DEVELOPING EFFECTIVENESS MEASURES FOR SYSTEM ANALYSIS TOOLS IN HYPERSONIC VEHICLE DESIGN

by

Eli V. Sitchin

A Thesis

Submitted to the Faculty of Purdue University

In Partial Fulfillment of the Requirements for the degree of

Master of Science in Aeronautics and Astronautics

School of Aeronautics and Astronautics

West Lafayette, Indiana

May 2022

THE PURDUE UNIVERSITY GRADUATE SCHOOL
STATEMENT OF COMMITTEE APPROVAL

Dr. Daniel A. DeLaurentis, Chair

School of Aeronautics and Astronautics

Dr. Jonathan Poggie

School of Aeronautics and Astronautics

Dr. Vikas Tomar

School of Aeronautics and Astronautics

Approved by:

Dr. Gregory A. Blaisdell

To Mom, Dad, and Wyatt

ACKNOWLEDGMENTS

I would like to thank my advisor, Dr. Daniel DeLaurentis for his invaluable mentorship during my time in the Center for Integrated Systems in Aerospace, as well as Dr. Kristopher Ezra for the encouragement, advice, and occasional handholding he provided as I worked on this thesis. I couldn't have done this without them.

I would also like to take this opportunity to thank Dr. Cesare Guariniello and Dr. Bill O'Neill for their work on the SDDA and RPO tools, and who helped me understand them and their applications better.

Finally, I would like to take this opportunity to thank the people in my life who made the last two years far more pleasant: Noah Stockwell, for his constant friendship; my climbing partners Nate Timmerman and Paul Fuchs, with whom I climbed over 200 routes and progressed five grades since the start of my graduate studies; my D&D group, for providing me a weekly escape from reality; my colleagues in the Center for Integrated Systems in Aerospace, with whom I bonded over the idiosyncrasies of graduate student life; and my mom, dad, and brother, for their unending love and support. I am eternally grateful to have all of you in my life.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

ABBREVIATIONS

AFRL	Air Force Research Laboratory
AWB	Analytic Workbench
COD	Criticality of Dependence
GHV	Generic Hypersonic Vehicle
INCOSE	International Council on Systems Engineering
IOD	Impact of Dependence
ITAR	International Traffic in Arms Regulations
MBSE	Model-Based Systems Engineering
NASP	National Aerospace Plane
RPO	Robust Portfolio Optimization
SDDA	System Developmental Dependency Analysis
SE	Self-Effectiveness
SOD	Strength of Dependence
SODA	System Operational Dependency Analysis
SoS	System-of-Systems
SoSE	System-of-Systems Engineering

ABSTRACT

Throughout the design process, systems engineers use analysis tools to characterize the composition, development, and behavior of complex engineered systems. As such, designers must ensure that these tools generate accurate results for the systems they're designed to analyze. One such suite of tools is the Analytic Workbench (AWB), a framework couched in a model-based systems engineering (MBSE) environment which provides methods for integrating constituent systems into a system-of-systems (SoS) or a complex engineered system, while taking into account the capabilities and requirements of the constituent systems, and the interdependencies between them. Hypersonic vehicles present a number of design challenges that result in a narrow performance envelope and a high degree of interdependence between subsystems, and so the AWB provides an appropriate tool set for characterizing the conceptual design of this class of system. The purpose of this thesis is to generate effectiveness metrics for the tools within the AWB — specifically Robust Portfolio Optimization (RPO) and System Developmental Dependency Analysis (SDDA) — in the context of a hypersonic vehicle design problem. In particular, this thesis focuses on verification and validation metrics, and applies these metrics to several demonstration cases, in which the outputs of RPO and SDDA analyses of a hypersonic vehicle model consisting of top-level subsystems are compared with a hypothetical physical vehicle. This thesis examines several candidate effectiveness metrics, and then applies the ones that satisfy the requirements for RPO and SDDA verification and validation to the appropriate demonstration cases. The AWB outputs for the vehicle models in these demonstration cases deviate slightly from the corresponding quantities for the hypothetical physical vehicles, and the effectiveness metrics decrease in value the greater these deviations are. Subsequent explorations of these metrics could apply these effectiveness to other types of design problems, including analyses involving lower-level subsystems of hypersonic vehicles.

1. INTRODUCTION

1.1 Context

In the past several years, the aerospace industry has increased its interest in hypersonic flight vehicles, both for military and civilian use. Hypersonic vehicles present a number of design challenges that are either not present in subsonic or supersonic aircraft, or are much more pronounced at higher velocities. For instance, vehicles with airbreathing propulsion systems require more than one engine type to accelerate to hypersonic speeds, as the ramjet and scramjet engines optimized for hypersonic cruise velocities can only operate efficiently at at high supersonic speeds [1]. Additionally, the variation in aerodynamic behavior over the operating velocity of the aircraft significantly constrains the vehicle's aerodynamic profile, as well as the materials comprising its structure — since the materials must be both light enough to allow sufficient acceleration by the engine, and durable enough to withstand temperatures of well over 1000 K [2]. As a result of these challenges, hypersonic vehicles have a relatively narrow performance envelope compared to other aerospace vehicles. This leads to high degree of interdependence between subsystems, and therefore must have a highly integrated design that's robust enough to account for performance uncertainties, especially when the uncertainty is highest during the conceptual design phase. To meet these design challenges, designers have transitioned to using model-based systems engineering (MBSE) throughout the vehicle design process. INCOSE defines MBSE as "the formalized application of modeling to support system requirements, design, analysis, verification and validation activities beginning in the conceptual design phase and continuing throughout development and later life cycle phases" [3] This definition distinguishes MBSE from document-based systems engineering by implying that systems designed with MBSE rely on a centralized digital model that serves as a single source of truth regarding all aspects of the system over the course of its development, as opposed to a decentralized collection of documents detailing the requirements, analysis, and other design processes separately.

At every stage of the design process, there is some uncertainty present regarding the actual parameters and performance of the physical system, which — given the complex nature of aerodynamics and other physical forces acting on aerospace vehicles — is impossible

to model with perfect accuracy. Thus, it's useful to have some sort of metric to evaluate the MBSE tools once the physical system has been built, so that systems engineers can improve upon it for use on subsequent projects. In this thesis, we will focus on one particular set of tools used for MBSE analysis — the analytic workbench (AWB), developed by the Center for Integrated Systems in Aerospace at Purdue University with funding from the Systems Engineering Research Council within the U.S. Department of Defense — and consider the metrics that could apply to these tools in the conceptual design of hypersonic flight vehicles.

