9-2021

Applying Model-Based Systems Engineering and Fidelity Quantification to Support Fair Fight in a Distributed Simulation System

Nathaniel Erbe

David Lemmer

APPLYING MODEL-BASED SYSTEMS ENGINEERING AND FIDELITY QUANTIFICATION TO SUPPORT FAIR FIGHT IN A DISTRIBUTED SIMULATION SYSTEM

THESIS

Nathaniel D. Erbe David P. Lemmer

AFIT-ENV-MS-21-S-074

DEPARTMENT OF THE AIR FORCE
AIR UNIVERSITY

AIR FORCE INSTITUTE OF TECHNOLOGY

Wright-Patterson Air Force Base, Ohio

AFIT-ENV-MS-21-S-074

APPLYING MODEL-BASED SYSTEMS ENGINEERING AND FIDELITY QUANTIFICATION TO SUPPORT FAIR FIGHT IN A DISTRIBUTED SIMULATION SYSTEM

THESIS

Presented to the Faculty

Department of Systems Engineering and Management

Graduate School of Engineering and Management

Air Force Institute of Technology

Air University

Air Education and Training Command

In Partial Fulfillment of the Requirements for the

Degree of Master of Science in Systems Engineering

Nathaniel D. Erbe David P. Lemmer

September 2021

AFIT-ENV-MS-21-S-074

APPLYING MODEL-BASED SYSTEMS ENGINEERING AND FIDELITY QUANTIFICATION TO SUPPORT FAIR FIGHT IN A DISTRIBUTED SIMULATION SYSTEM

Nathaniel D. Erbe David P. Lemmer

Committee Membership:

John M. Colombi, Ph.D.
Chair

Thomas C. Ford, Ph.D.
Member

David R. Jacques, Ph.D.
Member

AFIT-ENV-MS-21-S-074

Abstract

Current 5th and 6th generation fighter aircraft capabilities, and the DoD's push towards digital engineering have created an environment in which simulated testing using live, virtual, and constructive assets has increasing utility. These simulations need to be credible in the eyes of the stakeholders, which for distributed simulation systems includes establishing fair fight between simulation services. Fair fight is when multiple simulations interoperate without generating one-sided systematic advantages. Issues that impede fair fight can be categorized into interoperability issues at the simulation's implementation level and incompatible representations of reality between underlying models.

This research used model-based systems engineering to generate non-functional system requirements for fair fight. The requirement for alignment of models' conceptual representation of reality, led the need to quantify fidelity. Fidelity metrics that speak towards system level complexity, adherence to property scope specifications, and real-world behavior accuracy were applied to a SysML driven simulation. It was concluded that metrics for structural decomposition can describe a model's limitations and scope, while traceability between model properties and real-world specifications has more utility than a singular metric. Lastly, by generating referent data with external simulation, behavioral differences can be quantified and downstream effects due to low model fidelity can be detected.

Acknowledgments

We would like to thank and express our appreciation to our research advisor, Dr. Colombi and committee members, Dr. Ford and Dr. Jacques, for their guidance and support throughout the course of this thesis effort. We would also like to thank Dr. Long for his efforts towards providing us with this opportunity and his mentorship along the way. Lastly, we'd like to thank AFLCMC/XA and the Mission Partner's Capability Office for sponsoring our research efforts over the past two years.

Nathan Erbe and David Lemmer

Table of Contents

Page

List of Figures

Page

List of Tables

Page

List of Acronyms

AGSS	Aggregated Simulation System
ASLP	Aggregate Level Simulation Protocol
BDD	Block Definition Diagram
CL	Conceptual Linkage
DIS	Distributed Interactive Simulation
DoD	Department of Defense
DSEEP	Distributed Simulation Engineering and Execution Process
DSS	Distributed Simulation System
DT	Developmental Testing
EA	Enterprise Architect
HLA	High Level Architecture
KM	Knowledge Management
KPP	Key Performance Parameters
KSA	Key System Attributes
LCIM	Levels of Conceptual Interoperability Model
LVC	Live Virtual Constructive
MBDE	Model Based Data Engineering
MBSE	Model Based System Engineering
MOE	Measure of Effectiveness
MOP	Measure of Performance
MOS	Measure of Success
MSaaS	Modeling and Simulation as a Service
OOSEM	Object-Oriented Systems Engineering Method
OT	Operational Testing
OWL	Web Ontology Language
PE	Process Engineering
RAPIDS	Rapid Analysis Processing Independent Deployable System
RDF	Resource Description Framework
QRIP	Quick Reaction Instrumentation Package
SDEM	Simulation Data Exchange Model
SME	Subject Matter Expert
STK	Systems Tool Kit
TENA	Test and Training Enabling Architecture
VEVA	Procedure Model for Application of the VintEL-Architecture
VVUQ	Verification Validation and Uncertainty Quantification
V & V	Verification and Validation
XML	Extensible Markup Language
XMSF	Extensible Modeling and Simulation Framework

APPLYING MODEL-BASED SYSTEMS ENGINEERING AND FIDELITY QUANTIFICATION TO SUPPORT FAIR FIGHT IN A DISTRIBUTED SIMULATION SYSTEM

I. Introduction

Background or General Issue

The US DoD operational environment requires highly skilled personnel that are trained and ready to operate complex systems. These complex systems also require verification and validation prior to deployment. Meanwhile, US National Security policy requires that classified information which can be related to the performance of these complex systems be protected from unauthorized disclosure. Finally, there are spatial, legal, and fiscal constraints around the discharge of munitions and emissions in the electromagnetic spectrum (FCC, 2021). The confluence of these requirements and constraints creates a situation where a Live Virtual and Constructive (LVC) simulation is one of the few solutions that enable these requirements to be met while being subject to these constraints (Colombi, Cobb, & Gallegos, 2012).

Simulated environments for the purpose of test and training have been a pursuit of the DoD since the 1980s (Rhees, 1981). During this 40-year history, an enduring goal of these digital test and training environments has been to improve the quality of developmental and operational testing while reducing its cost. Early iterations allowed a researcher to change parameters and examine the impacts and outcomes resulting from behaviors and assumptions given to entity models being simulated in a scenario. In the 1990s, there was focus on creating joint simulated training environments which seemed

to struggle with atomic level interoperability issues resulting in the development of data type standards (Miller, 1995). As these interoperability standards have evolved in the 2010s the focus shifted from data type standards to metamodels and more conceptual level interoperability considerations (Tolk, Interoperability, Composability, and Their Implications for Distributed Simulation, 2013) (Gore, 2007).

The shift in focus of interoperability standards from the atomic data level concerns to conceptual level concerns outlines how Distributed Simulation Systems (DSS) have matured from a technical perspective. As a result, simulation experiments and training environments now regularly utilize hardware that is geographically separated and developed by different technical teams. This created new incentives in the domain of simulated test and training that value the modularity of simulation components in order to enable reuse, extensibility, and configurability of the simulation environment. The combination of distributed conditions and model modularity introduce new interoperability requirements that cannot be satisfied through the atomic data interoperability standards previously developed. At present there are numerous efforts that seek to manage the interoperability concerns of DSS by utilizing various architectures and standards such as Extensible Modeling and Simulation Framework (XMSF), Test and Training Enabling Architecture (TENA), Distributed Interactive Simulation (DIS), Aggregate Level Simulation Protocol, High Level Architecture (HLA), Ontological and Semantic Web methods (SR, 2011) (Tolk, Engineering Principles of

Combat Modeling and Distributed Simulation, 2012). This new domain of challenges and capabilities is the subject of the work completed in this research.

Problem Statement

Current 5th and 6th generation fighter aircraft capabilities may be better tested in fully simulated environments to both safeguard test results and completely characterize performance (Menke T. , 2019). Distributed Simulation Systems are a potential solution. However, there are a number of challenges associated with these environments around the interoperability of simulations and the composability of their baseline models (Tolk, Engineering Principles of Combat Modeling and Distributed Simulation, 2012). Problems arising from interoperability and model composability degrade the fidelity of the results produced through simulation. These problems are exacerbated by the trend over the last 20 years to have these simulated environments operate across distributed networks and even across simulation platforms (Menke T. , 2019). Given the disadvantages of fully testing capabilities on live ranges, stakeholders are dependent on assumptions about the fidelity of information generated through simulation experiments to inform their decision-making regarding gaps and investment prioritization. These factors point to a need to be able to address model and simulation interoperability and fidelity. These issues are encapsulated within the concept of fair fight. Fair fight is when two or more simulation systems interoperate in a way that does not lead to systematic advantage or disadvantage for one of the systems (Siegfried, Luthi, Herrmann, & Hahn, 2011). Currently, there are no methodologies that would create a direct traceability between

relationships of “Measures” (MOE, MOS, MOP, KPP, KSA etc.) that speak to the stakeholder concerns and considerations of fair fight which assert the validity of results produced through distributed simulations (Mission Engineering Guide, 2020). Traceability between these domains of stakeholder level concerns and simulation performance is critical to ensure that technical development efforts remain in line with stakeholder goals and mission level objectives.

Research Objective and Questions

The objective of this research is to facilitate the verification and validation of fair fight considerations in a distributed simulation through the use of Model-Based Systems Engineering (MBSE). The research will attempt to establish system level fair fight requirements and provide traceability to their verification in a model. Doing so will provide stakeholders the necessary confidence to view their developmental and operational tests through the use of joint simulations as credible.

In support of this objective, this research proposes the following questions:

1. What are the fair fight considerations for a distributed simulation system and how can they be organized and categorized?
2. How does one ensure that fair fight considerations are built into the specifications and requirements of distributed simulation systems?
3. How can fair fight of an as-is distributed simulation system be assessed, either during setup/configuration or throughout its verification and validation process?

Methodology

The research will analyze current literature on fair fight considerations and the interoperability of distributed simulation systems, to form a baseline of stakeholder needs and requirements associated with the topic. This background research will be integrated and analyzed using MBSE tools and techniques to facilitate further exploration of the topic. The Object-Oriented Systems Engineering Method (OOSEM) will be followed to provide a top-down, scenario driven process to support the analysis, specification, design, and verification (Friedenthal & Moore, 2015) of fair fight concepts. This process will utilize SysML to generate modeling artifacts related to the structure, behavior, parametrics, and requirements of fair fight concerns. Based on the requirements for fair fight that are established, methods will be analyzed to validate them within a DSS. These methods will encompass quantifying the fidelity of executable models. A CubeSat reference architecture will be used as a test bed to examine fidelity metrics and look for their utility in assessing fair fight of DSS throughout set-up and configuration.

Assumptions/Limitations

Distributed simulations are highly complex systems with elements that are classified or limited. This makes access difficult, with an unmanageable task to learn all the elements in a timely manner. These limitations generate the need for a few assumptions to conduct this research. The first assumption is that fair fight requirements can be generated while analyzing the most basic form of a DSS. This basic DSS structure consists of a simulation environment and two external simulations which all need to

interoperate together. The fair fight requirements generated using this structure can then be extended to an operational DSS consisting of a multitude of simulation services. The second assumption is that that an aggregated simulation system (AGSS) can be used in place of a DSS for analysis. An AGSS is a simulation composed of multiple simulation tools and programs containing discrete underlying models that are composed together to create on output simulation occurring over a single processing system. Using an AGSS allows research to take place on one computer processor, significantly decreasing cost and time resources needed for the research. It also removes potential infrastructure interoperability issues allowing the research to focus on requirements for fair fight at a higher level of conceptualization. For this research, the AGSS and its fair fight problem set is assumed to be a subset of the DSS problem set. Therefore, the conclusions derived from fair fight research within an AGSS are applicable in both systems.

Implications or Expected Contributions

The results of this thesis provide the simulation and modeling community with a MBSE framework built for assessing fair fight consideration within DSS. Using the MBSE process, traceability is created between stakeholder needs and system requirements in the context of ensuring valid data outputs through the application of fair fight and fidelity requirements.

This research contributes a categorization of fair fight requirements into interoperability considerations at the implementation level and fidelity considerations at the modeling level. Additionally, the methodology developed in this research effort is

applied to an AGSS of a CubeSat reference architecture providing an example of quantitative fidelity analysis.

The quantitative fidelity analysis is a significant contribution to the fields of modeling and simulation and MBSE because this enables models and/or simulation systems to be comparable to one another with quantitative fidelity metrics. This comparison benefits these communities as well as their stakeholders because the fidelity framework deployed here considers the model as well as the intended real-world use. Secondly, mechanisms for building these fidelity metrics into an MBSE tool are proposed which can facilitate the automation of fidelity scoring. Through these artifacts and the modeling of system structure, behavior, and parametrics the verification and validation of fair fight can be documented for these systems.

Preview or Summary

This thesis is composed of five chapters encompassing an introduction, literature review, methodology, results, and a conclusion section. Chapter I, the introduction, sets the need for DSS for operational and developmental testing of 5^{th} and 6^{th} generation fighter aircraft capabilities. It then details the research questions and objectives of this thesis. Within Chapter II, the literature review, topics such as DSS, interoperability, fair fight, and V & V will be discussed providing the background information necessary to both understand what fair fight is and why it is crucial element in producing credible distributed simulations. Chapter III, the methodology, will categorize requirements for fair fight and pose the need for the quantification of the fidelity of simulations underlying

models. This will be done with a step-by-step process following MBSE practices. Chapter IV will display the results of applying fidelity evaluator functions to a CubeSat reference architecture composed of Cameo, MATLAB, and STK and discuss pros and cons of each fidelity metric produced. Chapter V will conclude the thesis, providing detail on how each research question was answered and propose future research topics that will facilitate creating distributed simulations that engage in fair fight and produce credible results for their stakeholders.

II. Literature Review

Chapter Overview

The purpose of this chapter is to provide further context for the need of fair fight requirements specification and V & V activities within a distributed simulation system. It will provide background definitions and overviews of distributed simulation systems, simulation interoperability and model composability, fair fight, and verification and validation. In addition to providing necessary context around these topics, the academic literature review provided in this chapter lays out the logical path in linking the background issue to the methodological and analysis choices in Chapters III and IV.

Distributed Simulation Systems

Simulation systems take many forms to fulfil the testing and training requirements within the Department of Defense. One common categorization of simulation systems is the construct of live, virtual, and constructive (LVC). Within this construct, live refers to real people operating real systems, virtual means real people interacting with a simulated system, and constructive refers to machine to machine interactions with human in the loop control (Colombi, Cobb, & Gallegos, 2012). Emerging DoD needs related to replacing operational and developmental test range activities create the need for integrating models, simulations, and live training events (Hill, Tolk, Hodson, & Millar, 2018). The integration of live, virtual, and constructive components is known as LVC simulation. Joint training or testing activities at the tactical, operational level, or above, lend themselves to LVC simulation using a distributed simulations where multiple

simulations can be composed into one higher level system. These environments are required for both operational and technical reasons. Operational requirements for using distributed simulation include the need for combined or joint training, and the ability or need to implement future systems into the simulation environment as desired (Siegfried, 2013). Technical reasons include the ability to integrate various systems from differing locations or manufacturers (Siegfried, 2013) and the resulting cost efficiency from use of legacy systems. DSS can be described as a service-based architecture, incorporating ideas from cloud computing under the term Modeling and Simulation as a Service (MSaaS) (Siegfried, et al.). Within an over-arching DSS a simulation environment exists which consists of a simulation data exchange model, a named set of application members, and a set of agreements that they must follow. These application members consist of services providing live, virtual, or constructive simulation assets, as well as supporting services such as data logging, or visualization tools (Siegfried, et al.). The use of these application members allows the simulation to plug and play services based on the requirements of the developmental or operational test activity. The simulation environment manages these members, also known as simulation services, through a defined architecture that establishes standards and protocols for their integration. Common architectures include DIS, HLA, Aggregate Level Simulation Protocol (ASLP), and TENA. These architectures along with other protocols established by the simulation environment focus on achieving interoperability between simulation services and the simulation environment as described in the following section.

Simulation Interoperability and Model Composability

Successful interoperation of services within the simulation system depends on three governing concepts of interoperability to be met. These are the "integratability of infrastructure, the interoperability of the simulation systems, and the composability of the underlying combat models" (Tolk, Engineering Principles of Combat Modeling and Distributed Simulation, 2012). From the bottom up, integration of the physical connection between systems such as networks, hardware and firmware must occur. Then the system must interoperate to establish a common understanding of data elements and information that is being exchanged. Lastly, the composition of the underlying models used to simulate the system must be understood and aligned with other model's abstractions and assumptions about reality. The layers of interoperability within a simulation can be further defined using the Levels of Conceptual Interoperability Model (LCIM) as shown in Figure 1 (Siegfried, Luthi, Herrmann, & Hahn, 2011). A general description of each layer has been pulled directly from (Siegfried, Luthi, Herrmann, & Hahn, 2011) with more robust definitions being described in (Tolk, Engineering Principles of Combat Modeling and Distributed Simulation, 2012).

- Level 1: Technical interoperability: Communication infrastructure and corresponding protocols exist.
- Level 2: Syntactic interoperability: Common structures for information exchange such as common data formats exist.
- Level 3: Semantic interoperability: Shared meaning of data.

- Level 4: Pragmatic interoperability: mutual awareness of methods and procedures between simulation systems.
- Level 5: Dynamic interoperability: interoperating simulation systems are also mutually aware of changes in assumptions and constraints over time.
- Level 6: Conceptual interoperability: a fully documented overall conceptual model exists.

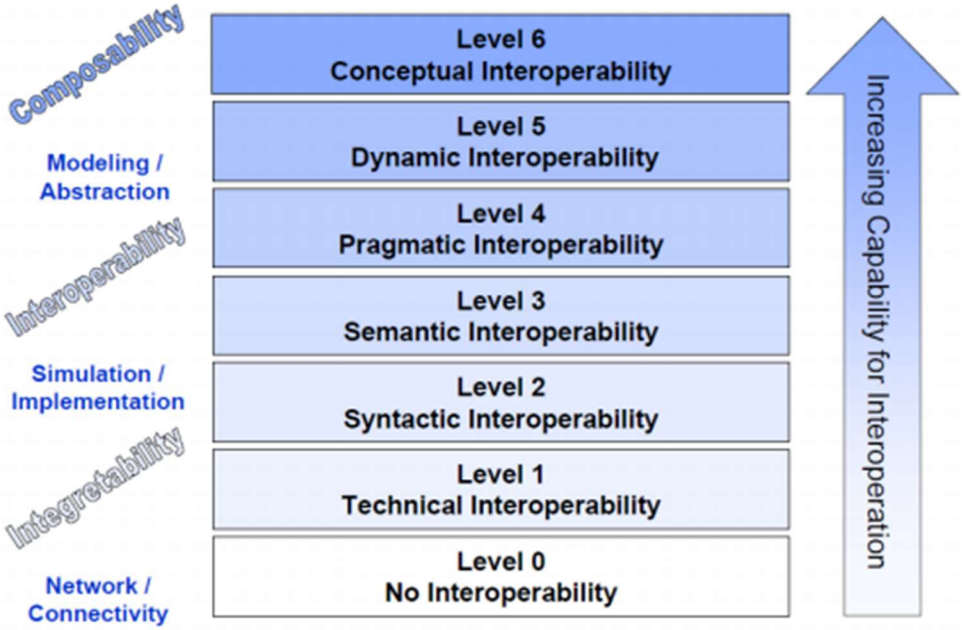

Figure 1: Levels of Conceptual Interoperability (Siegfried, Luthi, Herrmann, & Hahn, 2011)

These levels of interoperability can be used to compare architectures, standards, and protocols used by the simulation environment and analyze up to what level they address interoperability challenges. The following diagram, Figure 2, compares the protocols of

DIS, HLA, Extensible Markup Language (XML), Resource Description Framework (RDF) for Services, OWL, OWL for services, and rules, as well as potential solutions to meet interoperability levels that are not met through these stated protocols. These potential solutions include model-based data engineering (MBDE), process engineering (PE), and conceptual linkage (CL) (Tolk, Diallo, King, & Turnitsa).

