit
SysML	Systems Modeling Language

Appendix B Modeling Notation



Block and Internal Block Definition Diagrams

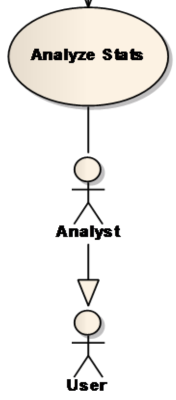
Artifact: Represents an artifact created or used by a system

Port: Describes the points at which a block interacts with other blocks
Block: basic building unit of block definition diagram used to represent a number of different elements, can be specified using a <<stereotype>>

Association (part) relationship: Indicates an element is part of another element.

Block: representing a different element, a <<logical>> component

Trace relationship: Provides general-purpose relationship between two elements.



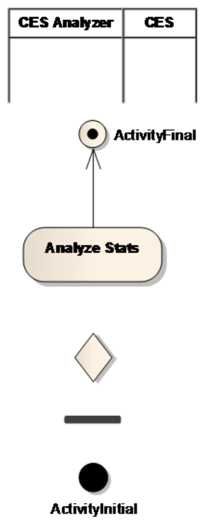
Use Case Diagram

Use Case: Indicates functionality system must provide to accomplish stakeholder goals.
Association relationship: A general linkage of two elements. May represent a data flow or general relationship.

Actor: Represents a person interacting with a system, component or Use Case.

Generalize relationship: Indicates an element is a generalization of a higher abstraction element.

Boundary: Represents an external boundary



Activity Diagram

Swimlane: (aka Activity Partition) Indicates responsibility for execution of a set of nodes contained within

Activity Final: Denotes an endpoint of a set of activities.

Directed Association relationship: Represents a generic flow of data, objects or influence between two elements.
Activity: Used to model flow behaviors of key functions or activities carried out by a stakeholder. Can be mapped to executable constructs in an execution environment.

Decision/Merge: Represents a split or merge in the flow of activities, with one input and one or more outputs (and vice versa). Denotes an OR relationship
Fork/Join: Represents a split or merge in the flow of activities, with one input and one or more outputs (and vice versa). Denotes an AND relationship or concurrency
Activity Initial: Denotes the beginning of a series of activities.