1.2 Overview of Analytic Workbench

1.2.1 Analytic Workbench as an MBSE Framework

The analytic workbench was developed as a way to address "pain points" in system-of-systems engineering (SoSE) [4]. In particular, the AWB focuses on the problem of integrating constituent systems into a system-of-systems (SoS) while addressing the issue of interdependencies between various constituent systems in an SoS, which have varying degrees of interconnectedness. To this end, the AWB has three main goals: Quantify performance (i.e., capability) for the SoS as a whole, its constituent systems, and the links between them; determine how how changes to the composition of the SoS affect its performance and risk; and identify optimal portfolios for SoS architectures, given constraints on risk, cost, and performance [5]. Of course, given the size and complexity of many SoSs — or other engineered systems with a high degree of interdependence between subsystems — many of these interdependencies are difficult to grasp intuitively. Thus, tools within the AWB typically present the SoS as a network of constituent systems. The interdependencies within this network are represented as connections between nodes, and are characterized by easy-to-understand parametric models. This representation of SoS as networks of constituent systems connected by parametric relationships allows the AWB to be used in nearly any discipline (i.e., the tools are "domain agnostic"), and on problems of varying scale. Within the SoS design process, the AWB tools are used with an iteratively refined truth model, allowing for the continued analysis, verification, and validation of the model throughout the design process, starting from an initial architecture in the conceptual design phase. Thus, the AWB satisfies

INCOSE's definition of an MBSE framework. Fig. 1.1 illustrates this process process within the larger context of the SoS design process.

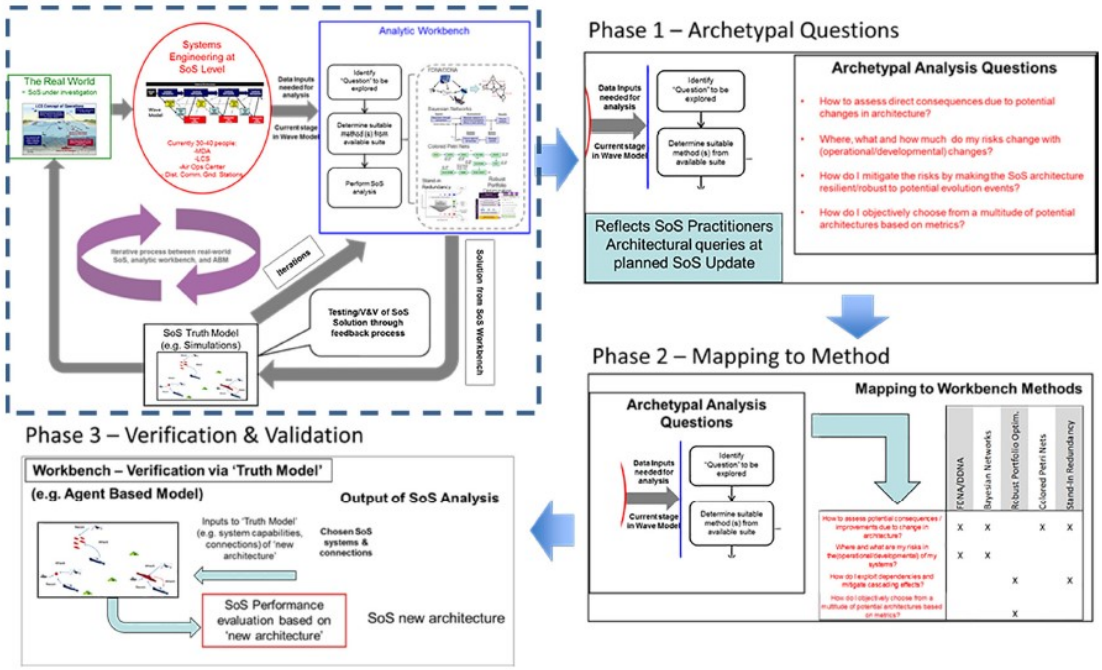

Figure 1.1. Role of the Analytic Workbench — shown in the blue outline — in the SoS design process [5].

The AWB was initially designed as a suite of tools intended for use in SoSE; however, the aforementioned "pain points" are by no means exclusive to SoS problems. While the subsystems comprising hypersonic flight vehicles lack the operational and managerial independence, geographical distribution, and evolutionary development characteristics of an SoS [6], hypersonic vehicles still face significant size, weight, and power (SWaP) constraints that strengthen the interdependencies between subsystems and make integration challenging. As such, the AWB would be an appropriate suite of tools for designing hypersonic vehicles.

1.2.2 Tools Within the Analytic Workbench

As applied to hypersonic vehicle design problems, the AWB contains three tools of interest: Robust Portfolio Optimization (RPO), System Operational Dependency Analysis (SODA), and System Developmental Dependency Analysis (SDDA). In the case of a com-

plex engineered system — rather than an SoS — RPO treats the system as a portfolio of potential constituent subsystems that can be selected or connected using a set of user-defined rules and constraints. The objective of RPO is to generate a complete and functional system by selecting a set of subsystems that maximizes the capability of the system as a whole, while keeping risk and cost within acceptable bounds. RPO frames this as an optimization problem that exactly parallels the portfolio optimization methods used in financial analysis [7]. Each constituent subsystem is defined by a set of input requirements — things the subsystem requires to function — and output requirements — outputs that could potentially satisfy other subsystems' input requirements — as well as capabilities, costs, and the uncertainties associated with them. There are also often compatibility constraints, dictating whether certain subsystems are required to accompany certain others, or if they're incapable of both being selected in the same portfolio. The overarching system also has its own set of capabilities, which map to the subsystem capabilities and can be assigned different weights in the objective function of the optimization problem. Fig. 1.2 illustrates how RPO generates portfolios system portfolios based on these capabilities, requirements, uncertainties, costs, and constraints.