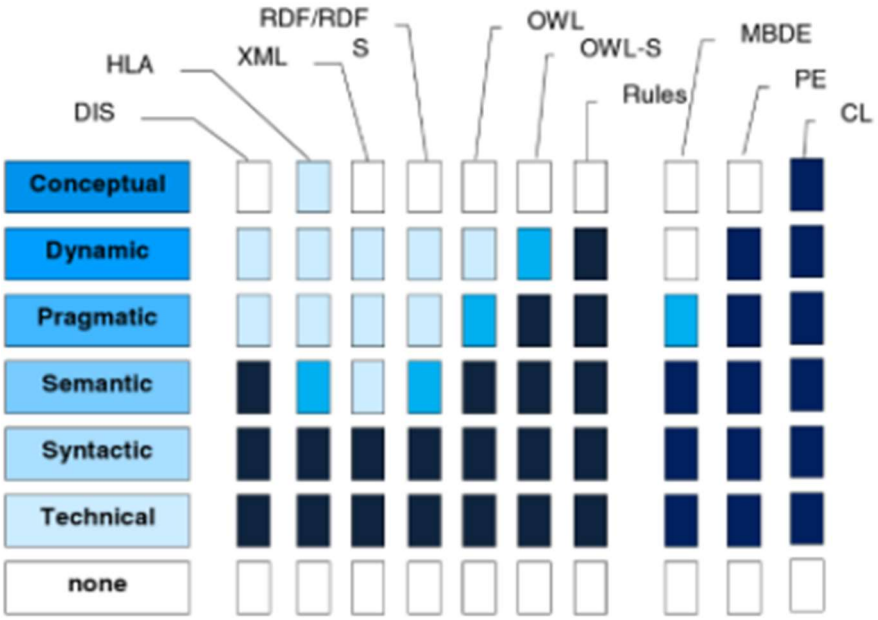

Figure 2: Mapping of Protocols to LCIM (Tolk, Diallo, King, & Turnitsa)

This diagram provides a brief glance into the challenges and attempts to achieve interoperability within a DSS. As shown, achieving dynamic and conceptual interoperability (the composability of the models) is of particular difficulty. It is not sufficient to have knowledge of distributed systems and computer engineering methods, one must also address specific modeling aspects (Tolk, Engineering Principles of Combat Modeling and Distributed Simulation, 2012).

Fair Fight

When dealing with DSS a unique challenge within the field of interoperability emerges, known as “fair fight". Fair fight is defined as when two or more simulation systems interoperate in a way that does not lead to systematic advantage or disadvantage for one of the systems (Siegfried, Luthi, Herrmann, & Hahn, 2011). Fair fight is not focused on establishing a fair playing field between blue and red forces but rather accurately portraying the effects of such an interaction to achieve reliable testing, training, and analysis. To further exemplify this issue, two models within a DSS can both be correct and consistent with themselves; however, when composed together the inconsistencies in their modeling characteristics and base assumptions generate undesirable effects that do not reflect reality. As (King, 2009) demonstrates, two conceptually incompatible simulations that have been integrated and are interoperable through technical means can operate without errors. The output of these simulators, however, may not accurately represent reality and fair fight violations will occur. These violations may occur through a variety of fair fight considerations that must be designed into the models behind distributed simulation systems. As described in (Siegfried, Luthi, Herrmann, & Hahn, 2011), fair fight problems may arise due to causes such as different environmental representation, object representation, capability definitions, computation of visibilities, weapon effects computation, time management inconsistencies, or bandwidth in communication between data. With the vast number of modeled entities on today’s battlespace, as well as increasingly complex avionics associated with systems and

ever-changing environments in which they operate, the need to harmonize these factors is crucial. Taking radar as an example, modeling choices such radar range equations, filtering, signal processing and different characterization of radar models all effect how the radar will impact and be received by other systems (Menke T. , 2019). These other systems may be simulated with a different set of assumptions and fidelity of their radar models creating an inherently unfair fight between the interacting systems. Relating fair fight issues back to the previously described LCIM, fair fight is considered to be a subset of pragmatic interoperability, since technical or architectural means are not sufficient to prevent violations (Siegfried, Luthi, Herrmann, & Hahn, 2011). The root causes of these violations, however, may occur throughout the levels of interoperability and violations may occur due to a combination of factors. To truly achieve a valid simulation result and not incur fair fight violations all levels of interoperability must be met. This includes awareness of methods and procedures between simulation systems, awareness of the impact of time on assumptions and constraints, and fully documented conceptual models for each element in the simulation system. It is important to note that while fair fight requires simulation interoperability at all levels, having simulation interoperability does not guarantee fair fight (Siegfried, 2013).

Verification and Validation

Fair fight at its essence can be considered an issue of verification and validation. Validation deals with ensuring that the models and simulations are an accurate representation of the real or envisioned system as defined by the stakeholder. Verification

aims to ensure that this valid model is implemented correctly through proper modeling mechanisms and transformations (Tolk, Engineering Principles of Combat Modeling and Distributed Simulation, 2012). Within the context of fair fight for a distributed simulation environment, validity contends with not only representing individual models to the fidelity desired by the stakeholders but also ensuring conceptual compatibility of the fidelities between models. Conceptual model validation is a key step within the model development process as laid out in Figure 3.

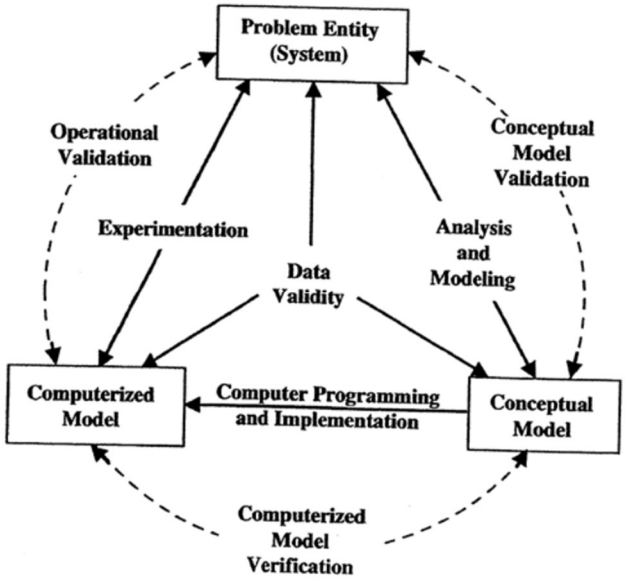

Figure 3: Verification & Validation (Sargent, 2014)

Conceptual model validation contends with ensuring the theories and assumptions underlying the conceptual model are correct and that the model's structure, logic, mathematical, and causal relationships reasonably represent the problem entity for the purpose defined by stakeholders (Sargent, 2014). When composing two simulation

services together in a DSS not only must their underlying models be conceptually valid, but their conceptualization must be compatible between one another to avoid fair fight violations. To do this, verification and validation (V & V) of model composability at both the technical and conceptual level must occur. Furthermore, V & V are asserted for a particular application with a given tool or model. This context is extremely important when considering a DSS and fair fight because of the modular and service-oriented architecture employed. The only feasible method for maintaining confidence in the validity of these simulation results must allow for verification to be asserted without requiring a manual V & V study for each possible configuration.

Finally, it is vital that V & V efforts are related to DSS stakeholder concerns in a way that describes how these technical considerations relate to desired information artifacts described in the Mission Engineering Guide such as Mission Satisfaction, Losses, Expenditures, and Readiness. There have been previous efforts that seek to standardize and operationalize the design and development processes for DSS and DSS generated artifacts such as VEVA and DSEEP (IEEE, 2011) (SR, 2011). In summary VEVA and DSEEP create a framework that bakes in the necessary interactions between stakeholders, SMEs, and developers into the creation process of a DSS or a DSS experiment. One complimentary or alternative approach would be to associate common stakeholder level concerns generated from documents such as Commander's Intent with linkage to the National Defense Strategy, Defense Planning Guidance, Campaign Plan, Combatant Command Operational Plan, Joint Warfighting Concept and the Mission

Engineering Guide with data validation concerns such as fair fight. For example, a stakeholder may desire to run a simulation experiment to gain insights into a best-case scenario for intercepting and mitigating an air-based strike on a blue target. In this case the stakeholder is concerned primarily with avoiding losses while minimizing cost and impacts to readiness. The DSS developers then generate a set of configurations which will test factors that may relate to the measures of effectiveness for minimizing loss, and based on results suggest responses optimized around the variables of losses incurred, cost of the response, and the effects on readiness at the blue cite after the encounter. The simulation environment can execute multiple variations to find the most relevant factors related to these points of interest; however, without the ability to speak to the validity of interaction between each element engaged in these scenarios the trustworthiness of these results cannot be asserted. The inclusion of the fair fight considerations relative to the stakeholder's need for information will also help to add transparency to the information generated from a DSS.

Summary

This chapter describes the areas of study that the authors find to be most relevant to addressing challenges faced in the modern DSS community. The area of simulation interoperability and model composability in the context of DSS deals primarily with ensuring that multiple models can function together and that they function as intended. The LCIM was developed in this area as a method to characterize interoperability and composability problems or conversely help with generating requirements to achieve a

desired level of conceptual interoperability. The LCIM is relevant in the current work due to its utility in scoping solutions to the interoperability level indicated. LCIM combined with fair fight provides an elegant approach towards classifying requirements for distributed simulation system and sub-system needs which speak to data validity directly. Fair fight considerations are a specific type of application for the field of verification and validation; by expanding the context out of the realm of DSS to the level of decision maker concerns it becomes apparent that by providing a framework for the application of these principles these processes can create a flow between them. It is the aim of the current work to encapsulate associations between the needs of simulation interoperability and model composability by expressing these as fair fight requirements and use the artifacts from the relationships of measures to create traceability from customer to developer.

III. Methodology

Chapter Overview

This chapter will describe the process followed to achieve the research objectives of establishing, specifying, and assessing fair fight requirements within a DSS. The process will begin with using MBSE to analyze stakeholder's needs and establish a set of functional requirements for achieving fair fight within a DSS. A method to quantify the fidelity of the underlying executable models will then be proposed to achieve specification and assessment of these functional requirements. The methodology will then discuss the choice of an aggregated simulation system (AGSS) composed of a CubeSat reference architecture in order to test the application of fidelity metrics on executable models. Lastly, criteria for choosing applicable fidelity metrics in relation towards fair fight considerations and their implementation within an MBSE tool will be discussed.

Overview of Research Methodology

MBSE methods and tools will facilitate answering the research questions and achieving the objectives of requirements specification and V & V of fair fight considerations. Using MBSE allows for the simplification of the validation process, given that if a system fulfills all requirement metrics within a specified margin, the system can be considered valid (Tolk, 2013). This differs from informal testing methods for assessing fair fight that rely on subject matter experts (Sargent, 2014). While tests based on experience and comparable solutions have their place in the V & V process (Sargent,

2014), manually detecting violations of simulation environment agreements is usually not possible (Siegfried, Luthi, Herrmann, & Hahn, 2011). Instead, automatic assessment of fair fight considerations should be the goal. This requires formal specification of verification criteria which is where the use of MBSE and SysML provide a pertinent application. Using SysML as the graphical modeling language will provide support for analysis, specification, design, verification, and validation of DSS/AS through some set of the nine common diagrams as shown in Figure 4 (Delligatti, 2013).

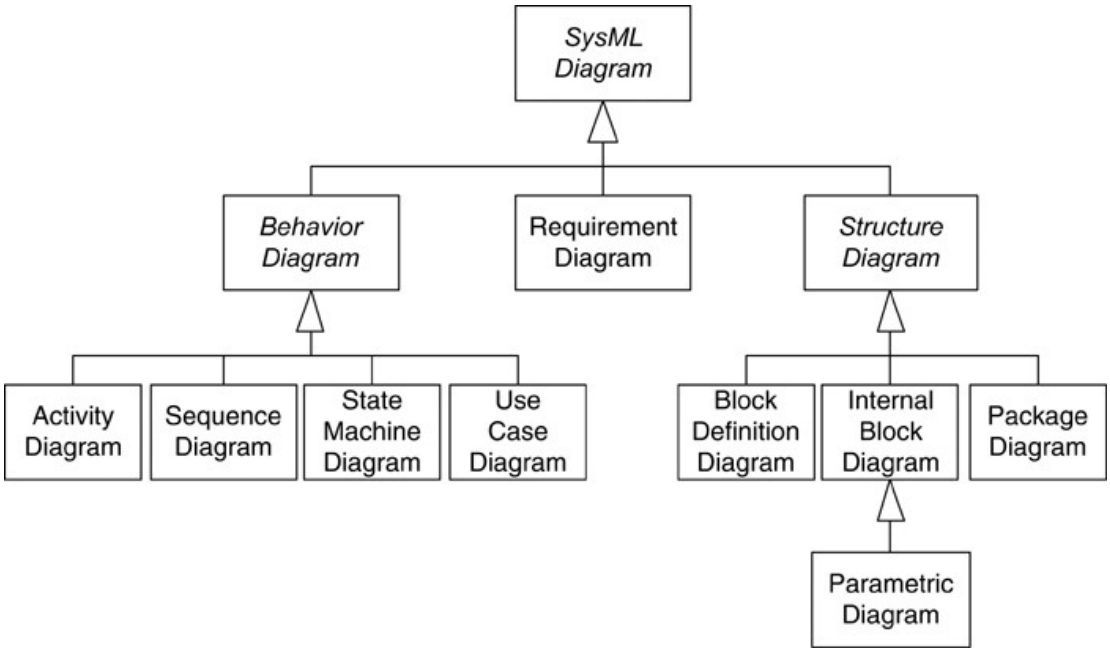

Figure 4: SysML Diagrams (Delligatti, 2013)

Throughout these diagrams the fair fight considerations of a distributed simulation environment with respect to its structure, behavior, parametrics and requirements can be modeled. The chosen modeling tool for this effort will be Cameo Systems Modeler due to

its relevance within the DoD/Air Force and systems engineering community, providing further applicability for the results and findings. However, any MBSE and/or SysML-based tool could have been used, such as IBM Rhapsody, SparxSystems Enterprise Architect (EA) or SPEC Innovation's Innoslate. Lastly, the MBSE approach in this thesis follows the Object-Oriented Systems Engineering Method (OOSEM) to provide a top-down, scenario-driven process. OOSEM includes fundamental systems engineering activities and will lead us from analysis of stakeholder needs through requirements specification and ending with verification of the system. This chapter will generally follow the steps in the simplified approach to OOSEM as seen in Figure 5 (Friedenthal & Moore, 2015) up through the requirements generation process.

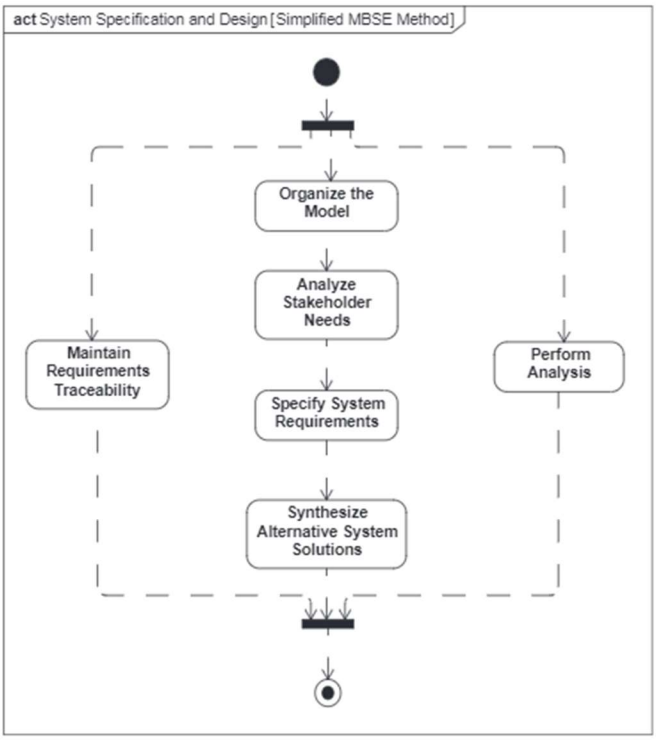

Figure 5: Simplified MBSE Process (Friedenthal & Moore, 2015)

Analyze Stakeholder Needs

Stakeholders desire a level of confidence in which they can view their developmental and operational tests within a DSS as credible. To establish this confidence mechanisms to assess fair fight considerations need to be established rather than relying on opinions of subject matter experts. By establishing a base set of functional requirements associated with achieving fair fight within a DSS, stakeholders will be able to better assert the validity of their simulation. This will increase the credibility of DSS as a whole and provide the confidence necessary for decision makers to act off the simulation results. In order to facilitate the process of defining fair fight requirements for a DSS, the decision has been made to scope and strip down the system to just the base components necessary to display the interaction needed for fair fight modeling. The system being analyzed will consist of a simulation environment with an established simulation data exchange model (SDEM) and two separate application members that each provide a LVC simulation asset. Descriptions of these components and a fuller list of services within a DSS can be seen in the LVC Simulation section of

the Literature Review. Pictured in Figure 6 is a block definition diagram (BDD) of the DSS's structure that is being analyzed in this research.

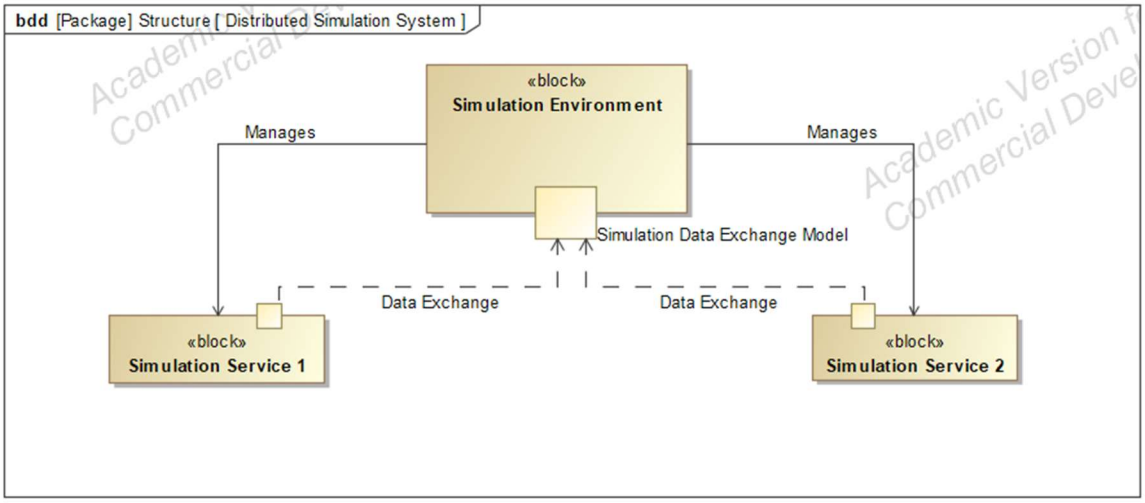

Figure 6: Basic DSS Structure

The next step in this analysis of stakeholder needs is defining use cases for the associated mission objective of achieving fair fight within the DSS. The use cases for a DSS are vast and extensive consisting of OT, DT, training, and experimentation. These main use cases for a DSS all have a common need for V & V which includes an assessment of fair fight considerations.

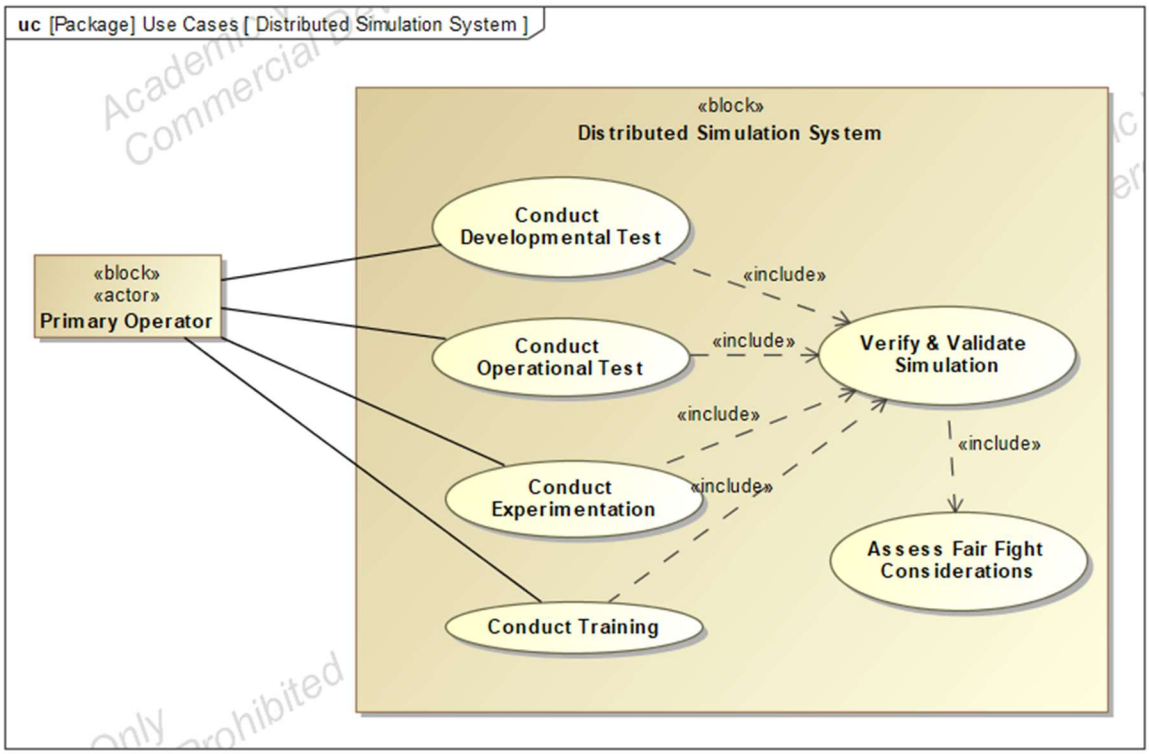

Figure 7: Fair Fight Use Case

This use case diagram along with the BDD, demonstrate the importance and need to generate requirements for achieving fair fight. By looking at fair fight through the lens of a basic DSS consisting of two models, a simulation environment, and the interface that interoperates all three, the SDEM, requirements are derived and related to real world application. This paradigm is especially important for establishing the need for interoperability at the implementation level and proper fidelity at the modeling level as described in the following section.

Specify System Requirements

The task of defining a set of fair fight requirements for a DSS began in the literature review stage with a collection of sources on typical fair fight problems and simulation interoperability issues. Three critical sources for this review were (Task Group MSG-086, 2015), (Siegfried, Luthi, Herrmann, & Hahn, 2011), and (Tolk, Engineering Principles of Combat Modeling and Distributed Simulation, 2012). Not only did these sources provide a collection of problems and issues, but (Task Group MSG-086, 2015) and (Tolk, Engineering Principles of Combat Modeling and Distributed Simulation, 2012) related these fair fight problems back to the LCIM. This established a baseline from which to begin writing formalized functional requirements using common systems engineering shall statements. As stated previously, for fair fight to occur within a distributed simulation system, interoperability at all levels of the LCIM must occur. This means meeting requirements criteria from the technical level through conceptual level. This need for interoperability occurs at the implementation level of DSS, between the simulation environment, SDEM and application members. While the SDEM facilitates this interoperation between the three it is considered a part of the simulation environment along with other established standards. Full interoperability of this interaction is the first step towards achieving fair fight. An overlay demonstrating the different needs between the implementation level and modeling level has been applied to the BDD of the DSS for clarity in the Figure 8.

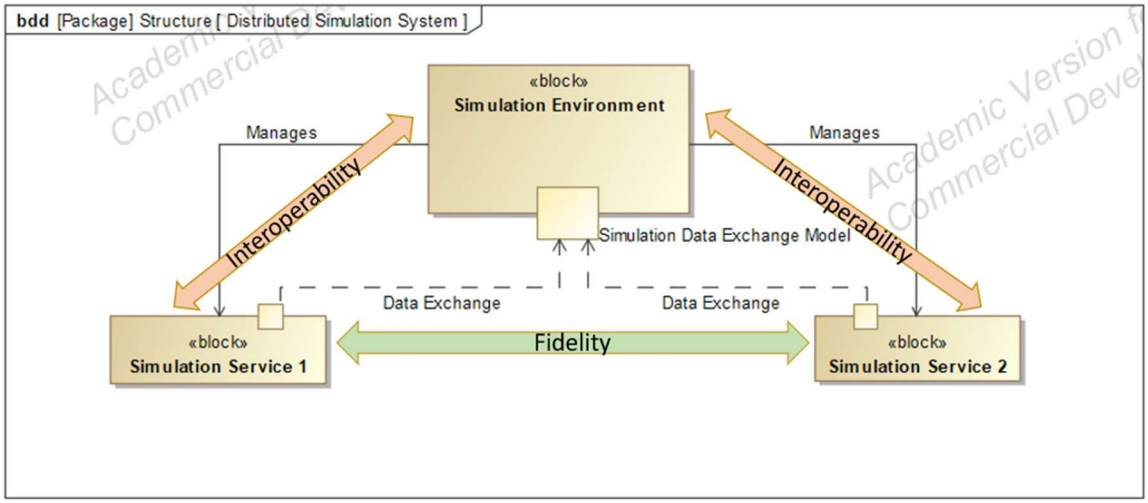

Figure 8: Interoperability & Fidelity within a DSS

The next set of requirements for fair fight deals with the modeling level and are associated with representation of the real world by the application member's models. It is at this level that it can be stated that full interoperability at the implementation level does not guarantee fair fight. At the modeling level, compatible fidelities of the conceptual models are required but only to the extent that the purpose of the DSS is not violated (Siegfried, 2013). This creates the need for purpose driven requirements associated with fidelity and a framework for which they can be established. To establish this framework, fair fight problems at the modeling level discovered in the literature review were analyzed and categorized based on common features and patterns. Three classes of issues emerged around the fidelity of conceptual models. The classes correlated with the entity-event-state paradigm, which is the minimal information needed for a simulation component at the modeling level (Tolk, Engineering Principles of Combat Modeling and Distributed Simulation, 2012). Within this paradigm, the model contains an entity, which

is an object of interest in the system; states, which are how that entity can be described throughout the flow of time; and events, which are instantaneous behavioral reactions which change the entity's state. These classes allow fair fight requirements to be established in the sub-categories of structure, timing, and behavior which all can be modeled with various fidelities. The full description of the established requirements for fair fight in a DSS are laid out in the requirements table, Figure 9. The collection of fair fight issues discovered in the literature review are related back to the categories proposed in the requirements table in Appendix 1.

#	Name	Text
1	1 Valid Simulation	The system (distributed simulation system) shall produce simulation results that are correct with reference to its intended real world application.
2	1.1 Fair Fight	The system shall provide equal treatment to both simulated models producing a fair representation of their interactions together.
3	1.1.1 Interoperability	At the implementation level, the system shall be fully inter operable between the simulation environment and application members.
4	1.1.1.7 Network Integratability	The systems technical connections shall be fully integratible including hardware, firmware, protocols, and networks.
5	1.1.1.7.1 Technical Interoperability	The system shall have communication infrastructure providing the ability to exchange data between the simulation environment and application members.
6	1.1.1.7.2 Syntactic Interoperability	The system shall have an agreed to protocol providing common information structures for the simulation environment and application members.
7	1.1.1.8 Simulation Interoperability	The systems implementation software shall be fully interoperable including exchange of data elements and middleware.
8	1.1.1.8.1 Semantic Interoperability	The system shall have an agreed to set of grammatic terms that provides a shared meaning of data between the simulation environment and application members.
9	1.1.1.8.2 Pragmatic Interoperability	The system shall have a shared meaning of terms and methods that provides context of states, processes, and meaning between the simulation environment and application members.
10	1.1.1.9 Model Composability	The systems modeling information shall be fully composable including assumptions, constraints, processes, states, and operations.
11	1.1.1.9.1 Dynamic Interoperability	The system shall provide the simulation environment with awareness to changes in the application member models' assumptions and constraints over time.
12	1.1.1.9.2 Conceptual Interoperability	The system shall provide the simulation environment with a conceptual model of the application members exposing its information, process, states, and operations.
13	1.1.2 Fidelity	The system shall have conceptually compatible fidelities at the modeling level to the extent required for the intended real-world application.
14	1.1.2.1 Behavior	The application member models shall have necessarily aligned fidelity of their behavior.
15	1.1.2.2 Timing	The application member models shall have necessarily aligned fidelity of timing mechanisms.
16	1.1.2.3 Structure	The application member models shall have necessarily aligned fidelity of their structure.

Figure 9: Fair Fight Requirements Table

Synthesize Alternative System Solutions

Now that requirements for fair fight of a DSS have been established, the research methodology will lay out a means to develop solutions to meeting those requirements. For fair fight to occur, the conceptual models of both application members must have fidelities of their components that are aligned to the degree required for the simulation's purpose. While this is an ambiguous definition, this thesis will work to establish quantifiable metrics that can describe the structural and behavioral fidelity of the underlying models and the simulations that they produce. Timing considerations will be left outside the scope of this thesis. The quantification of fidelity facilitates the achievement of developing conceptual models that then allow a comparison between model fidelity for preemptive detection of fair fight violations.

Quantifying Fidelity

To describe the fidelity of models in a robust and methodical way, the concept of fidelity must be quantified. Prior work has been done towards reaching this goal and a background of these efforts is described in (Roza, 2005). This Dissertation, a culmination of the research done by SISO's fidelity group lead at the time, took these prior efforts and established a unified approach for measuring fidelity within simulations. Two major conclusions were proposed by this thesis. One that "fidelity is an absolute property of any model or simulation characterizing its degree of realism and doesn't equate to the relative judgement of model or simulation validity" (Roza, 2005). And two, "this characterization of realism is best expressed by an enumeration of various multidimensional and multi-

facetted measurement methods and metrics " (Roza, 2005). In conjunction with these conclusions, the concept of a model referent establishes the backbone on which one can begin measuring the fidelity of models and simulations. A fidelity referent is a formal specification of real-world knowledge that is accepted as the truth about how reality is perceived or understood (Roza, 2005). This establishment of a referent model takes fidelity from an esoteric concept to a pragmatic one (Roza, 2005). In its esoteric sense, fidelity can never truly be measured because current human epistemology can only establish approximations of reality on which to compare. By defining a fidelity referent, one can measure against the best abstraction of reality with errors and uncertainties understood (Roza, 2005). Figure 10 (Roza, 2005), demonstrates how the interpretation of reality is included within the fidelity referent as well as methods for generating this body of knowledge.

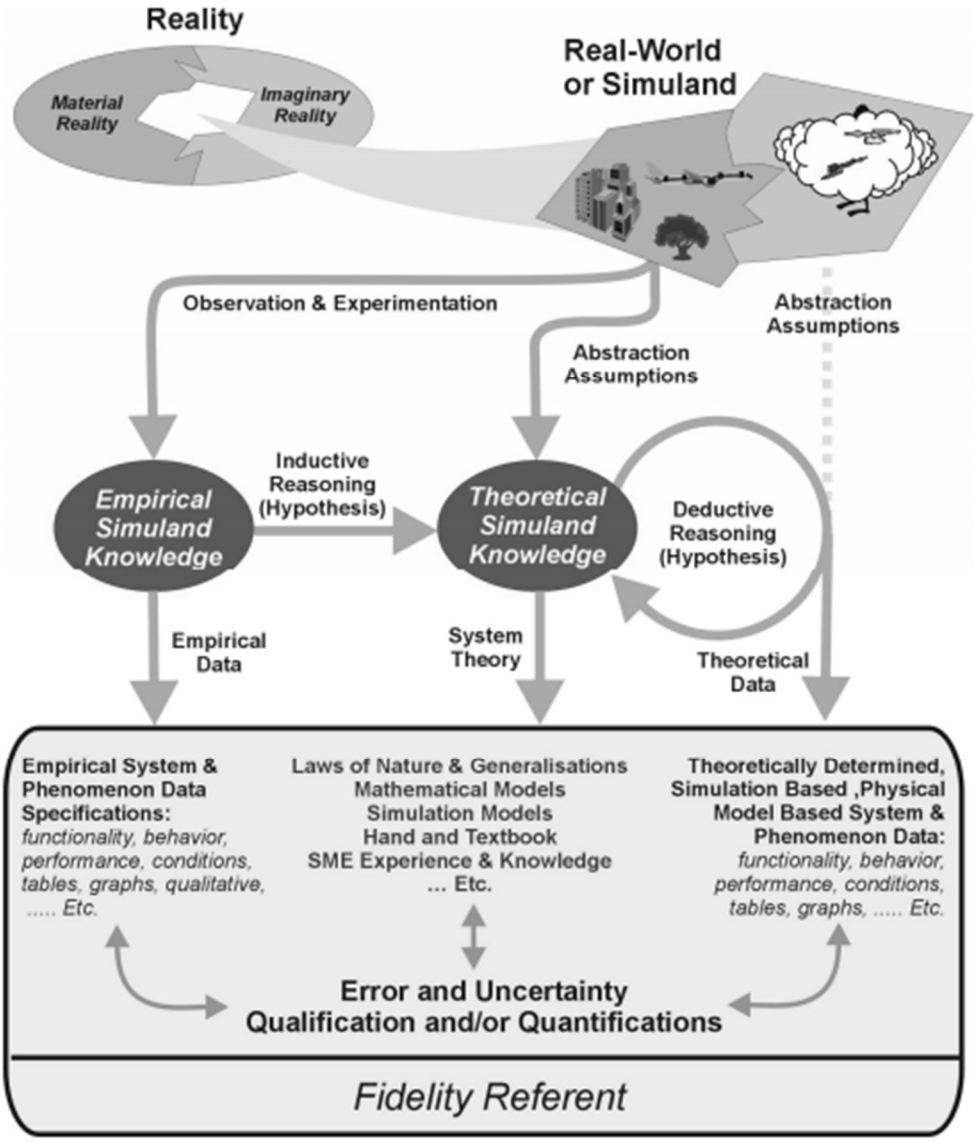

Figure 10: Fidelity Referent (Roza, 2005)

To define the knowledge within the fidelity referent, the system under analysis can be conveniently decomposed into a structural description and behavioral description. The structure specifies the inner workings of the system while behavior specifies the observable manifestation of a system over time (Roza, 2005). These two specifications of

the system trace back to the established requirements for fair fight associated with the alignment of structure and behavioral fidelity at the conceptual modeling level.

Fidelity evaluator functions are the means in which one actually measures the difference between real world reference knowledge and the simulation system (Roza, 2005). Roza's dissertation presented a taxonomy and non-exhaustive list with the most common evaluator functions used for simulation qualification and quantification as presented in Figure 11 (Roza, 2005).

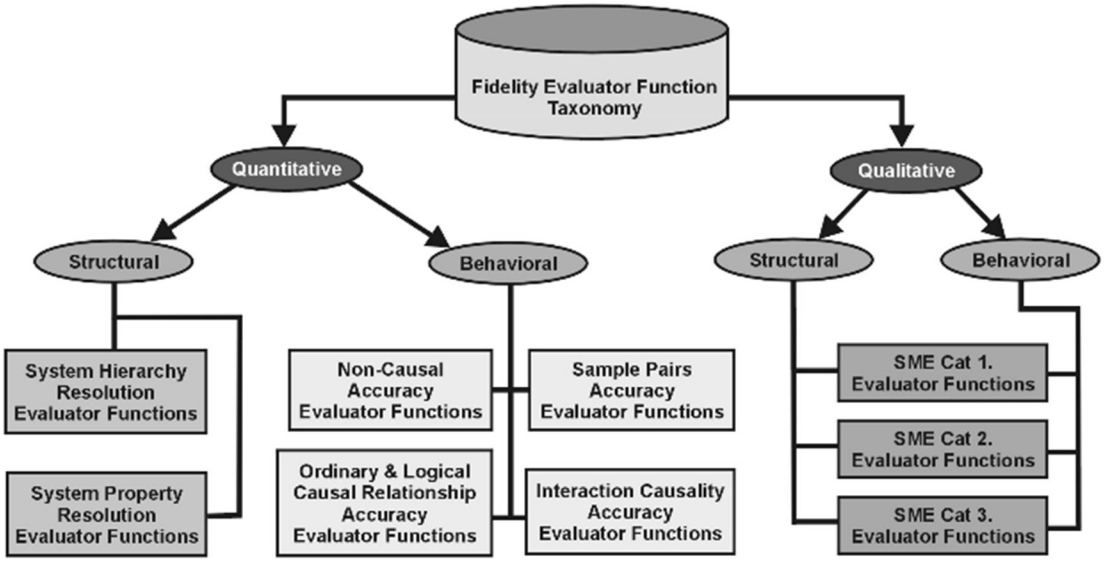

Figure 11: Fidelity Evaluator Functions (Roza, 2005)

For the purpose of the assessment of fair fight considerations, as well as scoping the research, only the quantitative functions will be analyzed. However, qualitative means may also provide some insight into fidelity of models and simulation and the implications for fair fight analysis. These common quantitative structural and behavioral evaluator functions provide the baseline in which fidelity can be measured. With these concepts in

place the next step in the methodology is to establish and gain access to a simulation environment. This will allow these fidelity evaluator functions to be applied, tested and analyzed.