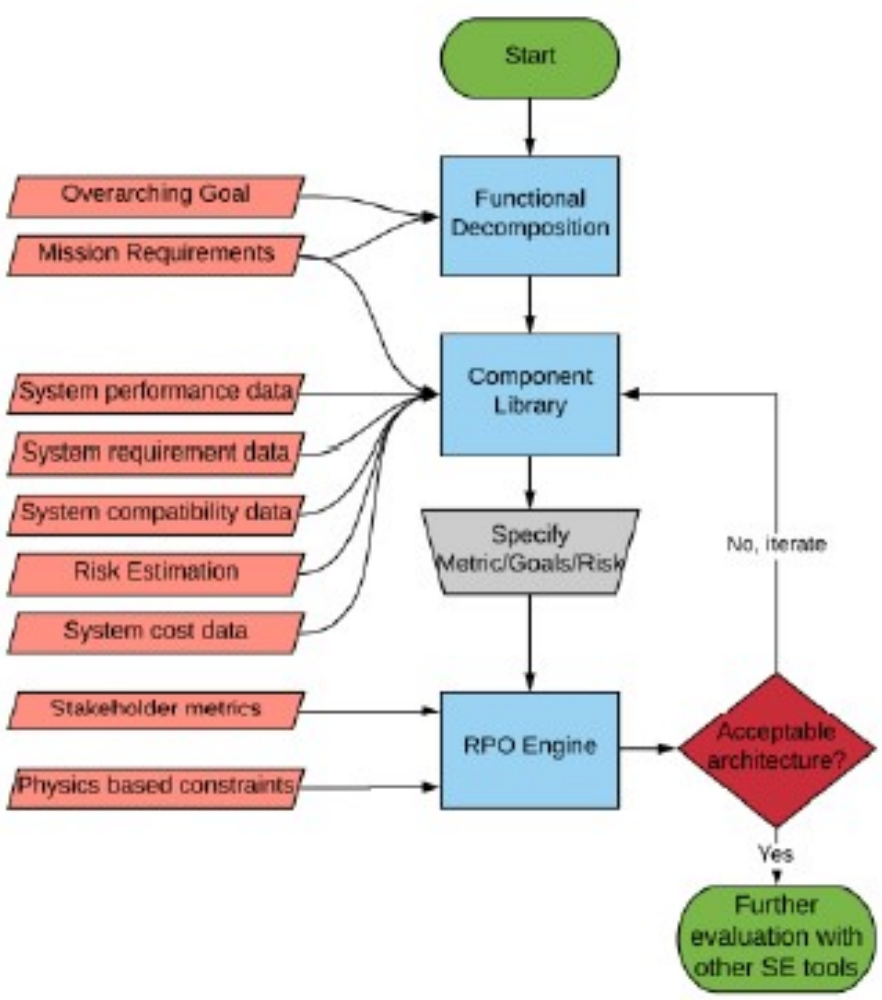

Figure 1.2. Graphic illustrating the RPO process, and how it connects with other systems engineering tools [8].

The other two tools — SODA and SDDA — are devoted to analyzing previously generated architectures, focusing on operational risks and developmental risks, respectively. Both SODA and SDDA were developed based on existing tools characterizing these risks — Functional Dependency Network Anlaysis (FDNA) in the case of SODA, and Program Evaluation and Review Technique (PERT) in the case of SDDA [9][10]. SODA and SDDA arrange the subsystems selected by RPO into dependency networks, where links between subsystems are characterized by a simple parametric relationship involving either two parameters in the case of SDDA, or three in the case of SODA [9][11]. The links are directional, with an input node i impacting the operability (SODA) or start of development (SDDA) of output node j.

Operability is a variable corresponding to the overall functioning of the subsystem — where a value of 100 signifies perfect performance, and a value of 0 signifies complete nonfunctionality — while the start of development is a measure of time. Additionally, each node in a SODA network has a measure of its internal health called self-effectiveness (SE), where 0 refers to a complete lack of functionality and 100 refers to the state of optimal performance. For SDDA, this measure is called the punctuality (P), and a value of 100 indicates that the subsystem will develop in the minimum possible time, while a value of 0 indicates that it will develop in the maximum possible time. In the absence of any input nodes for a subsystem in a SODA network (i.e., the operability of a subsystem is entirely independent of that of any other subsystems), the operability of a particular node is equal to it's SE. For an SDDA network, meanwhile, the beginning time of development for a subsystem without any input nodes is equal to zero (i.e., the subsystem begins its development before any others). For SODA specifically, we can also define the operability of an entire system in relation to the operabilities of the subsystems. We often define the total operability as the average operability of each sink node (i.e., subsystems that do not impact the operability of other subsystems), the minimum sink node operability, or the output of a user-defined scoring function — that can vary depending on the problem — based on the operabilities of all nodes in the network.

Fig. 1.3 illustrates how the AWB tools impact the design process for hypersonic vehicles. From a basic conceptual model, systems engineers generate conceptual requirements to make a model-based representation of the vehicle, defined mainly in terms of its intended operational capabilities and design requirements. These are then fed into RPO, which uses these capabilities and requirements to select portfolios of suitable subsystems from a library of all possible hypersonic vehicle subsystems. RPO then outputs these portfolios into SODA and SDDA, where the constituent subsystems form dependency networks based on the operational and developmental relations between them, and the analysis tools use these to characterize the risk associated with the operation and development of the system. The analyses output by all three tools would then form the basis for a digital twin of the vehicle, which would mirror the vehicle's behavior in real time. The development of this digital twin is outside the scope of this thesis; this thesis exclusively focuses on the AWB tools themselves, shown in blue.

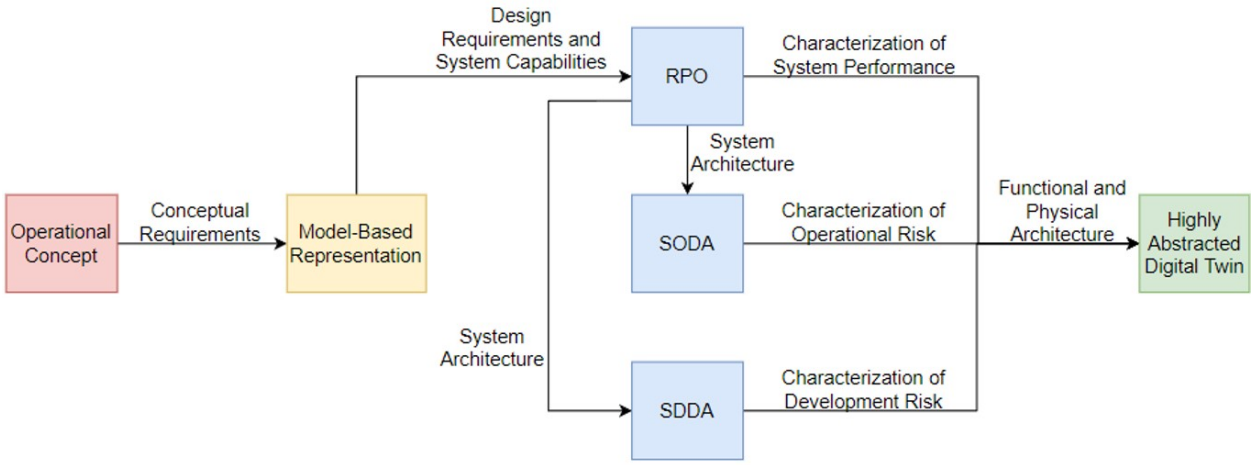

Figure 1.3. Graphic illustrating the role of the AWB in the hypersonic vehicle design process.

1.3 Research Question

Since the importance of the Analytic Workbench tools in MBSE and their applications to hypersonic flight vehicle design has been established, this thesis will focus on how to evaluate the effectiveness of these tools in the context of hypersonic vehicle conceptual design. In particular, this thesis will focus on the following research question:

What quantitative measures can we use to evaluate the effectiveness of Analytic Workbench tools in generating and evaluating hypersonic vehicle architectures?

To answer this question, this thesis will specifically focus on RPO and SDDA, and investigate the tools' effectiveness when applied to example systems, both as a way to apply the quantitative measures and to validate the model itself. In this context, "effectiveness" means accuracy within a particular abstraction level, in this case top-level subsystems of hypersonic vehicles (the effectiveness metrics explored within this thesis are therefore most useful when applied to problems within this level of abstraction). This thesis also assumes that any analysis tools feeding RPO and SDDA are accurate, so that the effects of the AWB tools can be isolated. Chapter 2 of this thesis will introduce the quantitative metrics

used for RPO and SDDA and the justifications for them, as well as further discussion of the tools themselves. Chapter 3 will outline the approach used in designing demonstration cases for applying these measures and validating the Analytic Workbench models. Chapter 4 shows the demonstration cases to which RPO and SDDA were applied, and discusses the results. Chapter 5 discusses the conclusions and limitations of this thesis, and makes recommendations for further exploration of this topic.

2. DEVELOPMENT OF EFFECTIVENESS METRICS

2.1 Introduction

This chapter discusses the development of effectiveness metrics for RPO and SDDA. To do so, we begin by discussing the types of effectiveness metrics systems engineers need to determine if the tools they use work as intended. In this thesis, we will present two — verification and validation — which consequently means that we must develop four metrics in all (two each for RPO and SDDA). In the case of other engineering analysis tools, researchers have already proposed various verification and validation effectiveness metrics, and we explore several of these in order to examine if any would be appropriate for RPO and SDDA.

We then develop the requirements for each of the four metrics, which provide guidance on generating the metrics themselves. Before doing so, we first discuss what constitutes a good requirement, as poorly written requirements would result in inadequate metrics, and a complete set of well written requirements would assist developers who wish to expand upon these effectiveness metrics further or apply them to other tools. The requirements presented in this thesis rely on the guidance provided by *Design for Safety*, by Louis J. Gullo and Jack Dixon [12], and the NASA System's Engineering Handbook [13]. These sources not only define the characteristics of good requirements, but divide system requirements into several categories.

Once defined, these requirements help us determine which types of verification and validation metrics are appropriate for RPO and SDDA. From the set of appropriate metrics, we select one metric for each of the four that we need — RPO verification, RPO validation, SDDA verification, and SDDA validation — which we can apply to our hypersonic vehicle demonstration case. We conclude this chapter by comparing the selected metrics for RPO and SDDA, and discussing the conditions under which they might be combined to form a unified framework for a system designed using the AWB.

2.2 Types of Effectiveness Metrics

When discussing whether an analysis tool conducts the task it's designed to do with a reasonable degree of accuracy, there are two questions that systems engineers should consider:

1. Do the results output by the tool correspond to those given by the mathematical formulae on which it's based?

2. Do the results output by the tool accurately predict the characteristics, behavior, or development of the physical system it models?

These two questions correspond to verification and validation of the analysis tool, respectively. Fig. 2.1 visualizes the relationship between the mathematical concepts that define an analysis tool, the computerized implementation of the tool, and the physical system the tool analyzes, and the role of verification and validation in reconciling these three things.

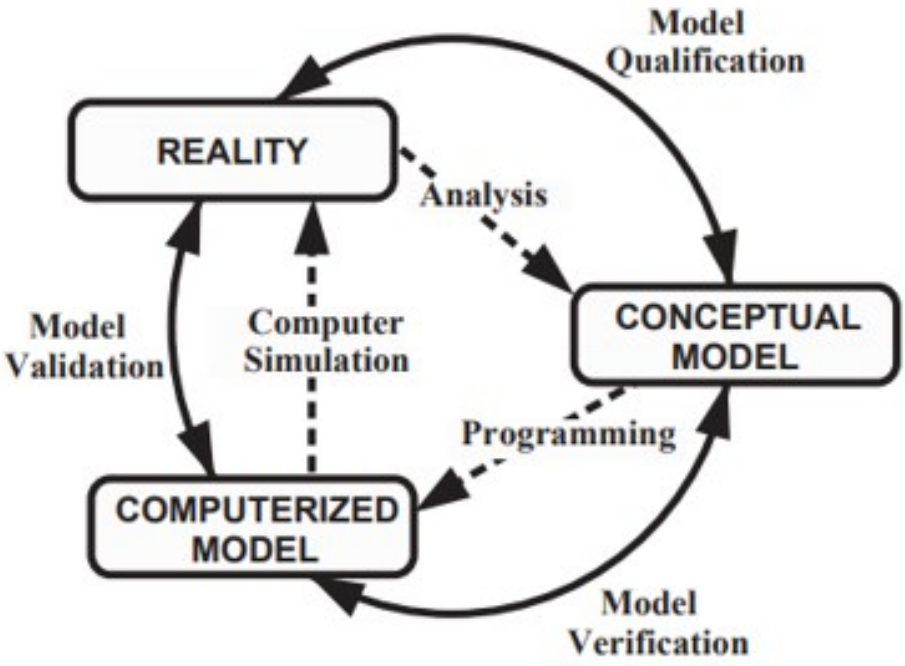

Figure 2.1. Relationship between physical systems, computerized models, and the concepts behind the computerized models, as well as the role of verification and validation [14].