Distributed Simulation System to Aggregate Simulation System

The purpose of this research is to first document fair fight considerations that are valid in DSS, establish a methodology for their assessment, and identify means to ensure their application throughout the lifecycle of a DSS. DSS are inherently complex and resource intensive requiring multiple computers and/or servers to interoperate. Because of the high investment and high value of their operation, these systems are difficult to access for the amount of time needed to develop protocols that would test fair fight considerations. Given these limitations the methodologies developed here will be tested on aggregate simulation systems (AGSS) composed of Cameo Systems Modeler, MATLAB, Simulink, and AGI's Systems Tool Kit. An AGSS is a simulation composed of multiple simulation tools and programs containing discrete underlying models that are composed together to create on output simulation occurring over a single processing system. When considering the differences between AGSS and DSS from the context of fair fight, AGSS can be considered a subset of the problem space that would be considered in a DSS. While both systems operate the execution of a single run of a simulation program, a distributed simulation occurs across multiple processors (Fujimoto, 2003). While an AGSS may occur on one machine, the elements that compose the simulation contain factors necessary for fair fight analysis. These factors include

technical, syntactic, semantic, pragmatic, dynamic, and conceptual elements of interoperability as well as variances in fidelity between structural, behavioral, and timing elements between models. By proposing that AGSS $\subset$ DSS, any analysis done within an AGSS in regards to fair fight considerations can be extrapolated up to the DSS level. The same cannot be said of analyzing fair fight issues of a DSS and stating that they all could occur at the AGSS level.

System of Study

The CubeSat reference architecture is an implementation of MBSE practices to enable rapid and consistent design, development, documentation, reuse, and testing of potential CubeSat payload builds (Kelly, 2021). For the purposes of the current research the CubeSat reference architecture offers a readily available AGSS composed of a Cameo Systems Modeler model that integrates with MATLAB to build a test scenario which is then executed by Simulink and Systems Tool Kit (Kelly, 2021). The scenario is generated based on the parameters contained within the Cameo Systems Modeler project and can therefore be adjusted to create experimental circumstances to explore AGSS for the purposes of the current research effort. Examining fair fight in this context allows for examination of structure, behavior, and timing mechanism fidelity within a built-out model. This will provide a testing ground for the methodology to be applied and results analyzed. Although this model does not have a clear referent that resultant data can be compared against, it does offer an AGSS with rapidly tunable parametric inputs that will be deployed in order to test the modeling of fair fight and fidelity considerations.

Description of Experiment Design

This thesis will analyze a subset of the Fidelity Evaluator Taxonomy presented in Roza's dissertation and apply fidelity metrics to the CubeSat reference model where deemed applicable. Considerations in this process will include a variety of factors. The first factor will be the ability to generate or obtain a fidelity referent that provides a knowledge base on which to make an assessment of fidelity. While this is important for the application of these metrics to the executable model within the thesis, it is also crucial for real world usage of these metrics. A second factor will be the ability of SysML and the modeling tool, Cameo System's Modeler, to help facilitate the execution of these metrics. By choosing fidelity metrics that can be automated and built into existing tools with a broad range of users, these metrics have a better chance of becoming mainstream mechanisms under which to assess model fidelity. The third and final factor will be the ability of the fidelity metrics to potentially provide insights into fair fight considerations within a DSS. While the establishment of fidelity metrics in the field of modeling and simulation has utility in its own right, the overall goal is to use these metrics to assess fair fight considerations throughout the verification and validation process. By doing so, one can provide stakeholders the necessary confidence to view their developmental and operational tests within a DSS as credible.

Summary

This methodology used MBSE to generate requirements for fair fight within a DSS. The requirements generation process included use case analysis and a literature

review. These requirements were categorized into interoperability requirements at the implementation level and fidelity requirements at the modeling level. To go after achieving requirements associated with fidelity, the need for a quantification of behavioral and structural fidelity was proposed. Prior work on a unified approach to simulation fidelity quantification was summarized and concepts such as fidelity referent and fidelity evaluator functions were defined. A CubeSat reference architecture was obtained as an environment in which to test and analyze these concepts. It was proposed that this reference architecture, as an aggregate simulation, was a suitable environment to define model fidelity and assess fair fight considerations in place of a DSS. Lastly, criteria were established in which to pare down the taxonomy of fidelity evaluator functions and apply suitable metrics to the CubeSat reference model.

IV. Analysis and Results

Results Introduction

During the process of scoping down Roza's Fidelity Evaluator Taxonomy, three metrics were chosen that met the criteria laid out in the methodology. These criteria were the ability to generate a referent model, ability for SysML and Cameo Systems Modeler to assist in the execution of the functions, and application towards a fair fight assessment between the executable models. Based on these criteria, evaluator functions for system level resolution, property level resolution, and real-world behavior sample accuracy were chosen and will be applied to areas of the CubeSat reference model. Within the CubeSat reference model, areas of the model were chosen for fidelity quantification that lent themselves towards these types of assessments. Detailed descriptions of the chosen modeling areas and the results that were generated are laid out in the following sections. Lastly, the CubeSat reference architecture's structure as an AGSS is discussed in relation to fair fight considerations that impacted the analysis and quantification of the model's fidelity.

System Level Resolution

System level metrics attempt to quantify the difference in aggregation, complexity, and completeness of a model. This is done by examining the existence of real-world entities within the system hierarchy (Roza, 2005). One method to examine the complexity of an executable model is to establish fidelity metrics attributed to its structural decomposition. Metrics that describe decomposition include the models

maximum and average decomposition depth as well as its width and bifurcation. Within (Roza, 2005), evaluator functions exist to compare these traits to a fidelity referent. The development of an all-encompassing fidelity referent within the domain of CubeSats is outside of the scope of this thesis. Instead, the executable model's decomposition traits will be calculated as standalone metrics for its complexity. SysML models, specifically BDD's of system structure, facilitate this type of analysis. Using the CubeSat reference architecture and its structure package, a quick calculation of these metrics can be made. Figure 12 demonstrates the package structure of the CubeSat reference model.

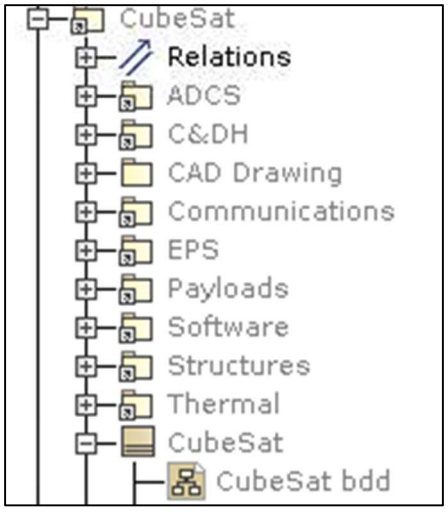

Figure 12: CubeSat Reference Architecture Package Structure

Within each sub-package exists a BDD of a subsystem that resides within the CubeSat BDD. Through analysis of these diagrams, the overall structural properties for the CubeSat executable model can be calculated. These properties are shown in Table 1.

Table 1: Model Structural Properties

Overall Structural Properties	
Total Subsystems	52
Total Leafs	36
Total Forks	18
Maximum Branch Length	5
Average Branch Length	3.154

Pulling the properties from the Table, the analyzed executable model has a maximum decomposition depth of 5, average decomposition depth of 3.154, decomposition width of 36, and decomposition bifurcation of 18. These metrics give a quick look into the scope, limitations, and complexity of the executable model (Roza, 2005). Ratios between these metrics provide further insight such as the max decomposition depth not being significantly larger than the average decomposition depth (Roza, 2005). This means no one subsystem is modeled to much greater detail than the others. While these metrics do not provide any groundbreaking insights into the model, they do provide a beneficial summarization of the model's structure. For the purpose of this thesis, these metrics were calculated manually. However, within the Cameo Systems Modeler tool, methods could be devised to automate the execution of these metrics providing a quick look into the system level structural fidelity of models.

Property Level Resolution

At a level lower than system resolution lies property level metrics. While system level metrics establish what entities exist in the scope of the executable model, property level metrics describe the completeness of these entities. These metrics can be used to

characterize the existence of parameters, inputs, outputs, and state properties (Roza, 2005). Within a Cameo model these metrics would be analyzing the completeness of part properties of blocks in relation to a referent model. A common fidelity referent for this analysis would take the form of spec sheets which correspond to the modeled entity. Within the CubeSat reference architecture, two separate blocks will have their fidelity quantified using property scope difference evaluator functions. A property scope difference is calculated using the following evaluator function provided by (Roza, 2005) and shown in Equation 1.

Equation 1: Property Scope Evaluator Functions (Roza, 2005)

$$EF(R^{Ref}, S^{Appx}) = (\frac{P^{Ref}}{P^{Exec}})$$

$EF =$ Evaluator function
$R^{Ref} =$ Real world reference knowledge specification
$S^{Appx} =$ Appoximated simulated system knowledge specification
$P^{Ref} =$ Referent system property range set
$P^{Exec} =$ Executable system property range set

The demonstration of this property scope quantification shall provide a look into the feasibility of application for this metric and any ambiguity into the information the metrics provides. Within the Cameo model for the CubeSat reference architecture, two blocks exist which model the primary transceiver, Cadet Plus, and back up transceiver, Iridium 9603. The fidelity referent for these transceivers will be their corresponding spec sheets. The executable model and specification sheet for the Cadet Plus Transceiver are shown in Figure 13 and Figure 14.

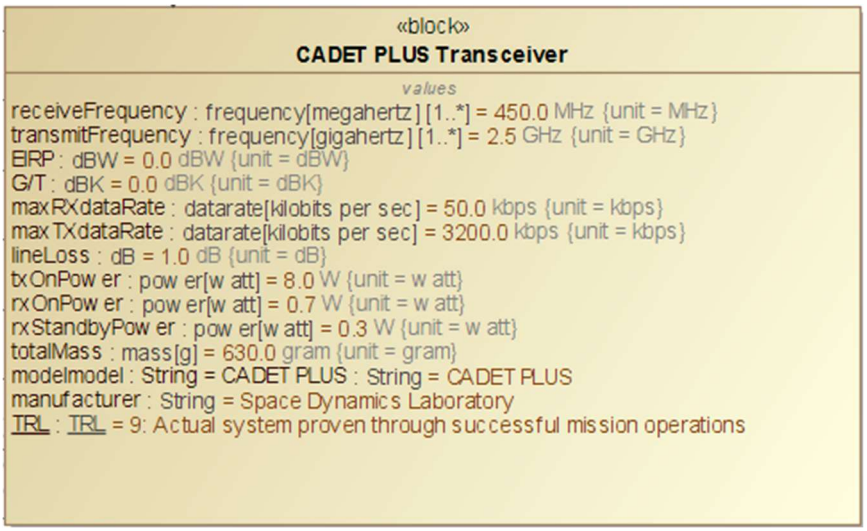

Figure 13: Cadet Plus Block

PARAMETER	Cadet PLUS
DOWNLINK	S-Band: 2.2 - 2.9 GHz; 2 W RF power out; up to 3.2 Mbps; OQPSK
UPLINK	UHF; 449.75 - 450.25 MHz; ≤50 kbps; GFSK
MASS	~630 g
DIMENSIONS	100 x 100 x 28 mm
C&DH INTERFACE	RS-422 & CAN
POWER CONSUMPTION	0.7 W receive, 8 W transmit
DATA LATENCY	Real-time & store-and-forward architecture
GROUND STATION	MC3 (6 CONUS nodes including SDL)

Figure 14: Cadet Plus Specs (Laboratory, Space Dynamics)

Between the Cadet Block and its corresponding spec sheet, a mapping of value properties and traceability from the referent model to the executable model can be established. Within this Cadet block, all properties can be considered parameters. Using the property scope difference evaluator function, the number of parameters in the executable model

can be divided by the number of parameters in fidelity referent. Executing this function computes a property scope fidelity score of 11/16=0.6875. The mapping of value properties between executable and fidelity referent as well as total value properties is demonstrated in Table 2.

Table 2: Cadet Plus model to Specification Comparison

Property	Units	Model	Spec Sheet
Receive Frequency	MHz	450.00	min 449.75
			max 450.25
Receive Phase Shift Keying	Type	N/A	GFSK
Transmit Frequency	GHz	2.50	min 2.2
			max 2.9
Transmit RF Power Out	watt	N/A	2.00
Transmit Phase Shift Keying	Type	N/A	OQPSK
EIRP	dBW	0.00	N/A
G/T	dBK	0.00	N/A
Max RX Data Rate	kbps	50.00	50.00
Max TX Data Rate	kbps	3200.00	3200.00
Line Loss	dB	1.00	N/A
TX On Power	watt	8.00	8.00
RX On Power	watt	0.70	0.70
RX Standby Power	watt	0.30	N/A
Total Mass	g	630.00	630.00
Length	mm	N/A	100.00
Width	mm	N/A	100.00
Depth	mm	N/A	28.00
C&DH Interface	Type	N/A	RS-422 & CAN
Existing Part Properties		11	16

The same method can be applied to the backup transceiver using its own block and spec sheet shown in Figure 15 and Figure 16.

Figure 15: Iridium 9603 Block

MECHANICAL SPECIFICATIONS

Dimensions	31.5 mm X 29.6 mm x 8.1 mm (L x W x H)
Weight	11.4 g

POWER PARAMETERS

Supply Input Voltage Range	5.0V +/- .5V DC
Supply Input Voltage Ripple	< 40mV pp
Idle Current (Peak)	156mA
Idle Current (Avg.)	34mA
Transmission Current (Peak)	1.3 A
Transmission Current (Avg.)	145mA
Receive Current (Peak)	156mA
Receive Current (Avg.)	39mA
SBD Transfer - Avg. Current	158mA
SBD Transfer - Avg. Power	≤ 0.8 W

RF INTERFACES

Frequency Range	1616 to 1626.5 MHz
Duplexing Method	TDD (Time Domain Duplex)
Input/Output Impedance	50Ω
Multiplexing Method	TDMA/FDMA

ENVIRONMENTAL SPECIFICATIONS

Operating Temperature	- 40C to +85C
Operational Humidity	≤ 75% RH
Storage Temperature	- 40C to +85C
Storage Humidity	≤ 93% RH

Figure 16: Iridium 9603 Specification (Iridium Satellite LLC, 2020)

Executing the property scope difference function for the Iridium 9603 transceiver provided us with a property scope fidelity score of 16/26=0.615. The mapping of value properties and subsequent totaling of properties for the executable and referent model are shown in Table 3.

Table 3: Iridium 9603 Model to Specification Comparison

Property	Units	Model	Spec Sheet
Total Mass	g	11.40	11.40
Max Operating Temp	deg C	85.00	85.00
Min Operating Temp	deg C	-40.00	-40.00
Max Frequency	MHz	1626.50	1626.50
Min Frequency	MHz	1616.00	1616.00
Duplexing Method	Type	TDD	TDD
Impedance	Ohm	50.00	50.00
Multiplexing Method	Type	TDMA & FDMA	TDMA & FDMA
Peak Idle Power	watt/mA	0.78	156.00
Average Idle Power	watt/mA	0.17	34.00
Peak TX On Power	watt/mA	6.50	1.30
Average TX On Power	watt/mA	0.725	145.00
Peak RX On Power	watt/mA	0.78	156.00
Average RX On Power	watt/mA	0.195	39.000
SBD TX On Power	watt	0.80	max 0.80
Max Data Rate	kbps	2.40	N/A
Supply Input Voltage Range	V	N/A	Min 4.5
			Max 5.5
Supply Input Voltage Ripple	mV pp	N/A	max 40
SBD Transfer Avg Current	mA	N/A	158.00
Max Operational Humidity	RH	N/A	75%
Max Storage Humidity	RH	N/A	93%
Max Storage Temp	deg C	N/A	85.00
Min Storage Temp	deg C	N/A	-40.00
Length	mm	N/A	31.50
Width	mm	N/A	29.60
Depth	mm	N/A	8.10
Existing Part Properties		16	26

While a fidelity metric for property scope difference has been provided for both transceivers, confounding factors make it both artificially high and reduce its meaning. First, there are properties modeled that are non-existent in a real-world system representation. This means that the fidelity referent is incomplete. In this case, the specification does not capture all knowledge known about the entity. This obscures other missing properties that exist in the fidelity referent but not the executable model. This brings into question whether the corresponding specifications are a complete picture of the body of knowledge on these entities. In application, an aggregation of all known properties would need to be compiled, with part properties detailed that may exist outside of the important specifications the designer had in mind. Secondly, with two transceivers modeled and fidelity scores calculated, a comparison between the two could be made. This comparison would be a futile effort due to large variances in provided properties by specifications and a lack of a standard fidelity referent between the two. In this particular case one could argue that the Iridium 9603 block was actually modeled to a higher degree of fidelity due the inclusion of more value properties, encompassing concepts such as operational temperatures, which the Cadet Plus block was lacking.

In conclusion, property scope difference fidelity quantification, in the manner carried out above, has too much ambiguity to provide meaningful utility within comparison of conceptual models. Specifically, it would provide little insight towards resolving and discovering fair fight violations within a DSS. This ambiguity exists due to the vastness of part properties for a specific entity and the inability to transfer meaning on

what properties are and are not represented through a singular fidelity score. This 0-1 value provides no additional meaning from the initial specification to block mapping and only obscures information. As a concept however, property scope difference may provide some utility towards achieving fair fight within a DSS. Stronger efforts would be required towards development of a fidelity referent for a transceiver and encompassing all critical and known information as applied toward the field of application in this case, CubeSats. Secondly, scope difference may best be left as mapping of properties from referent to executable model. This would provide for traceability between value properties of blocks and a glimpse into what properties are and are not represented. Lastly, while this analysis analyzed the existence all value properties within the block, this method can also be applied to examine just value properties used in the execution of the model. By establishing the scope of properties which are used to execute the behavior of the simulation, a better characterization of the entity's fidelity may be realized.

Real World Behavior Sample Accuracy

The sample pairs accuracy evaluator function captures real world behavior sample accuracy. This is the most common applied method for determining differences between a simulation and reality. This method uses either direct observations of a real-world system or artificial samples generated by other simulations of the same system (Roza, 2005). For demonstration of the application of this fidelity evaluator function a behavior sample analysis on the payload stateflow within the CubeSat reference model shall be completed. Payload state behavior is modeled as on, off, or standby and this state is

determined by the STK access data for the Target, and fault mode status of the CubeSat. As demonstrated in the Figure 17, the Payload is cycling from off to standby to on then back to off every time the target is in view.