INCOSE defines verification as "a set of activities that compares a system or system element against the required characteristics. This may include, but is not limited to, specified requirements, design description, and the system itself" [15]. In the context of an analysis tool, verification involves identifying the mathematical concepts that the model implements, and ensuring that the outputs of the analysis tool match the results of applying these concepts independently (i.e., ensuring that the tool correctly solves the problem it was built to solve). This often involves generating a suite of example problems simple enough that a design engineer can calculate the outputs of the analysis tool on their own, using only the underlying mathematical concepts [14].

Validation, meanwhile, "is the set of activities ensuring and gaining confidence that a system is able to accomplish its intended use, goals, and objectives (i.e., meet stakeholder requirements) in the intended operational environment" [15]. In other words, validation ensures that the design of a system is capable of satisfying all the requirements for said system. For an analysis tool, validation involves comparing the outputs of the computerized implementation of the tool with the physical system being analyzed, and confirming that the analysis tool predicts the behavior, development, or characteristics of the system within a reasonable degree of uncertainty. Additionally, validation doesn't just refer to comparison with the completed system; it can also apply to the results of physical tests of individual components and subsystems (e.g., a wind tunnel test of an airfoil). The rest of this chapter will involve the development of effectiveness metrics that can be applied to the verification and validation of RPO and SDDA.

2.3 Purpose of Analytic Workbench Tools

Before we develop effectiveness metrics for RPO and SDDA, we need to explore how these tools function in depth. In the context of RPO, which frames the generation of system portfolios as an optimization problem that maximizes capability while minimizing cost and risk, there are several different ways to define risk. One way is to focus on the risks incurred during the development of the system, as the robust mean-variance optimization method of RPO does [7]. Another is to take into account the uncertainties in subsystem performance

during operations, using methods developed by Bertsimas and Sim [16]. Each of these methods assumes that the uncertainties follow symmetrical distributions [17], which does not always hold in real-world operations. The Conditional Value-at-Risk (CVaR) approach allows for more complex distributions of risk functions, which are often derived from agent-based simulation data [18]. For the purposes of this thesis, all future references of RPO will refer exclusively to the Bertsimas-Sim approach, as our implementation of Bertsimas-Sim RPO is more complete and better tested than the other two methods. Fig. 2.2 shows the setup for an RPO example problem using the Bertsimas-Sim approach. The two columns on the left represent the candidate systems, while the next five outline system capabilities, the two columns following that outline system support requirements, and the column on the right indicates the range of the uncertainty bounds. The red and blue boxes indicate which uncertainties being simulated apply to which capabilities.

Package		Weapon Range	Detect. Range	Anti Mine	Comm. Capability	Power Capability	Power Require.	Comm. Require.	Uncertainty Set
ASW	Variable Depth	0	50	0	0	0	100	200	0
	Multi Fcn Tow	0	40	0	0	0	90	120	0
	Lightweight tow	0	30	0	0	0	75	100	0
MCN	RAMCS II	0	0	10	0	0	70	120	0
	ALMDS (MH-60)	0	0	20	0	0	90	150	0
SUW	N-LOS Missiles	25	0	0	0	0	0	250	0
	Griffin Missiles	3	0	0	0	0	0	100	0
Seaframe	Package 1	0	0	0	0	300	0	0	10
	Package 2	0	0	0	0	450	0	0	70
	Package 3	0	0	0	0	500	0	0	100
Comm.	System 1	0	40	0	180	0	100	0	30
	System 2	0	200	0	200	0	120	0	35
	System 3	0	0	0	240	0	140	0	45
	System 4	0	0	0	300	0	160	0	55
	System 5	0	0	0	360	0	180	0	60
	System 6	0	0	0	380	0	200	0	80

Figure 2.2. The Bertsimas-Sim version of RPO applied to a littoral combat ship example [17].

SDDA, meanwhile, simulates delays in the development of a particular subsystem to determine how it impacts the overall development schedule of the system by characterizing the relationships between pairs of subsystems in terms of two parameters: strength of dependence (SOD), and criticality of dependence (COD) [11]. The nodes themselves, meanwhile, are characterized by three variables: The minimum development time t_{min}, the maximum development time (in the absence of any input nodes) t_{max}, and the punctuality P, the

latter of which is equivalent to the SE parameter for a node in a SODA network. Unlike in a SODA network, cycles are impossible in an SDDA network, as a subsystem cannot be developmentally dependent on itself. The development time of node i is equal to

$$t_c^{\mathrm{i}} = t_{min}^{\mathrm{i}} + (1 - \frac{P_{\mathrm{i}}}{100})(t_{max}^{\mathrm{i}} - t_{min}^{\mathrm{i}}) \tag{2.1}$$

If node j is dependent on node i, then the beginning time of node j is equal to either Eq. 2.2 (if the punctuality of node i is greater than the COD) or Eq. 2.3 (if the punctuality of node i is less than the COD).

$$t_b^{\mathrm{j}} = t_c^{\mathrm{i}} - \frac{t_{min}^{\mathrm{j}}(1 - SOD_{\mathrm{ij}})(P_{\mathrm{i}} - COD_{\mathrm{ij}})}{100 - COD_{\mathrm{ij}}} \tag{2.2}$$

$$t_b^{\mathrm{j}} = t_c^{\mathrm{i}} \tag{2.3}$$

This relationship is shown in Fig. 2.3. Now, if a subsystem is developmentally dependent on two or more other subsystems, the dependent subsystem calculates multiple possible beginning development times, one from the relationship with each input node. From these calculated beginning times, there are two ways to determine a single beginning time: Either by using the average beginning time output by all the two-node relationships, or by using the latest beginning time. The former algorithm is known as standard SDDA, while the latter algorithm is known as $\mathrm{SDDA}_{\mathrm{Max}}$.