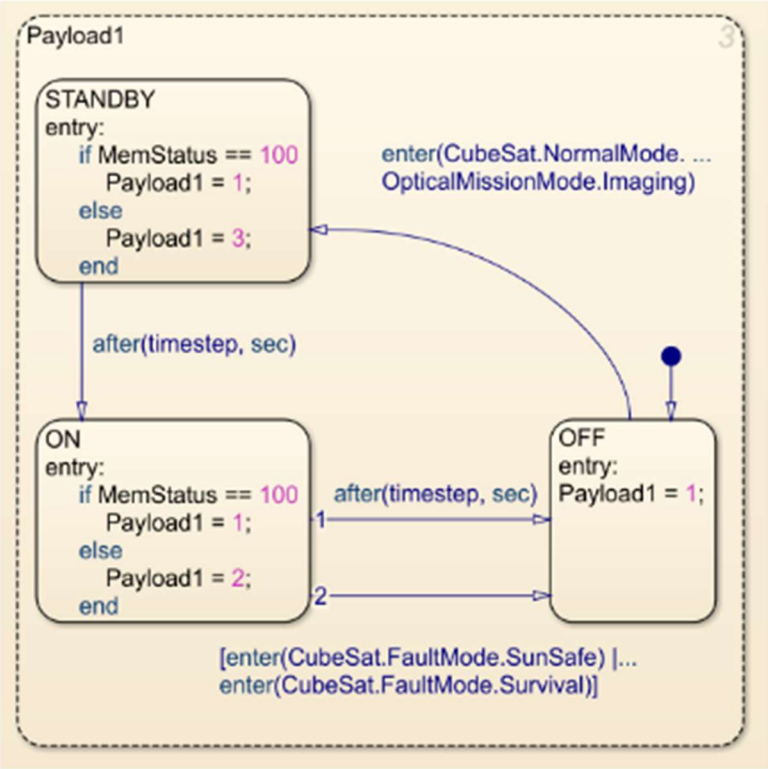

Figure 17: CubeSat Stateflow

This, however, is a low fidelity model that does not accurately represent the reality of state transitions within the Payload. In reality, a payload transitions from off to standby when the CubeSat goes in orbit and stays in standby until the Payload is engaged. It will only transition to an off state when the CubeSat enters a Fault Mode. This accurate representation of reality is shown in Figure 18; however, it has not been successfully simulated within the CubeSat reference architecture.

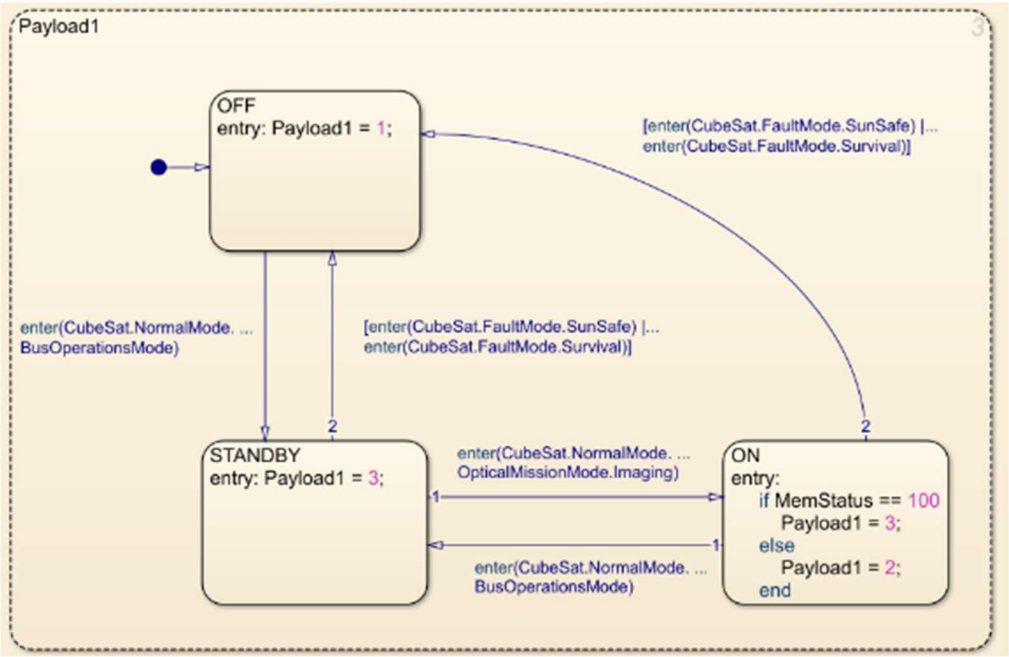

Figure 18: High-Fidelity CubeSat Stateflow

For the purposes of calculating the fidelity of the current Payload state, the realistic Payload state is used as the fidelity referent. This is a means to accommodate a lack of experimental data and behavior samples from an actual CubeSat. Ideally, the fidelity referent for this application would be experimental data of state transitions for the Payload as well as corresponding power draws associated with that Payload over time. To implement the referent state behavior the in-view status is first extracted from the STK outputs. A comparator is applied through a MATLAB function that extracts the indices of the simulation time vector for when the ground sites are in-view to the CubeSat. This process generates a new state array that was inserted into the appropriate Simulink component in order to produce the referent state behavior.

When comparing behavior samples, for both experimental and simulated data, similar initial conditions are required. The expressions in Equation 2, taken from (Roza, 2005) quantify the similarity in initial conditions for behavior samples.

Equation 2: Initial Conditions Sample Trajectory Evaluator Functions (Roza, 2005)

$$EF(R^{Ref}, S^{Appx}) = \{(q^{ref}(t_1) - q^{exec}(t_1) \dots (q^{ref}(t_1) - q^{exec}(t_1)\}$$

$$EF(R^{Ref}, S^{Appx}) = \{f_{diff}(w^{ref}, w^{exec})_1 \dots f_{diff}(w^{ref} - w^{exec})_n\}$$

$EF =$ Evaluator function
$R^{Ref} =$ Real world reference knowledge specification
$S^{Appx} =$ Appoximated simulated system knowledge specification
$q^{Ref} =$ Single element of referent system state variable set
$q^{Exec} =$ Single element of executable system state variable set
$t_1 =$ Initial time
$f_{diff} =$ Trajectory differnece function
$w^{ref} =$ Referent system input segment
$w^{exec} =$ Executable system input segment

In simple terms and related to this application, the first equation calculates the difference in initial conditions for the payload. For this analysis, these value properties are the Payload's power while on, off, and at standby. For optimal comparison these values should be the same between simulated and referent samples resulting in the output of the equation being zero or as small as possible. The second equation quantifies the difference between input vectors, with data such as the bus operation mode, fault status, and CubeSat location all ideally being equal. The results of behavior sample analysis are generated in the following two expressions shown in Equation 3, also taken from (Roza, 2005).

Equation 3: Behavior Sample Trajectory Evaluator Functions (Roza, 2005)

$$EF(R^{Ref}, S^{Appx}) = \{f_{diff}(q^{ref}, q^{exec})_1 \dots f_{diff}(q^{ref} - q^{exec})_n\}$$

$$EF(R^{Ref}, S^{Appx}) = \{f_{diff}(o^{ref}, o^{exec})_1 \dots f_{diff}(o^{ref} - 0^{exec})_n\}$$

$EF =$ Evaluator function
$R^{Ref} =$ Real world reference knowledge specification
$S^{Appx} =$ Appoximated simulated system knowledge specification
$f_{diff} =$ Trajectory differnece function
$q^{Ref} =$ Single element of referent system state variable set
$q^{Exec} =$ Single element of executable system state variable set
$o^{ref} =$ Referent system output segment
$o^{exec} =$ Executable system output segment

The first expression is the difference in behavior trajectory of the system states which for this application is when the payload is on, off, and standby, while the second one is the difference in system output which is the power draw at each instance in time. This difference is inputted into any sort of trajectory functions to produce a singular real value that quantifies the fidelity of the executable model's sample.

The resultant difference between power vectors are plotted in Figure 19, Figure 20, and Figure 21 to show how the simulation system outputs are changed by introducing a higher fidelity state transition function. Values below zero are power consumption unaccounted for by the lower fidelity version of these models.

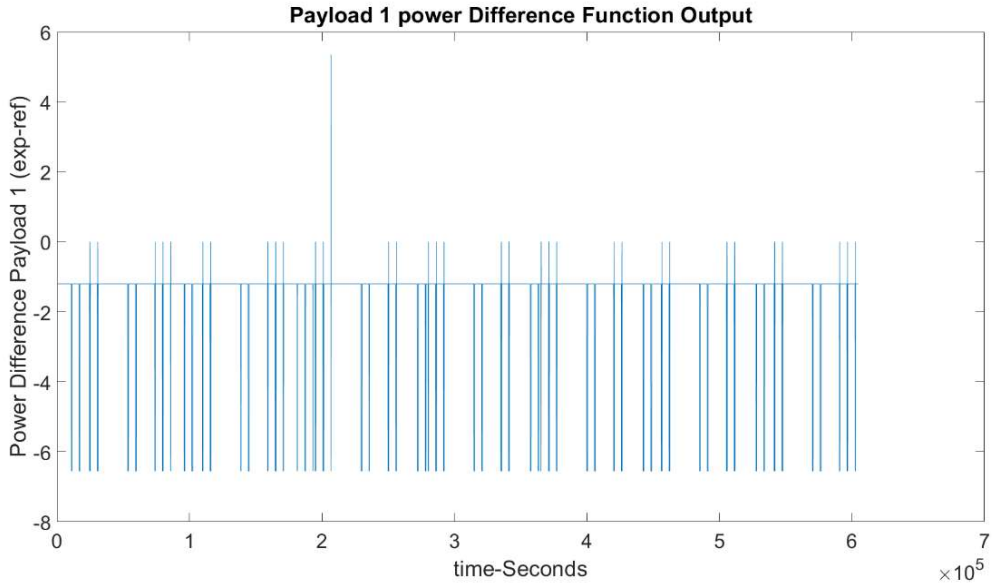

Figure 19: Payload 1 Power Difference

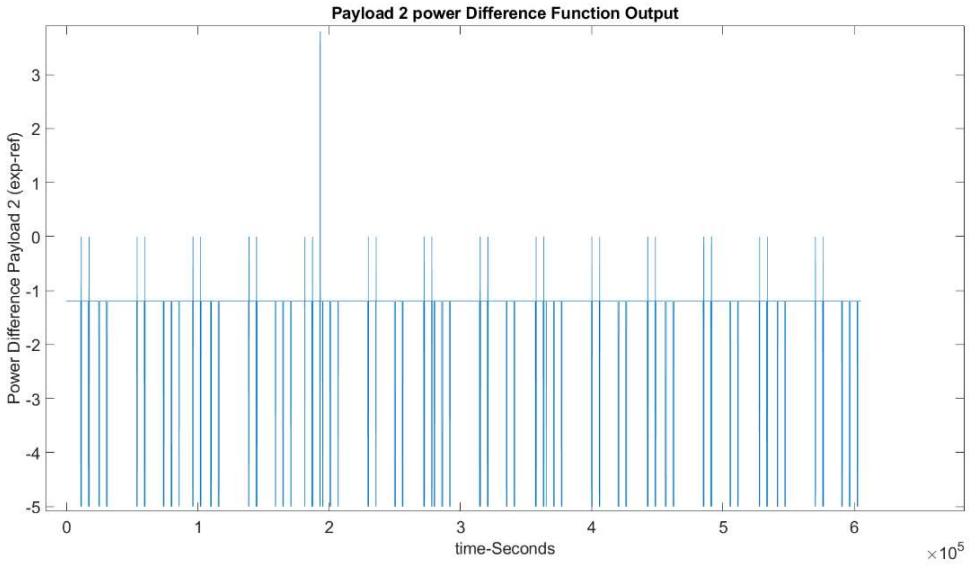

Figure 20: Payload 2 Power Difference

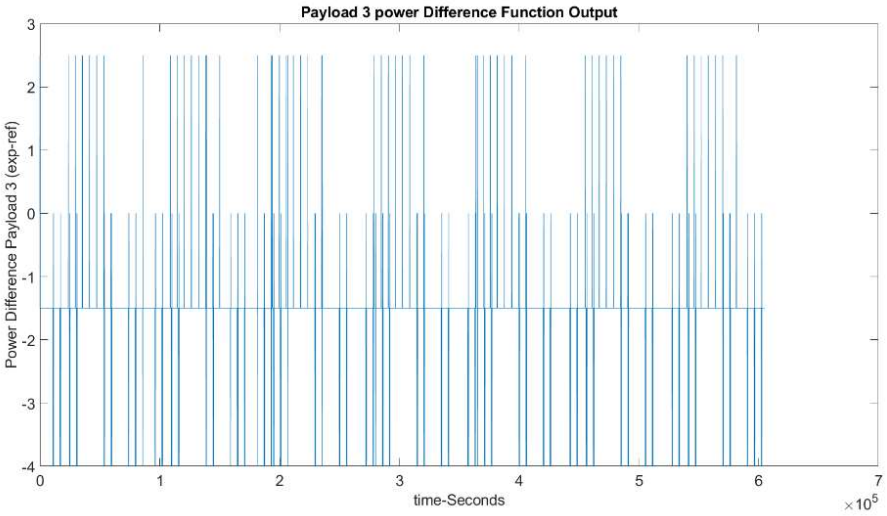

Figure 21: Payload 3 Power Difference

By taking the mean power shown in Table 4 over the simulated 1-week time period from the application of the difference functions a total unaccounted power can be determined to be approximately 741 Watt-hours.

Table 4: Descriptive Statistics Behavior Sample Trajectory Difference

Descriptive Statistics of Difference Function Outputs	Median	Mean	Minimum	Maximum
Payload 1	-1.20 W	-1.46 W	-6.55 W	5.35 W
Payload 2	-1.20 W	-1.38 W	-5.00 W	3.80 W
Payload 3	-1.50 W	-1.57 W	-4.00 W	2.50 W
Depth of Discharge	-2.18 %	-2.37 %	-8.01 %	0.00 %

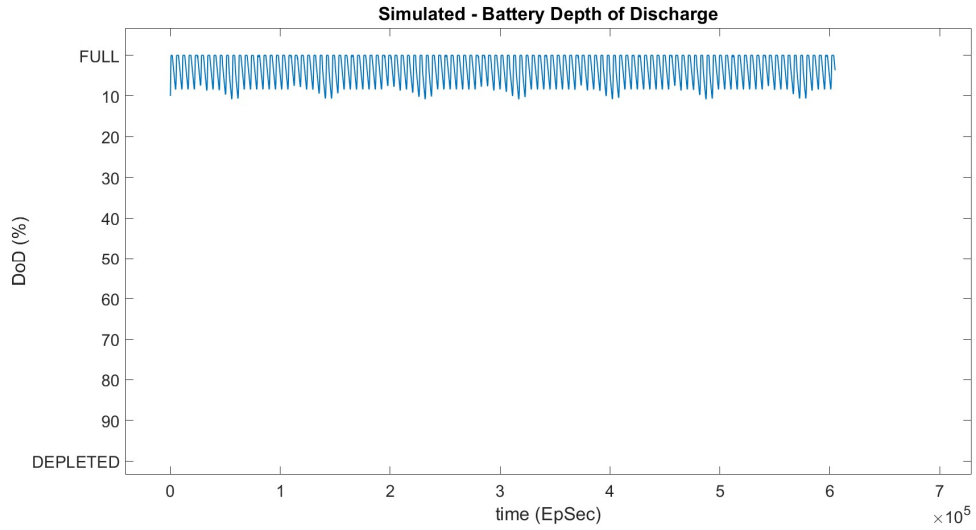

Figure 22: Simulated Battery Depth of Discharge

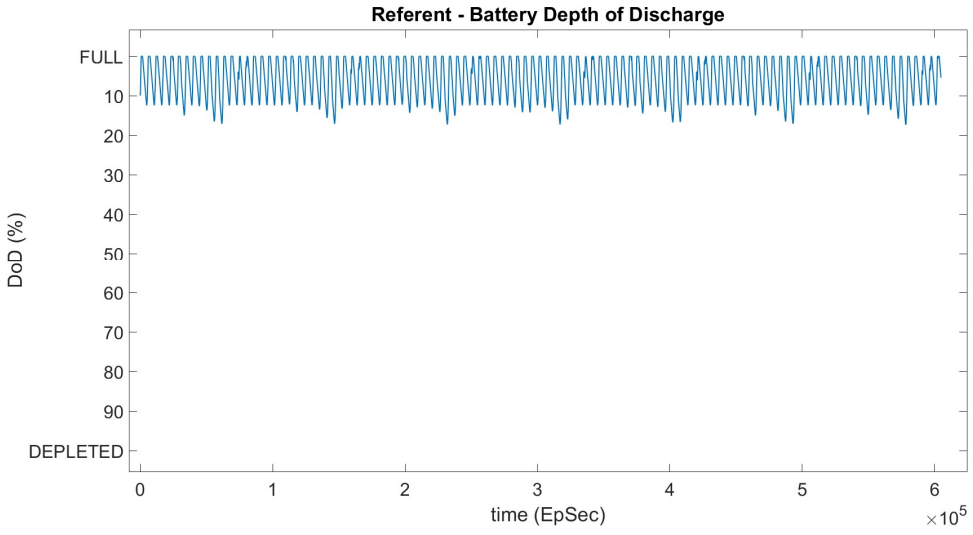

Figure 23: Referent Battery Depth of Discharge

In order to achieve a quantitative comparison between the referent and simulation behaviors, the following trajectory function of the integrated absolute error, Equation 4, was applied to the cubesat payload data.

Equation 4: Integrated Absolute Error Trajectory Function (Roza, 2005)

$$f_{diff}^{abs}(Z_{ref}, Z_{exec}) = \int_{t_2}^{t_1} |Z_{ref}(t) - Z_{exec}(t)| \cdot dt$$

f_{diff}^{abs} = Absolute trajectory function
Z_{ref} = Reference trajectory
Z_{exec} = Simulated trajectory
t_1, t_2 = Initial time, Final time

The results of this trajectory functions are shown in Table 5 for each of the three Payloads and the Battery Depth of Discharge.

Table 5: Integrated Absolute Errors of Payloads and Depth of Discharge

	Payload 1	Payload 2	Payload 3	Battery Depth of Discharge
absolute error	14733.39	13953.8	16097.5	23890.03

A time weighted version of these results is then calculated using Equation 5.

Equation 5: Time Weighted Integrated Absolute Error (Roza, 2005)

$$(d_1)^{-1} = \frac{f_{diff}^{abs}(Z_{ref}, Z_{exec})}{(f_{diff}^{abs}(Z_{ref}, Z_{exec}) + \frac{1}{\int_{t_1}^{t_2} |Z_{ref}(t)| \cdot dt}}$$

$(d_1)^{-1}$ = Proportional inverse scaled equivalent of Equation 4
f_{diff}^{abs} = Absolute trajectory function
Z_{ref} = Reference trajectory
Z_{exec} = Simulated trajectory
t_1, t_2 = Initial time, Final time

The results of the inverse time weighted trajectory function are shown in Table 6.

Table 6: Time Weighted Errors of Payloads and Battery Depth of Discharge

	Payload 1	Payload 2	Payload 3	Battery Depth of Discharge
Simulated trajectory accuracy	$4.53*10^{-9}$	$5.07*10^{-9}$	$3.77*10^{-10}$	$7.21*10^{-10}$

The time weighted integrated absolute error trajectory function yields a result between zero and one where the closer the resultant value is to zero the less accurately the modeled trajectory matches the externally generated referent trajectory. The resultant fidelity scores for each payload and the Battery Depth of Discharge as shown in Table 6 are very close to zero demonstrated their low fidelity. Even still, little information on the fidelity of the simulation over time can be gained by this singular value. The following

section will examine a piecewise version of the behavior sample trajectory function to provide a clearer picture on the fidelity of the executable simulation over time.

The CubeSat Simulation being examined takes place over a 1-week orbital period (Kelly, 2021). This orbital period covers many full orbits around the Earth and interactions with ground stations based on the distance from the CubeSat to the ground location. The orbital periods are a driver for cyclical behaviors of power generation, while the CubeSat proximity to ground locations are drivers for payload state behavior. Due to the cyclical behavior of the CubeSat payload's power consumption and generation, a piecewise analysis of the simulated trajectory function, Figure 24, was performed to show a greater temporal resolution of behavior.

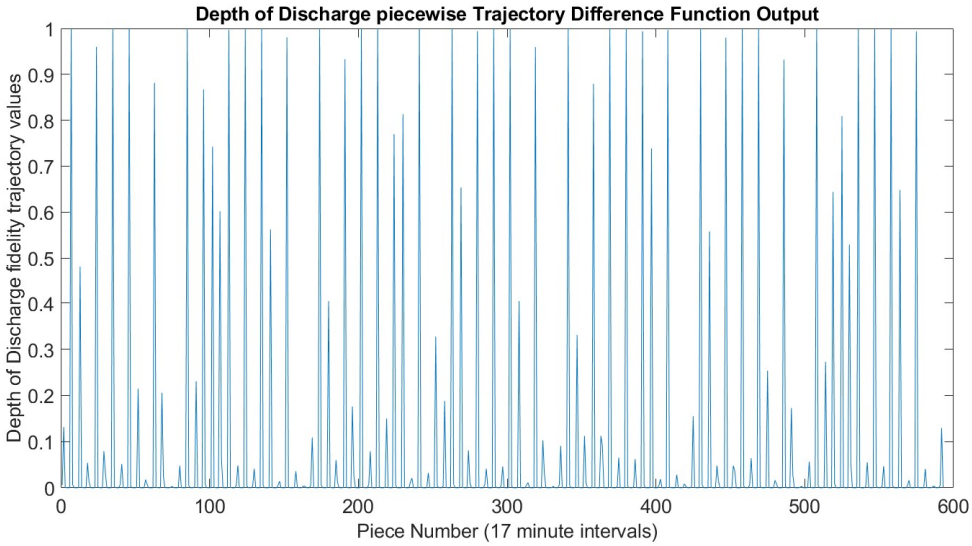

Figure 24: DoD Piecewise Trajectory Difference Function Output

This piecewise execution of the simulated trajectory function shows the variety of fidelity to the referent behavior occurring over the duration of the simulation. This representation

captures both the results of the simulation trajectory function and the time series correspondences. Values of 1.0 indicate an exact match of the 17 values between the simulation and referent data sets corresponding to the piece number. While lower values indicate a proportionally large difference of simulation and referent values corresponding to that piece. In order to capture the distribution of these trajectory values, a histogram was generated in Figure 25.

Each of the simulated CubeSat payloads power consumption were also examined using the simulated trajectory function. The results can be found in Figure 26-31. It is noteworthy that the results of performing this piecewise analysis were much less varied than the results of the Depth of Discharge simulation.

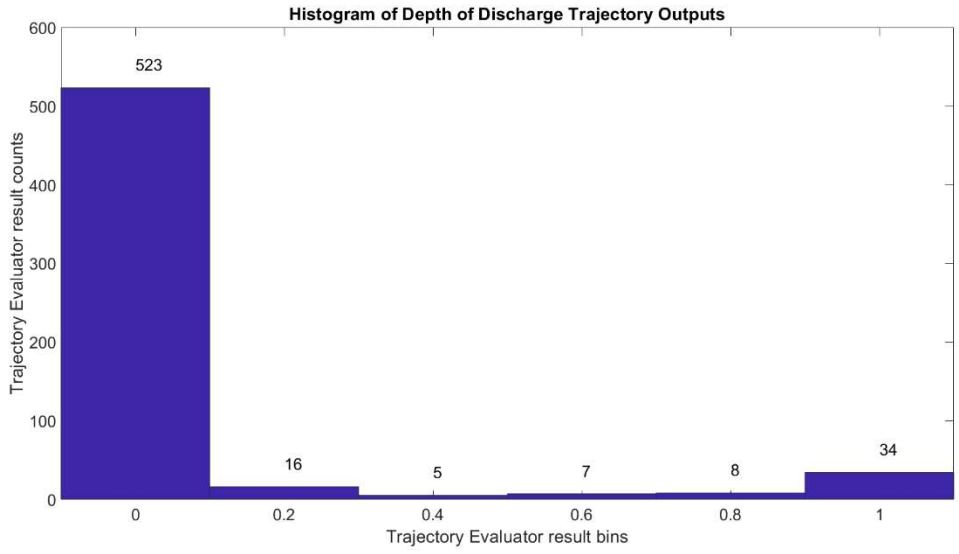

Figure 25: Histogram of DoD Discharge Trajectory Outputs

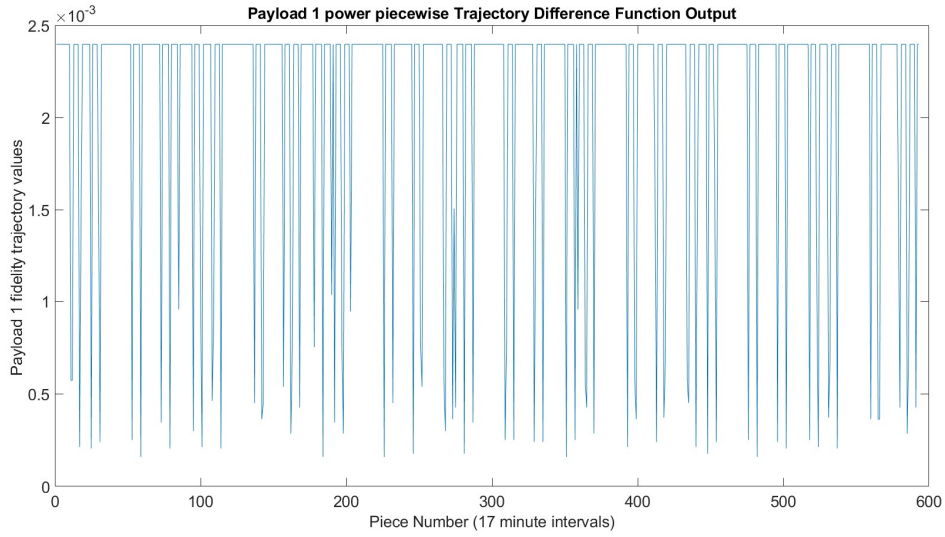

Figure 26: Payload 1 Piecewise Trajectory Function Output

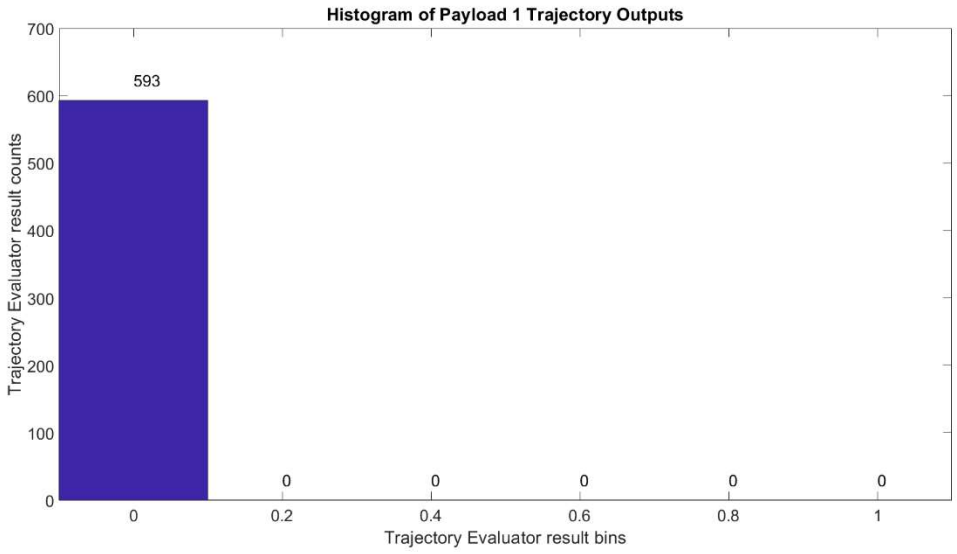

Figure 27: Histogram of Payload 1 Trajectory Outputs

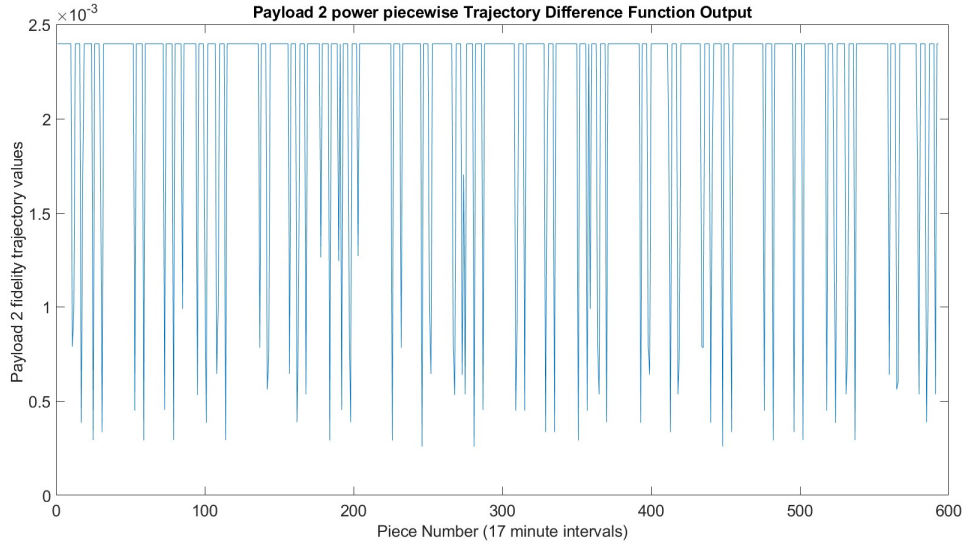

Figure 28: Payload 2 Piecewise Trajectory Function Output

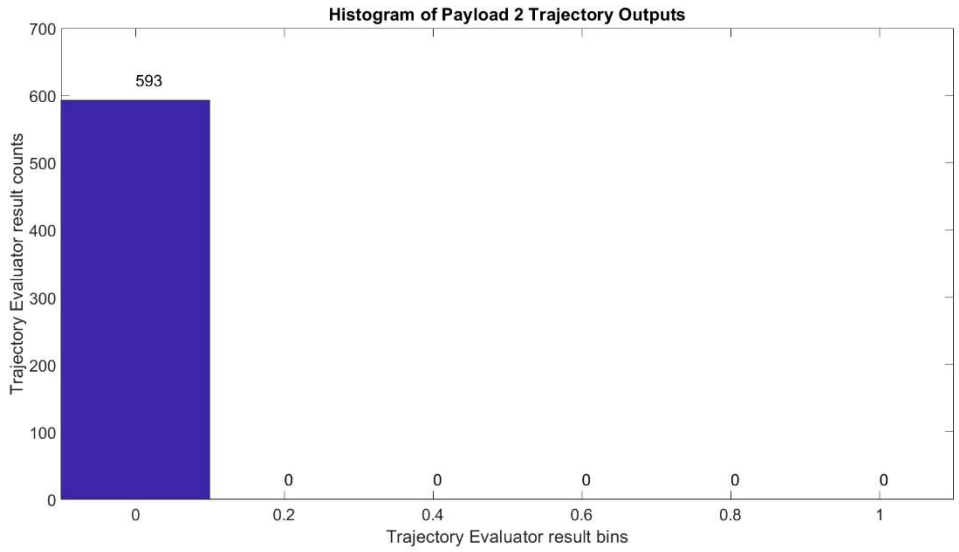

Figure 29: Payload 2 Piecewise Trajectory Function Output

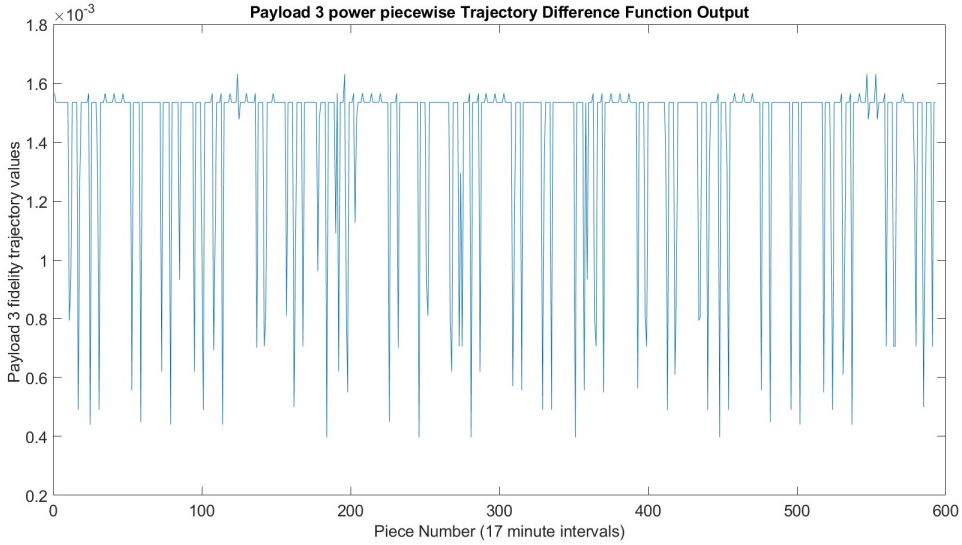

Figure 30: Payload 3 Piecewise Trajectory Function Output

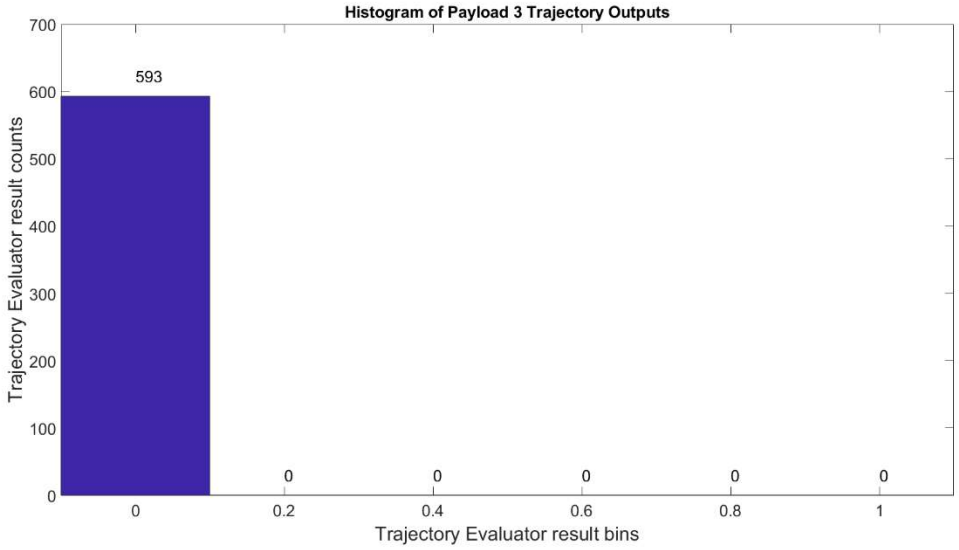

Figure 31: Payload 3 Piecewise Trajectory Function Output

Fair Fight within the CubeSat AGSS

Throughout the process of establishing fidelity metrics to describe the state transitions of the CubeSat model, external elements outside of just the fidelity of the behavior were discovered that impacted the results produced for payload power and battery depth of discharge. By analyzing the structure of the AGSS and associated data flows shown in Appendix 2, fair fight violations were observed at a dynamic interoperability level due to the real-time asynchronicity for phases 3) and 4). In practical terms the fair fight violation means that any fault states triggered in phase 4) cannot have any affect in phase 3); due to the phase 3) simulation already having been executed before phase 4) begins. This leads to fidelity considerations along the lines of timing and behavior. The data are generated at separate times in 3) and 4) and later associated using the simulation data time vector. Regarding behavioral fidelity this implementation would not allow for feedbacks between the positional behaviors generated in the STK simulation 3) and the CubeSat systems behavior generated in the Simulink simulation 4). With systems on the CubeSat that affect and maintain attitude it is reasonable that fault states or battery depletion could have feedbacks with positional data which is generated previously in phase 3). The important factors to understand from these observations of fair fight and fidelity considerations comes back to the intended use of this AGSS. These fair fight violations leading to a reduced fidelity of timing and behavioral activities imply prototyping and design decisions that relate to managing possible error or emergency states cannot be verified using this model.

It must be noted that the original intended purpose of the CubeSat reference architecture is to validate variable payloads riding on the CubeSat Bus. Therefore, it is worth noting that this intended scope was fully achieved by the as-is CubeSat reference architecture model because the nature of payloads tested up to this point are passive in mechanical terms. None of the fair-fight violations cited here would impact the ability to test loads on the power storage or data transfer to and from the CubeSat. However, payloads that are mechanically active, such as, propulsion would require that these fair fight and interoperability concerns be addressed in order to produce valid simulation results. Ultimately this analysis has uncovered what would be required to expand the scope of the CubeSat reference architecture to include payloads or fault state behavior which impact position data generated through STK. Analysis such as this points to the fact that fair fight analysis may not always be conducted on simulation services that are operating under their intended purpose. One aspect of DSS is the ability to reuse legacy systems and integrate them with future capabilities. By understanding the limitations of legacy systems outside of their intended application an assessment of possible fair fight violations if integrated with other simulation services can be conducted.

Summary

This section detailed the process, results, and analysis of applying fidelity evaluator functions to the CubeSat reference architecture model. This section produced metrics for system level properties which quantified the aggregation, complexity, and completeness of the CubeSat model. Also, a primary and backup transceiver had their

property level scope quantified in order to establish a comparison between the fidelity of the two. It was determined that tracing a block's part properties back to referent data such as a spec sheet provides more utility for comparison rather than a standalone fidelity metric of 0-1. Lastly, the model's state change behavior had its fidelity quantified through the use of a high-fidelity external simulation. By extracting information from the CubeSat model and generating a new simulation using MATLAB, analysis of the fidelity of payload power and battery depth of discharge could be quantified. Using trajectory difference functions, time weighting, and piecewise analysis of the data, internal and externally produced behavior samples from the CubeSat model could have the fidelity conceptualized. This allowed for further analysis of fair fight issues that impacted these results.