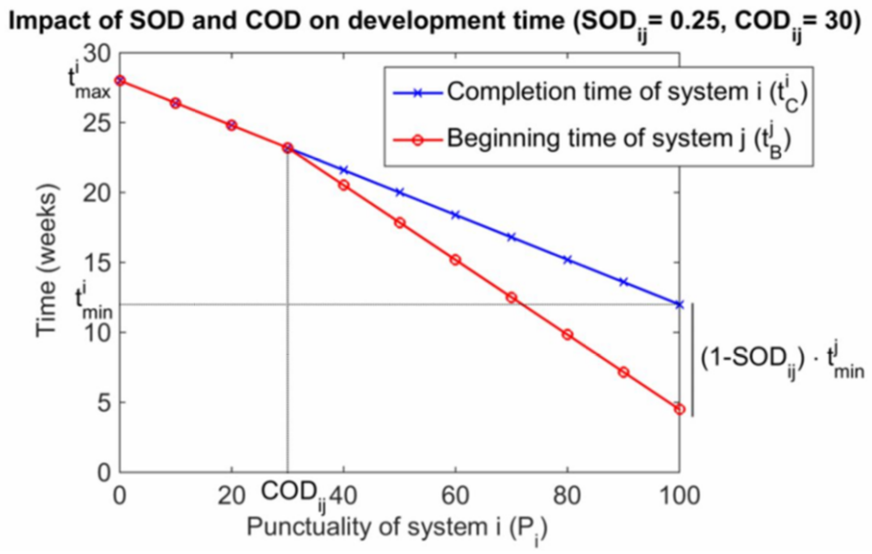

Figure 2.3. Relationship between the development time of an input node and the beginning time of an output node in an SDDA network [11].

Additionally, SDDA can treat the punctuality of each node either a deterministic variable or as following a probability density function. In the latter case, SDDA assumes that the punctuality follows a symmetric beta probability distribution, where the user-input punctuality represents the mode of the distribution. The mode may not correspond to the mean or median, as the tails of the distribution may be cut off — and the resulting asymmetric distribution normalized — if they correspond to values outside the range of 0 to 100. SDDA uses a beta probability density function, rather than a normal distribution, to describe the punctuality because the latter is explicitly unbounded, while the beta distribution only has a nonzero value over a finite interval.

While each tool carries out a different set of tasks, RPO and SDDA share a common purpose within the AWB: manage the complexity of a system with a high degree of interdependence between subsystems, and provide methods for addressing these interdependencies to characterize the conceptual design of such a system in a way that's easily understandable by designers. In the case of RPO, this complexity arises from the large library of subsystems from which an architecture for a complex system can be generated, and RPO manages it by considering the overall system capabilities (i.e., the mission objectives that the system must accomplish), and the subsystem capabilities that contribute to these system capabil-

ities being satisfied [18]. Thus, RPO generates portfolios by selecting the subsystems that most effectively satisfy the subsystem capabilities that fulfill the overall system capabilities, within acceptable risk tolerances (for the Bertsimas-Sim method, risk is characterized as a symmetric uniform uncertainty distribution for the subsystem capabilities and output requirements [17]). RPO can also generate portfolios for multiple different overall cost constraints, and the resulting system capability measure can be plotted against the system cost in a Pareto frontier. As such, verification for RPO involves creating an example problem where the portfolios that maximize system capability at different cost constraints are obvious enough that a systems engineer can generate them intuitively, and ensuring that the RPO implementation generates the correct portfolios. Validation for a hypersonic vehicle problem, meanwhile, takes an RPO-generated portfolio at a single cost constraint, and compares the resulting capabilities and costs for each subsystem with those of the physical vehicle designed and manufactured from that portfolio.

For an SDDA analysis, the complexities arise from the interdependencies between subsystems during the development phase. SDDA characterizes these interdependencies as a mathematical relationship between the beginning time of development of a particular subsystem and the end time of development of all the subsystems that impact its development. Thus, verification for SDDA involves calculating the beginning and end times of a network of subsystems in an example problem using the mathematical relationships that define SDDA, and comparing them to the outputs of a computerized implementation of SDDA. SDDA validation for a hypersonic vehicle problem is similar to verification, only the beginning and end times output by the implementation of SDDA would be compared to the development timeline of an existing hypersonic vehicle instead of the results of a conceptual model.

2.4 Requirements for Effectiveness Metrics

Before defining the effectiveness metrics for RPO and SDDA, we must first outline exactly what each metric should take into account when calculating how effectively each tool functions. By generating requirements for the metrics, we can define the exact scope of each

metric, and provide guidelines to narrow down the class of effectiveness metric that could apply to each verification and validation metric for RPO and SDDA.

2.4.1 Designing Good Requirements

In *Design for Safety*, authors Louis J. Gullo and Jack Dixon explain the objective of system requirements: "A system requirement specifies what a system must do to satisfy a customer need or goal ... Requirements capture customer objectives, expectations, desires, limitations, and constraints" [12]. To this end, a complete set of requirements for a system must contain directives for every task within its scope, any constraints that affect the way it must go about completing these tasks, and the minimum standards of performance it must achieve in doing so. To ensure that system requirements satisfy these aforementioned needs, Gullo and Dixon present a list of attributes that well written requirements must possess [12]:

- Clear — The meaning of the requirements shall be unambiguous
- Complete — The requirements shall cover everything relevant to what is being designed
- Consistent — The requirements shall not conflict with each other
- Correct — The requirements shall align with actual needs
- Feasible — It shall be possible to satisfy the requirements
- Objective — Each requirement shall have only one possible interpretation
- Need-Oriented — The requirements shall state what is needed, not the solution
- Singular — Each requirement shall focus on only one need
- Succinct — The requirements shall be as concise as possible
- Verifiable — It shall be possible to quantitatively demonstrate that each requirement is satisfied

Furthermore, systems engineers should be careful when writing negative requirements (i.e., "the system shall not...") rather than positive requirements (i.e., "the system shall...").

Gullo and Dixon further break down requirements into six types: operational, functional, performance, design, derived, and allocated [12]. Operational requirements focus on the basic objectives of the system, as well as any constraints that may apply to it and the environment in which it operates. For example, in the case of a flight data recorder for a commercial aircraft, a reasonable operational requirement could be, "the system shall permit recovery of flight data for a commercial flight to aid in accident reconstruction." Functional requirements detail necessary tasks that must be completed to satisfy these objectives (e.g., "the system shall continuously broadcast its location to ground controllers during normal operations"). Performance requirements dictate the minimum performance and reliability thresholds of the system (e.g., "the system shall operate continuously from the moment the aircraft begins its takeoff roll until the moment it concludes its landing roll"). Design requirements specify how the system needs to be designed or built (e.g., "the system shall broadcast all signals at a frequency of [X] MHz"). Derived requirements, which are implied from higher-level requirements, and allocated requirements, which apportion a higher level requirement into multiple lower-level requirements, are outside the scope of this thesis.