V. Conclusions and Recommendations

Overview

The following section will describe in detail how each of the three established research questions were answered and supported by efforts within the literature review, methodology, and analysis sections. Next, study limitations will be described, as well as recommendations for future research within the field of fidelity to further support the achievement of fair fight with DSS. Lastly, the significance of this research will be summarized.

Conclusions to Research Question 1

1. *What are the fair fight considerations for a distributed simulation system and how can they be organized and categorized?*

Throughout the literature review process, it was determined that a common framework for organizing and categorizing fair fight issues and consideration did not exist. Efforts had been completed by (Siegfried, Luthi, Herrmann, & Hahn, 2011) in which the fair fight issues were related back to the LCIM. Further efforts were taken by (Task Group MSG-086, 2015) in which issues with simulation interoperability were categorized based on issues including, lack of conceptual model, federation development, fidelity, infrastructure, LVC coupling, organizational and legal, scenarios, environment, and time. Each of the specific issues within these categories were then further related back to the LCIM. While not a specific categorization of fair fight issues, many of the interoperability issues examined were due to fair fight considerations. By examining this

list and other sources from the literature review, a comprehensive categorization of fair fight consideration was established. This categorization took the form of functional requirements with fair fight issues occurring when these requirements are not met. These functional requirements took the form of interoperability requirements at the implementation level and fidelity requirements at the modeling level. These requirements further decomposed into each of the levels of conceptual interoperability in the LCIM and the fidelity of the model's structure, behavior, and timing mechanisms. Through this requirements decomposition, fair fight issues can be categorized and traced back to their root cause.

Conclusions to Research Question 2

2. *How does one ensure that fair fight considerations are built into the specifications and requirements of distributed simulation systems?*

Ensuring that fair fight considerations are built into specification and requirements of a DSS begins by defining the functional requirements for the system. By establishing these requirements this thesis now provides a means on which lower-level non-functional requirements can be described for the system. This will take the form of interoperability requirements based on specific simulation environments, architecture, networks, and simulation tools. The development of these types of non-functional requirements was outside the scope of this thesis but can be facilitated by the functional requirements produced. Other lower-level requirements need to be established that deal with the alignment of fidelity between two conceptual/executable models. In order to write

requirements that can be validated and satisfied, a method to quantify the fidelity of these model needs to be established. This thesis presented three fidelity metrics that quantify the behavior of models and structure at the system and property level. Two of these provided promise in facilitating the quantification of models while also having capabilities of being automated within SysML tool. By providing a means and potential tool automation for assessing fidelity, fair fight specification with relation to alignment of fidelity issues can more readily built into system specifications.

Conclusions to Research Question 3

3. *How can fair fight of an as-is distributed simulation system be assessed, either during setup/configuration or throughout its verification and validation process?*

While the quantification of fidelity was initially driven by the ability to compare conceptual/executable models, it has further utility after DSS configuration and during the verification and validation process. The ability to quantify the fidelity of models has not progressed to the point in which a comparison of fidelity metrics could provide definitive proof on whether a simulation between the two models will result in fair fight. The development and application of the fidelity metrics did make progress towards this end goal. This was seen by the demonstration that a low fidelity model of state transitions of a CubeSat can cause downstream effects on battery discharge throughout a simulation. Generalizing on this demonstration indicates that the conceptual framework of fair fight and the quantitative analysis techniques applied through the fidelity evaluator functions are useful tools for assessment of simulation systems. Describing the flow of data from

one simulator to another with respect to the flow of time, variables, and events as described in the appendix enables consideration of fair fight.

In this specific example, examination showed potential fair fight violations between the models of payloads and behavior held in Cameo, MATLAB, and STK. These potential violations exist because variables containing the information that instantiates the payload models can be modified in the other simulators. If such a modification occurs, then the models between the simulators are no longer consistent. Furthermore, this mapping prompted consideration that uncovered another fair fight issue between the MATLAB and STK arising from STK executing the position based on the orbit for the entire simulation experiment. The problem in this case is that there is no feedback mechanism created from MATLAB to STK that would alter orbital behavior if a fault state occurred affecting orbital parameters. Therefor any changes that occur with respect to ground sites or solar array power generation cannot be captured by the as-is AGSS. Because fair fight deals with simulation systems at a conceptual level these techniques help to explore an as-is simulation system from this perspective. Applying the fidelity evaluator operations while considering application member models from the perspective of fair fight offers a structured approach to as-is DSS analysis. These concepts can be applied to a DSS environment where fidelity levels of one model will impact the outcome of a simulation by another model. By quantifying the fidelity of each element within the model, these potential issues can be detected, and proper fidelity can be built into the model during its setup and configuration.

Study Limitations

A key limitation to this study was a lack of access to an operational Distributed Simulation System and executable simulations within the focused domain. This is a common limitation for academic research when dealing with operational systems. DSS within the DoD have simulations services dispersed throughout organizations and stringent security controls. Working through these limitations still brought contributions to the field of fair fight analysis for DSS. By positing that aggregated simulation systems (AGSS) are a subset of Distributed Simulation Systems (DSS), future research can be accomplished in environments composed of various simulators on one or more processors. This will reduce research costs and increase ease of access to simulation systems that can be analyzed.

Recommendations for Future Research

Within the DoD and broader modeling and simulation community, a general need exists for continued research of fidelity concepts. This includes further examination of a fidelity framework and its role in establishing model and simulation credibility. Aspects of this research could include how fidelity is associated with V & V and what role it plays in the accreditation process. Secondly, research should proceed into how the concept of fidelity can help the DoD define the necessary detail and resolution that must be built into its simulations to achieve their purpose. This is a vast and extensively complex field of research that cannot be achieved by one individual or researcher.

Smaller and more actionable research needs within the field were discovered throughout the development of this thesis and they are described below.

Further research efforts are needed on the quantification of fidelity and means to establish meaningful metrics that can be applied to a model. This thesis utilized metrics proposed by (Roza), however more effort is required in the establishment of standard methods and techniques to quantify fidelity. Research could include a literature review of suitable metrics and an application of metrics toward models in a variety of domains.

Additionally, actionable areas of research include developing an automated means to quantify the structural properties of models within SysML. This would mean automatically generating information on total subsystems, leaves, forks, and maximum and minimum branch length. This information could be shown within a package diagram or within blocks allowing a quick assessment into the level of decomposition the model contains. Other research threads include providing greater traceability between part properties and the documents of the specifications from which they came. This thesis attempted to score the inclusion of properties between a block and its specification. While the metric produced provided little insight, the concept of tracing properties back to the referent data they were sourced from and automatically tying part properties to their source within Cameo should be further examined.

Research could also to take place on applying a confidence level to fidelity metrics. Fidelity metrics get evaluated after assessing the degree to which a model or simulation represents reality. Just as important is having an estimate of the uncertainty of

the capabilities in measuring the fidelity (Z.C. Roza, 2000). This uncertainty is due to a referent having various degrees of sophistication depending on the domain and application. Research efforts in the field of verification validation and uncertainty quantification (VVUQ) (Committee on Mathematical Foundations of Verification, Validation, and Uncertainty Quantification, 2012) provides an avenue to begin this research.

The last component of research that must take place is the creation of referents for specific domains and applications. Referents can take many forms and each domain and application will have their own knowledge source. Advances in big data capture and analytic tools such as QRIP, RAPIDS, and KM (Malloy) provide potential sources for acquiring data for the referent. The acquisition and analysis of big data can help inform and standardize a source to define fidelity for models and simulation in a domain. The combination of these research avenues described will help move the application of fidelity within the modeling and simulation community towards the point in which fidelity requirements can be written which will help facilitate the achievement of fair fight within DSS.

Summary or Significance of Research

Emerging DoD technology, as well as the push toward digital engineering and MBSE have created an environment in which DSS will have increased importance in how operational and developmental testing is conducted in the future. It is critical that the results produced through these simulations are credible and decision makers have

confidence in their results. A critical concept in this discussion of credibility is fair fight. This research established that future discussions on achieving fair fight between simulation services can be broken into components of interoperability between systems and compatible fidelities of their models. The research then made an important first step towards the quantification of fidelity by examining it within a simulation composed of a SysML/MBSE tool (Cameo Systems Modeler), AGI Simulation Tool Kit (STK), and MATLAB. By applying a variety of evaluator functions to different structural and behavioral aspects of a model, an analysis of the utility of the fidelity metrics was performed which can inform further research efforts. This, in conjunction with the hypothesis that an AGSS can be used as a research test bed for a DSS, should facilitate further research towards achieving credible simulations.

APPENDICES

Appendix 1: Fair Fight Issue Categorization

Technical	Syntactic	Semantic
Data Overflow	Missing Standards for Application Services	Different FOM or SDEM Versions
Missing Proper Network Configuration	Multi-Architecture Simulation Environments	Incomplete Specification of Federation Agreements
Temporal Anomalies Caused by Unsynchronized Time	Different HLA Versions	Missing Comprehensive Reference Architectures
Temporal Anomalies Caused by Network Latency	Incompatible FOM Modules	Lack of Agreed Classifications for Data Fidelity Levels
Different bandwidth in communication of simulation data: One simulation system may lose data packets with simulation data due to low communication bandwidth. Consequently, the resulting incomplete information about other entities or objects may lead to a disadvantage and thus to unfair fight.	Inconsistent Data Marshalling	Limitations Due to Integration of Live Systems
		Use of Different Formats for Executable Scenarios
		Difficult to Exchange Synthetic Environments Before Runtime

Pragmatic	Dynamic	Conceptual
Lack of Formalized Description for Entity Aggregations	Lack of Formalized Transfer Mediation Functions Between Incompatible Entity Resolution and/or Data Fidelity Levels	No Explicit Development of a Conceptual Model
Lack of Agreed Levels for Entity Resolution	Difficult to Exchange Synthetic Environment Updates at Runtime	Missing Standards for Development of Conceptual Models for DistributedSimulation Environments
Inconsistent Human Machine Interfaces		Missing Reference Conceptual Models
		Lack of Standard Methodologies, Techniques and Tools for AutomatedTransition from Conceptual Models to Simulation Data Exchange Models (SDEMs)or (Automated) Comparison and Verification Between Conceptual Models and SDEMs
		Missing Standards and Templates for Artifacts
		Lack of Agreed Critical Behaviours and Corresponding Algorithms

Behavior	Timing	Structure
Lack of Agreed Classifications for Data Fidelity Levels	Difficult to Exchange Synthetic Environment Updates at Runtime	Lack of Formalized Description for Entity Aggregations
Lack of Agreed Critical Behaviours and Corresponding Algorithms	Temporal Anomalies Caused by Differences in Precision of Time Representation	Lack of Agreed Levels for Entity Resolution
Difference in Accuracy for Position	Temporal Anomalies Caused by Differences in Time Resolution	Missing Authoritative Operational Scenarios
Use of Different Doctrines and ROEs	Time management inconsistencies: for example, a tank in one simulation system is always shown an airplane at an “old” position, which makes aiming and firing at that plane virtually impossible.	Incomplete or Inconsistent Operational Scenario Description
Synthetic Environment Data is Not Correlated		Missing Formal Scenario Specification
Different Levels of Fidelity are Used for Synthetic Environment		Missing or Incomplete Definition of Application Domain
Different definitions of capabilities for entities: for example, in one simulation, a tank may be able to pass narrow points with a width of 6m, whereas in another simulation system an entity of the same type may not pass the same bottleneck		Different environment representation: consider the situation where a river is represented only in one simulation participant preventing it from moving forward, whereas in another simulation system the same position can easily be passed.
Different computation of visibilities: for two entities in two different simulation systems that should be visible to each other, entity A sees entity B, but not vice versa.		Different object representation: for example, a house may not be represented in one simulation system allowing a tank to continue driving “through” that house as it is represented in another simulation system.
Different weapon effect computation: the same hit by the same ammunition at the same target causes different effects in two simulation systems, because different algorithms to calculate the weapon effects are used.		

Appendix 2: Description of real and simulated data flow within AGSS

Phases of AGSS execution	Passage of data (real time) 1) CSM 2) MATLAB 3) STK 4) Simulink 5) MATLAB.
1)	Creation of initial conditions and state-conditions for payloads, simulation starting time-date, instantiation values for other simulators, longitude-latitude location values of ground sites, activity parameters for CubeSat equipment and payloads Power(consumption+generation, data, effector parameters(momentum etc.)) (t=0 to t=0 [no simulation time passes], Initialization of Matlab ends 1))
2)	MATLAB – Receives parameters from CSM and creates variables to hold and represent that information, Creates the experimental time variables, Creates variables to represent total data transfer rates, battery storage capacity, initialize connections to STK and pass initial conditions from MATLAB workspace to STK(t=0 to t=0[no simulation time passes], initialization of STK ends 2))
3)	STK – Executes orbital simulation to generate CubeSat positional/orientation data with respect to: earth's surface, and the orbital bodies[sun, moon], as well as the ephemeris time series data, generates data files in containing folder on current hard drive of the computer running the AGSS. (t=0 to t=604800 (ephemeris time seconds(ET s) at

	steps of 60 ET s per data point), This step ends after MATLAB imports the STK data generated into its workspace]
4)	MATLAB calls the Simulink PAT.slx (Payload Analysis Tool). Simulink receives variables from the MATLAB Workspace and simulates the behavior of the CubeSat subsystems along the lines of timing, data, and electrical power. The PAT utilizes various state machines, and simulated digital logic elements to generate the power, data, state, and fault status behavior of the CubeSat system over the course of the simulation ([t=0 to t=604800 (ephemeris time seconds(ET s) at steps of 60 ET s per data point), the exporting of the data variable named Telem to the MATLAB workspace ends 4), It is essential to note that while the simulation time is the same time period this occurs AFTER the STK simulation in real-time)
5)	MATLAB utilizes the Telem data to generate reports for the end user of the AGSS.

Appendix 3: MATLAB Code for Real World Behavior Sample Analysis

```
% initial conditions
%   In-orbit            - True
%   Payload State       - Variable
%   Starting Location   - identical


In_Orbit= 1;
AS_IS_Payload_IC= [AS_IS_Payload1States(1),AS_IS_Payload2States(1),...
    AS_IS_Payload3States(1)];
 Starting_timedate='11 Jul 2023 12:00:00.000';

In_view_Payload_IC=
[In_View_AS_IS_W_Payload1Power(1),In_View_AS_IS_W_Payload2Power(1),...
   In_View_AS_IS_W_Payload3Power(1)];

Difference_vector=[In_Orbit-In_Orbit,AS_IS_Payload_IC-In_view_Payload_IC];

normalizedDif=Difference_vector/sum(Difference_vector)


PL1_Traj_Difference = AS_IS_Payload1Power-In_View_AS_IS_W_Payload1Power;
PL2_Traj_Difference = AS_IS_Payload2Power-In_View_AS_IS_W_Payload2Power;
PL3_Traj_Difference = AS_IS_Payload3Power-In_View_AS_IS_W_Payload3Power;
DoD_Difference=AS_IS_DoD-DoD_Referent;

figure(1);plot(t,PL1_Traj_Difference);set(gca,'FontSize',20);
title('Payload 1 power Difference Function Output','FontSize',20);
xlabel('time-Seconds','FontSize',20);ylabel('Power Difference Payload 1 (exp-
ref)','FontSize',20);
PL1_Traj_Descriptives=[median(PL1_Traj_Difference),...
    mean(PL1_Traj_Difference),min(PL1_Traj_Difference),...
    max(PL1_Traj_Difference)];


figure(2);plot(t,PL2_Traj_Difference);set(gca,'FontSize',20);
title('Payload 2 power Difference Function Output','FontSize',20);
xlabel('time-Seconds','FontSize',20);ylabel('Power Difference Payload 2 (exp-
ref)','FontSize',20);
PL2_Traj_Descriptives=[median(PL2_Traj_Difference),...
    mean(PL2_Traj_Difference),min(PL2_Traj_Difference),...
    max(PL2_Traj_Difference)];

figure(3);plot(t,PL3_Traj_Difference);set(gca,'FontSize',20);
title('Payload 3 power Difference Function Output','FontSize',20);
xlabel('time-Seconds','FontSize',20);ylabel('Power Difference Payload 3 (exp-
ref)','FontSize',20);
PL3_Traj_Descriptives=[median(PL3_Traj_Difference),...
    mean(PL3_Traj_Difference),min(PL3_Traj_Difference),...
    max(PL3_Traj_Difference)];

figure(4);plot(t,DoD_Difference);set(gca,'FontSize',20);
title('Depth of Discharge Difference Function Output','FontSize',20);
xlabel('time-Seconds','FontSize',20);ylabel('Discharge Difference (exp-
ref)','FontSize',20);
DoD_Traj_Descriptives=[median(DoD_Difference),...
    mean(DoD_Difference),min(DoD_Difference),...
    max(DoD_Difference)];

  figure(5);
```

```
    plot(t,DoD_Referent);set(gca,'LineWidth',1.5,'fontsize',24)
    title('Battery Depth of Discharge')
    xlabel('time (EpSec)')
    ylabel('DoD (%)')
    % Conversion is done to flip the graph to commonly depicted view
        set(gca,'YLim',[0,100]);
        ax = gca;
        ax.YTick = [0:10:100];
        ax.YTickLabel = {'FULL','10','20','30','40','50','60','70','80','90','DEPLETED'};
        set(gca,'YDir','reverse');

Payload_Power_Diff=sum((max(t)/(60^2))*([PL1_Traj_Descriptives(2),PL2_Traj_Descriptives(2
),PL3_Traj_Descriptives(2)]));

Integrated_Absolute_Error_PL1=sum(abs([AS_IS_Payload1Power-
In_View_AS_IS_W_Payload1Power]));

Integrated_Absolute_Error_Squared_PL1=sum(abs([AS_IS_Payload1Power-
In_View_AS_IS_W_Payload1Power].^2));

Integrated_Absolute_Error_PL1_Watt_Hours=Integrated_Absolute_Error_PL1/(60);

Integrated_Absolute_Error_PL2=sum(abs([AS_IS_Payload2Power-
In_View_AS_IS_W_Payload2Power]));

Integrated_Absolute_Error_Squared_PL2=sum(abs([AS_IS_Payload2Power-
In_View_AS_IS_W_Payload2Power].^2));

Integrated_Absolute_Error_PL2_Watt_Hours=Integrated_Absolute_Error_PL2/(60);

Integrated_Absolute_Error_PL3=sum(abs([AS_IS_Payload3Power-
In_View_AS_IS_W_Payload3Power]));

Integrated_Absolute_Error_Squared_PL3=sum(abs([AS_IS_Payload3Power-
In_View_AS_IS_W_Payload3Power].^2));

Integrated_Absolute_Error_PL3_Watt_Hours=Integrated_Absolute_Error_PL3/(60);

Integrated_Absolute_Error_DoD=sum(abs([AS_IS_DoD-DoD_Referent]));


refernt_trajectoryPL1=1 - Integrated_Absolute_Error_PL1/...%numerator
    ((Integrated_Absolute_Error_PL1)+((1)/(sum(abs(In_View_AS_IS_W_Payload1Power)))))
%denominator


refernt_trajectoryPL2=1 - Integrated_Absolute_Error_PL2/...%numerator
    ((Integrated_Absolute_Error_PL2)+((1)/(sum(abs(In_View_AS_IS_W_Payload2Power)))))
%denominator

refernt_trajectoryPL3=1 - Integrated_Absolute_Error_PL3/...%numerator
    ((Integrated_Absolute_Error_PL3)+((1)/(sum(abs(In_View_AS_IS_W_Payload3Power)))))
%denominator

refernt_trajectoryDoD=1 - Integrated_Absolute_Error_DoD/...%numerator
    ((Integrated_Absolute_Error_DoD)+((1)/(sum(abs(DoD_Referent))))) %denominator
[DoDIntegrated_Absolute_Error_PeiceWise,DoDrefernt_trajectory_PeiceWise] =
PeiceWise_Behavioral_fidelity_Analyzer (AS_IS_DoD,DoD_Referent);
[PL1_Absolute_Error_PeiceWise,PL1_trajectory_PeiceWise] =
PeiceWise_Behavioral_fidelity_Analyzer
(AS_IS_Payload1Power,In_View_AS_IS_W_Payload1Power);
```

```
[PL2_Absolute_Error_PeiceWise,PL2_trajectory_PeiceWise] =
PeiceWise_Behavioral_fidelity_Analyzer
(AS_IS_Payload2Power,In_View_AS_IS_W_Payload2Power);
[PL3_Absolute_Error_PeiceWise,PL3_trajectory_PeiceWise] =
PeiceWise_Behavioral_fidelity_Analyzer
(AS_IS_Payload3Power,In_View_AS_IS_W_Payload3Power);

figure(6);plot(PL3_trajectory_PeiceWise);set(gca,'FontSize',20);
title('Payload 3 power piecewise Trajectory Difference Function Output','FontSize',20);
xlabel('Piece Number (17 minute intervals)','FontSize',20);ylabel('Payload 3 fidelity
trajectory values ','FontSize',20);
PL3_Traj_Descriptives_PeiceWise=[median(PL3_trajectory_PeiceWise),...
    mean(PL3_trajectory_PeiceWise),min(PL3_trajectory_PeiceWise),...
    max(PL3_trajectory_PeiceWise)];

figure(7);plot(DoDrefernt_trajectory_PeiceWise);set(gca,'FontSize',20);
title('Depth of Discharge piecewise Trajectory Difference Function
Output','FontSize',20);
xlabel('Piece Number (17 minute intervals)','FontSize',20);ylabel('Depth of Discharge
fidelity trajectory values','FontSize',20);
DoD_Traj_Descriptives=[median(DoDrefernt_trajectory_PeiceWise),...
    mean(DoDrefernt_trajectory_PeiceWise),min(DoDrefernt_trajectory_PeiceWise),...
    max(DoDrefernt_trajectory_PeiceWise)];

figure(8);plot(PL2_trajectory_PeiceWise);set(gca,'FontSize',20);
title('Payload 2 power piecewise Trajectory Difference Function Output','FontSize',20);
xlabel('Piece Number (17 minute intervals)','FontSize',20);ylabel('Payload 2 fidelity
trajectory values ','FontSize',20);
PL3_Traj_Descriptives_PeiceWise=[median(PL3_trajectory_PeiceWise),...
    mean(PL3_trajectory_PeiceWise),min(PL3_trajectory_PeiceWise),...
    max(PL3_trajectory_PeiceWise)];

figure(9);plot(PL1_trajectory_PeiceWise);set(gca,'FontSize',20);
title('Payload 1 power piecewise Trajectory Difference Function Output','FontSize',20);
xlabel('Piece Number (17 minute intervals)','FontSize',20);ylabel('Payload 1 fidelity
trajectory values ','FontSize',20);
PL3_Traj_Descriptives_PeiceWise=[median(PL3_trajectory_PeiceWise),...
    mean(PL3_trajectory_PeiceWise),min(PL3_trajectory_PeiceWise),...
    max(PL3_trajectory_PeiceWise)];
% Resultant fidelity values are extremely low. It may be worth while to
% parse out the data (factor(10081) ans =  17   593) and see how running
% the analysis on smaller segments changes this (if at all) the most
% interesting data to run this on would be the DoD data which has the most
% variablility.0
DoD_Traj_1_counts=sum(DoDrefernt_trajectory_PeiceWise==1);
[counts,centers] =hist(DoDrefernt_trajectory_PeiceWise,5);
figure(10); hist(DoDrefernt_trajectory_PeiceWise,5); hold on ;
text(centers(1),counts(1)+30,num2str(counts(1)),'FontSize',20);
text(centers(2),counts(2)+30,num2str(counts(2)),'FontSize',20);
text(centers(3),counts(3)+30,num2str(counts(3)),'FontSize',20);
text(centers(4),counts(4)+30,num2str(counts(4)),'FontSize',20);
text(centers(5),counts(5)+30,num2str(counts(5)),'FontSize',20);
title('Histogram of Depth of Discharge Trajectory Outputs','FontSize',20);hold off;

PL1_counts=sum(PL1_trajectory_PeiceWise==1);
[countsPL1,centersPL1] =hist(PL1_trajectory_PeiceWise,5);
figure(11); hist(PL1_trajectory_PeiceWise,5); hold on ;
text(centersPL1(1),countsPL1(1)+30,num2str(countsPL1(1)),'FontSize',20);
text(centersPL1(2),countsPL1(2)+30,num2str(countsPL1(2)),'FontSize',20);
text(centersPL1(3),countsPL1(3)+30,num2str(countsPL1(3)),'FontSize',20);
text(centersPL1(4),countsPL1(4)+30,num2str(countsPL1(4)),'FontSize',20);
text(centersPL1(5),countsPL1(5)+30,num2str(countsPL1(5)),'FontSize',20);
title('Histogram of Payload 1 Trajectory Outputs','FontSize',20);hold off;
```

```
PL2_counts=sum(PL2_trajectory_PeiceWise==1);
[countsPL2,centersPL2] =hist(PL2_trajectory_PeiceWise,5,'FontSize', 20);
figure(12); hist(PL2_trajectory_PeiceWise,5); hold on ;
text(centersPL2(1),countsPL2(1)+30,num2str(countsPL2(1)),'FontSize',20);
text(centersPL2(2),countsPL2(2)+30,num2str(countsPL2(2)),'FontSize',20);
text(centersPL2(3),countsPL2(3)+30,num2str(countsPL2(3)),'FontSize',20);
text(centersPL2(4),countsPL2(4)+30,num2str(countsPL2(4)),'FontSize',20);
text(centersPL2(5),countsPL2(5)+30,num2str(countsPL2(5)),'FontSize',20);
title('Histogram of Payload 2 Trajectory Outputs','FontSize',20);hold off;

PL3_counts=sum(PL2_trajectory_PeiceWise==1);
[countsPL3,centersPL3] =hist(PL3_trajectory_PeiceWise,5);
figure(13); hist(PL3_trajectory_PeiceWise,5); hold on ;
text(centersPL3(1),countsPL3(1)+30,num2str(countsPL3(1)),'FontSize',20);
text(centersPL3(2),countsPL3(2)+30,num2str(countsPL3(2)),'FontSize',20);
text(centersPL3(3),countsPL3(3)+30,num2str(countsPL3(3)),'FontSize',20);
text(centersPL3(4),countsPL3(4)+30,num2str(countsPL3(4)),'FontSize',20);
text(centersPL3(5),countsPL3(5)+30,num2str(countsPL3(5)),'FontSize',20);
title('Histogram of Payload 3 Trajectory Outputs','FontSize',20);hold off;
```

Function built to perform the Piecewise Trajectory analysis

```
function [outputArg1,outputArg2] =
PeiceWise_Behavioral_fidelity_Analyzer (Simulated,Referent)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
if length(Simulated)-length(Referent)==0

Pieces = max(factor(length(Simulated)));
Incrament=(length(Simulated)/Pieces);
AS_IS_Payload1Power=Simulated;
In_View_AS_IS_W_Payload1Power=Referent;
for i=1:Pieces
  FactoredInput1(i,:)=Simulated((i-1)*(Incrament)+1:(i*(Incrament)));
  FactoredInput2(i,:)=Referent((i-1)*(Incrament)+1:(i*(Incrament)));



    Integrated_Absolute_Error(i,:)=sum(abs([FactoredInput1(i,:)-
FactoredInput2(i,:)]));
    Integrated_Absolute_Error_Squared_PL1=sum(abs([FactoredInput1(i,:)-
FactoredInput2(i,:)].^2));
    refernt_trajectory(i,:)=1 -
Integrated_Absolute_Error(i,:)/...%numerator

((Integrated_Absolute_Error(i,:))+((1)/(sum(abs(FactoredInput2(i,:))))))
) ;%denominator

end
outputArg1 =  Integrated_Absolute_Error;
outputArg2 = refernt_trajectory;
```

```
end
end
```

Function created to extract In View State data from STK data outputs

```
function [outputArg1,outputArg2] =
TargetAccessExtractor(accessTarget1,t)
%UNTITLED2 Summary of this function goes here
%   Detailed explanation goes here
indexList=[1];
Temporary1=length(accessTarget1);
for i=1:Temporary1
    holder= find((t>=[accessTarget1(i,2)])&(t<=[accessTarget1(i,3)]));
   indexList(length(indexList):(length(indexList)+length(holder)-1),1)=
(holder);

end
InViewState(1:length(t),1)=0;

InViewState(indexList)=1;

outputArg1 = indexList;
outputArg2 = InViewState;

end
```

Lines of script added to IntegrationScript.m of CubeSat RA to pass In View state vector into PAT.slx

```
[outputArg1,outputArg2] = TargetAccessExtractor(accessTarget1,t);
[outputArg12,outputArg22] = TargetAccessExtractor(accessTarget2,t);
[outputArg1GS,outputArg2GS] = TargetAccessExtractor(accessGS,t);
a1_StateVector(:,1)=t;
a1_StateVector(:,2)=3;
a1_StateVector(outputArg1,2)=2;
a1_StateVector(outputArg12,2)=2;
a1_StateVector(outputArg1GS,2)=2;
```

Bibliography

Colombi, J., Cobb, C., & Gallegos, D. (2012). Live-Virtual-Constructive Capabilities for Air Force Testing and Training. *ITEA*, 49-57.

Colombi, J., Miller, M. E., Schneider, M., McGrogan, J., Long, D. S., & Plaga, J. (2012). Predictive mental workload modeling: implications for system design. *Journal of Systems Engineering, 15*(4), 448-460.

Committee on Mathematical Foundations of Verification, Validation, and Uncertainty Quantification. (2012). *Assessing the Reliability of Complex Models.*

Cutts, D. C. (2020). *Enhancing Simulation Composability and Reuse with Modular FOMs.* JPC-I/MSIS.

Delligatti, L. (2013). *SysML Distilled: A Breif Guide to the Systems Modeling Language.*

Farrell, L. J. (2020). *A Reference Architecture for CubeSat Development.* WPAFB: Air Force Institute of Technology.

FCC. (2021). *FEDERAL COMMUNICATIONS COMMISSION 47 CFR .* FCC.

Folk, M. C. (2010). *Transforming the geocomputational battlespace framework with HDF5.* Alexandria: Engineer Research and Development Center.

Friedenthal, S., & Moore, A. (2015). *A Practical Guide to SysML: The Systems Modeling Language.*

Fujimoto, R. M. (2003). Distributed Simulation Systems. *Winter Simulation Conference* (pp. 124-134). IEEE.

Gore, R. &. (2007). An exploration-based taxonomy for emergent behavior analysis in simulations. *IEEE 2007 Winter Simulation Conference* (pp. 1232-1240). IEEE.

HFM-216, N. (2015). *Syntheitic Environments for HSI Application, Assessment, and Improvement.* NATO.

Hill, R. R. (2017). A history of United States military simulation. *2017 Winter Simulation Conference (WSC)* (pp. pp. 346-364). IEEE.

Hill, R., Tolk, A., Hodson, D., & Millar, J. (2018). OPEN CHALLENGES IN BUILDING COMBAT SIMULATION SYSTEMS TO SUPPORT TEST, ANALYSIS AND TRAINING. *Winter Simulaiton Conference.*

IEEE. (2011, Jan 24). IEEE Recommended Practice for Distributed Simulation Engineering and Execution Process (DSEEP) std. *Revision of IEEE Std 1516.3-2003*, pp. 1-79.

Iridium Satellite LLC. (2020). *Iridium 9603 Data Sheet.*

Kelly, S. R. (2021). *A Reference Architecture for Rapid CubeSat Development.* Wrigth-Patterson Air Force Base: Air Force Institute of Technology.

King, R. (2009). *On the role of assertions for conceptual modeling as enablers of composable simulation solutions.* Old Dominion University.

Laboratory, Space Dynamics. (n.d.). *CADET UHF & UHF S-BAND RADIOS.* Utah State University.

Malloy, M. G. (n.d.). *Acquisition and TE in 2025 & Beyond... Meeting the Pace of Need.*

Menke. (2019). Joint Simulation Environment for United States Air Force Test Support. *The United States Air Force Material Command Foreign Disclosure and Release Process, 88th ABW, Case No. 2019-4265*, 1-14.

Menke, T. (2019). *Joint Simulation Environment for United States Air Force Test Support.* WPAFB: NATO S&T Organization.

Miller, D. C. (1995). SIMNET: The advent of simulator networking. *Proceedings of the IEEE*, pp. 1114-1123.

Mission Engineering Guide. (2020). *Mission Engineering Guide.* Washington, D.C.: United States Department of Defense.

Nedza, J. A. (2010). *Transforming the Geocomputational Battlespace Framework with HDF5.* ALEXANDRIA: ENGINEER RESEARCH AND DEVELOPMENT CENTER ALEXANDRIA VA TOPOGRAPHIC ENGINEERING CENTER.

Rhees, T. R. (1981). *Research on Management Concepts for Large-Scale Simulations of Naval Warfare. .* MCLEAN: DECISIONS AND DESIGNS INC .

Roza, Z. M. (2005). Simulation Fidelity Theory and Practice: A Unified Approach to Defining, Specifying and Measuring the Realism of Simulations.

Sargent, R. (2014). VERIFYING AND VALIDATING SIMULATION MODELS. *Winter Simulation Conference.*

Siegfried. (2013). Effective and Efficient Training Capabilities through Next Generation Distributed Simulation Environements. *NATO S&T*.

Siegfried, R., Diehl, A., Diallo, S., Bertschik, M., Herrmann, G., & Rother, M. (n.d.). *Outline of a Service-Based Reference Architecture for Effective and Efficient Use of Modelling and Simulation.* NATO S&T Organization.

Siegfried, R., Luthi, J., Herrmann, G., & Hahn, M. (2011). *How to ensure Fair Fight in LVC Simulations: Architectural and Procedural Approaches.* NATO S&T.

SR, H. e. (2011). VEVA as German approach towards operationalizing the DSEEP: Overview and first experiences. *Spring Simulation Interoperability Workshop*, (pp. 1-10). Boston, MA.

Task Group MSG-086. (2015). *Simulation Interoperability .* NATO S&T.

Tolk. (2012). *Engineering Principles of Combat Modeling and Distributed Simulation.* Wiley.

Tolk. (2013). Interoperability, Composability, and Their Implications for Distributed Simulation. *IEEE ACM International Symposium on Distributed Simulation and Real Time Applications.*

Tolk, A., Diallo, S. Y., King, R. D., & Turnitsa, C. D. (n.d.). A Layered Approach to Composition and Interoperation in Complex Systems. In *Complex Systems in Knowledge-based Environments: Theory, Models and Applications* (pp. 41-74).

Z.C. Roza, D. G. (2000). *Report Out of the Fidelity Experimentation ISG.*

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. REPORT DATE *(DD-MM-YYYY)*	**2. REPORT TYPE**	**3. DATES COVERED** *(From - To)*
16-9-2021	Master's Thesis	Aug 2019 – Sept 2021

4. TITLE AND SUBTITLE
Applying Model-Based Systems Engineering and Fidelity Quantification to Support Fair Fight in a Distributed Simulation System

5a. CONTRACT NUMBER

5b. GRANT NUMBER

5c. PROGRAM ELEMENT NUMBER

6. AUTHOR(S)
Erbe, Nathaniel D
Lemmer, David P

5d. PROJECT NUMBER

5e. TASK NUMBER

5f. WORK UNIT NUMBER

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Air Force Institute of Technology
Graduate School of Engineering and Management (AFIT/EN)
2950 Hobson Way, Building 640
WPAFB OH 45433-7765

8. PERFORMING ORGANIZATION REPORT NUMBER
AFIT-ENV-MS-21-S-074

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Air Force Life Cycle Management Center-Architecture and Integration Directorate
Bldg 802 2303 8th Street
WPAFB, Ohio 45433-7222

10. SPONSOR/MONITOR'S ACRONYM(S)
AFLCMC-XA

11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION / AVAILABILITY STATEMENT
DISTRUBTION STATEMENT A. APPROVED FOR PUBLIC RELEASE; DISTRIBUTION UNLIMITED.

13. SUPPLEMENTARY NOTES

14. ABSTRACT

Current 5th and 6th generation fighter aircraft capabilities, and the DoD's push towards digital engineering have created an environment in which simulated testing using live, virtual, and constructive assets has increasing utility. These simulations need to be credible in the eyes of the stakeholders, which for distributed simulation systems includes establishing fair fight between simulation services. Fair fight is when multiple simulations interoperate without generating one-sided systematic advantages. Issues that impede fair fight can be categorized into interoperability issues at the simulation's implementation level and incompatible representations of reality between underlying models.

This research used model-based systems engineering to generate non-functional system requirements for fair fight. The requirement for alignment of models' conceptual representation of reality, led the need to quantify fidelity. Fidelity metrics that speak towards system level complexity, adherence to property scope specifications, and real-world behavior accuracy were applied to a SysML driven simulation. It was concluded that metrics for structural decomposition can describe a model's limitations and scope, while traceability between model properties and real-world specifications has more utility than a singular metric. Lastly, by generating referent data with external simulation, behavioral differences can be quantified and downstream effects due to low model fidelity can be detected.

15. SUBJECT TERMS
Fair Fight, Modeling and Simulation, Distributed Simulation System, MBSE

16. SECURITY CLASSIFICATION OF:			**17. LIMITATION OF ABSTRACT**	**18. NUMBER OF PAGES**	**19a. NAME OF RESPONSIBLE PERSON**
a. REPORT	**b. ABSTRACT**	**c. THIS PAGE**			Dr. John Colombi, AFIT/ENV
U	U	U	UU	98	**19b. TELEPHONE NUMBER** *(include area code)* (937)-255-6565 John.Colombi@afit.edu