Appendix C of the NASA Systems Engineering Handbook, which details how to write good requirements, agrees with Gullo and Dixon, and adds that systems engineers should be careful to distinguish between needs and wants by asking, "what is the worst that could happen if the requirement was not included?" [13]. Additionally, any requirement should be traceable to either a higher-level requirement or the mission/system-of-interest scope.

2.4.2 Requirements for Verification Metrics

From INCOSE's definition of verification, verifying RPO and SDDA involves comparing the outputs of a computerized implementation of each tool with those given by the tools in their conceptual forms. For most analysis software, this generally involves designing an example problem simple enough that a design engineer can calculate the results without running the computerized implementation [14]. Thus, the verification metrics for RPO and

SDDA should apply to scenarios where the outputs of each tool can be easily predicted by hand, and should describe how well each tool replicates these results. For RPO, this output would be the portfolio of subsystems selected at each value of cost for each cost point considered. The SDDA outputs, meanwhile, would consist of the beginning and end development times for each subsystem.

Because the verification process involves comparing the output of each tool with a known result generated by the conceptual model, any deviation from this result should be considered a partial failure of the computerized implementation. As such, it doesn't matter what type of error occurs (in the case of RPO, the errors would be a false inclusion or exclusion of a subsystem), or if the error generates a more optimistic or pessimistic result than the conceptual model (such as an earlier or later beginning development time, in the case of SDDA); the metrics should make no distinction between them.

As mentioned above, the output of interest for the RPO verification process is the portfolio of subsystems chosen for each cost value considered in an example system. We can quantify the generation of this portfolio as a set of decisions made by RPO at each cost point, where a decision can be whether to include or exclude a particular subsystem, and how many instances of the subsystem should be selected should RPO include it. As such, the verification metric for RPO should consider all of the decisions made for each portfolio, as well as how many of them are correct. Furthermore, when generating portfolios for multiple cost values, RPO doesn't inherently consider any one to be more important than the others. Therefore, the verification metric shouldn't allow a single portfolio to have a disproportionate influence on the value of the metric. Additionally, since RPO can be applied to systems containing a large number of subsystems and to an indefinitely large number of cost values, the number of decisions made in a single run of RPO can vary wildly. The verification metric should therefore be normalized to ensure consistency across a number of example problems. Otherwise, a particular value of the verification metric could have a number of different meanings depending on the context of the problem. For example, without such context, a systems engineer conducting a verification test of RPO would have no idea whether a verification metric of 417 was good or bad. A normalized metric, by contrast, requires no *a priori* knowledge of the scenario used in the verification process: A value of 1 indicates that

the computerized implementation of RPO aligns perfectly with the conceptual model, while a value of 0 indicates complete unalignment. Table 2.1 lists the requirements for an RPO verification metric.

Table 2.1. Requirements for RPO Verification Metrics

Requirement	**Type**
The metrics shall describe how well RPO generates portfolios of subsystems in a scenario where the expected solutions can be easily predicted	Operational
The metrics shall incorporate how many correct decisions RPO makes in generating each portfolio	Functional
The metrics shall incorporate how many total decisions RPO makes in generating each portfolio	Functional
The metrics shall prevent a large deviation in a single portfolio's generation from strongly influencing the total value of the metrics	Performance
The metrics shall be normalized to take value between 0 and 1, where 0 refers to low effectiveness and 1 to high effectiveness	Design
The metrics shall make no distinction between a false inclusion and a false exclusion	Design

In contrast with RPO, SDDA has two outputs that already exist in quantitative terms: beginning time and end time of subsystem development. When verifying an implementation of SDDA, any deviation from the beginning and end times given by the conceptual model for each subsystem can be considered an error. As such, the verification metric for SDDA should take into account any deviation in the computerized implementation from the subsystem development beginning and end times given by the easily predicted solution. While the

outputs for stochastic SDDA include probability distributions, only the mean value need be considered when comparing it to the known solution of an example problem. Moreover, neither beginning development time or end development time should be treated as more important than the other. Additionally, since SDDA doesn't treat the development of any one subsystem as more important than any other, the verification metric shouldn't allow a deviation in a single subsystem's development to have a disproportionate influence on the value of the metric. Because SDDA outputs correspond with an unbounded physical quantity (time) and can incorporate an arbitrarily large number of subsystems, the sum of all deviations from the expected beginning and end times has no upper limit. As such, like with RPO, the verification metric for SDDA should be normalized. Table 2.2 lists the requirements for an SDDA verification metric.

2.4.3 Requirements for Validation Metrics

In contrast with verification, which involves comparing the results of a test case with those of the conceptual implementation of the tool, validation involves comparing the outputs of a predictive analysis using the tool in question with the behavior, development, or composition of the physical system. Thus, the validation metrics for RPO and SDDA need not only apply to highly simplified example problems, but to any application corresponding to a physical system. The output of interest for RPO would be the portfolio of subsystems selected a particular cost point and its characteristics (subsystem cost and performance) — which differs from the output of interest for the verification process, which consists of the portfolios generated at every value of cost considered in the example problem. The SDDA outputs of interest, meanwhile, are unchanged from the verification process: the beginning and end development times for each subsystem.

Similar to the verification process, because the validation process involves comparing the output of each tool with a known result, the metrics should treat any deviation from the characteristics of the physical system as an undesirable result, regardless of what direction the error falls in. For example, while an overestimate of the development time of each subsystem would result in the physical system developing faster than expected — which most design

Table 2.2. Requirements for SDDA Verification Metrics

Requirement	**Type**
The metrics shall describe how well SDDA produces a design schedule where the expected solution can be easily predicted	Operational
The metrics shall incorporate the deviation from the expected beginning time of subsystem development	Functional
The metrics shall incorporate the deviation from the expected end time of subsystem development	Functional
The metrics shall prevent a large deviation in a single subsystem's development from strongly influencing the total value of the metrics	Performance
The metrics shall place equal weight on the beginning and end development times	Design
The metrics shall be normalized to take value between 0 and 1, where 0 refers to low effectiveness and 1 to high effectiveness	Design
The metrics shall make no distinction between a subsystem developing before expected and a subsystem developing after expected	Design

engineers would consider a good thing — it still means that the SDDA predictions were inaccurate, and the validation metric should penalize them accordingly. Furthermore, since — like with the verification process — the deviations from the outputs given by each tool can vary considerably depending on the problem, the validation metrics should be take value between 0 and 1.

In the case of RPO, we have two outputs of interest for the validation process: subsystem cost and performance. We can't assume that one of these outputs has more importance than the other, so the validation metric for RPO should weight both of them equally. Cost is a quantitative variable, so the validation metric should take into account the difference between the cost of each subsystem predicted by RPO and that of the actual physical subsystem. Performance, meanwhile, can be treated as a Boolean variable: Is each subsystem capability satisfied? As such, the validation metric should incorporate both the total number of subsystem capabilities and the number that the physical system satisfies. Since the Bertsimas-Sim method models subsystem capabilities as following uniform, symmetrical probability distributions [17], the validation metric should only count a subsystem capability as unsatisfied if it falls outside that distribution. Moreover, since RPO doesn't treat any one subsystem or subsystem capability as more important than any other, the validation metric should prevent any one deviation in cost or performance from disproportionately impacting the value of the metric. Table 2.3 lists the requirements for an RPO validation metric.

Table 2.3. Requirements for RPO Validation Metrics

Requirements	**Type**
The metrics shall describe how well RPO predicts the cost and performance of a portfolio of subsystems for a physical system	Operational
The metrics shall incorporate how many pertinent subsystem capabilities are met	Functional
The metrics shall incorporate the deviation from the expected cost of each subsystem	Functional
The metrics shall prevent a large deviation in a single subsystem's performance from strongly influencing the total value of the metrics	Performance

Continued on next page

Table 2.3 – continued from previous page

Requirements	Type
The metrics shall prevent a large deviation in a single subsystem's cost from strongly influencing the total value of the metrics	Performance
The metrics shall place equal weight on the cost and performance of the portfolio	Design
The metrics shall be normalized to take value between 0 and 1, where 0 refers to low effectiveness and 1 to high effectiveness	Design
The metrics shall incorporate the uncertainty in the pertinent subsystem capabilities	Design
The metrics shall make no distinction between a cost overrun and a cost below expected	Design
The metrics shall make no distinction between a performance overestimate and a performance underestimate	Design

As with the verification metric for SDDA, the validation metric for SDDA should take into account how the beginning and end development times of the physical system differ from those output by the analysis tool, since both outputs are quantitative variables. The validation metric also shouldn't allow a deviation in a single subsystem's development to have a disproportionate influence on the value of the metric, since SDDA doesn't treat the development of any one subsystem as more important than any other, and the beginning and end times should be weighted equally. However, unlike with verification, validation assumes that uncertainty will always be present in the model, and the validation process must take this into account [19]. Since deterministic SDDA does not incorporate uncertainty in any

Table 2.4. Requirements for SDDA Validation Metrics

Requirement	**Type**
The metrics shall describe how well SDDA predicts the design schedule of a physical system	Operational
The metrics shall only apply for a stochastic SDDA simulation	Operational
The metrics shall incorporate the deviation from the expected time of subsystem development	Functional
The metrics shall incorporate how close each subsystem begins and ends development relative to the expected value	Functional
The metrics shall prevent a large deviation in a single subsystem's development from strongly influencing the total value of the metrics	Performance
The metrics shall place equal weight on the beginning and end development times	Design
The metrics shall be normalized to take value between 0 and 1, where 0 refers to low effectiveness and 1 to high effectiveness	Design
The metrics shall make no distinction between a subsystem developing before expected and a subsystem developing after expected	Design

capacity [11], the validation metric must apply exclusively to stochastic SDDA. Table 2.4 lists the requirements for an SDDA validation metric.

2.5 Effectiveness Metrics

Now that we have defined requirements for the RPO and SDDA verification and validation metrics, we can explore candidate metrics from which to select one metric for each tool and process. For verification specifically, Oberkampf and Trucano propose taking a straightforward error measurement, as shown in Eq. 2.4, where u_{exact} and $u_{discrete}$ are scalar fields representing the mathematically correct solution and that output by the computerized tool, respectively [14]. However, this error measurement is not normalized, so it doesn't satisfy our requirements for verification metrics. Nevertheless, it does provide a good starting point for generating verification metrics, as it considers the total error for the entire simulation at once.

$$V = u_{exact} - u_{discrete} \tag{2.4}$$

One way to normalize these errors would be to frame the verification process as a regression problem, where the independent variable comprises the mathematically correct outputs, and the dependent variable comprises the outputs of the computerized model. The coefficient of determination — or R^2 — is a normalized metric that divides the sum of squared residuals (the residuals in this case being the computerized outputs minus the outputs of the conceptual model) by the total sum of squares, as shown in Eq. 2.5. This solves the problem of normalizing the errors, but has the potential to cause one data point to disproportionately affect the metric, since the denominator depends on the arithmetic mean. For example, if the subsystems in an RPO portfolio all have a cost between \$10,000 and \$50,000, except for one which has a cost of \$10,000,000, the latter subsystem will have a disproportionate impact on R^2, to the point where the other subsystems can deviate wildly in cost from their expected values with negligible impact. Additionally, because $R^2 = 1$ whenever the data follow a linear trend, regardless of what that trend is, the metric could potentially equal 1 even if the output of the computerized model is biased. For example, if the beginning and end times output from SDDA equaled exactly 0.9 times what the corresponding times were

for the actual development schedule, the validation metric would still equal 1, since the data are perfectly linear.

$$V = R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - \hat{y}_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y}_i)^2} \tag{2.5}$$

To resolve these issues, we turn to another metric presented by Oberkampf and Trucano, which they suggest for use in validation but can apply to a verification metric as well [14]. Eq. 2.6 presents a weighted sum of hyperbolic tangents, with the error in the numerator and the output quantity in the denominator. The y_i corresponds to the mathematically correct output quantity for the verification metric and the physical system characteristic for the validation metric, while $\hat{y}_i$ corresponds to the output of the computerized model in either case. This metric avoids both pitfalls of the coefficient of determination, as $\hat{y}_i$ doesn't depend on a regression model and each term in the summation doesn't depend on any others. The one weakness of this metric is that if $y_i = 0$, the absolute value of the hyperbolic tangent equals 1, regardless of the value of the numerator. Since the quantitative outputs of RPO and SDDA are all positive real numbers, we can easily fix this by adding an arbitrarily small number to the denominator.

$$V = 1 - \frac{1}{n}\sum_{i=1}^{n} \tanh \left| \frac{\hat{y}_i - y_i}{y_i} \right| \tag{2.6}$$

When conducting validation tests involving probability distributions, we may not want the validation metric to penalize deviations from the output values of each tool if the true values are relatively close to the mean. To that end, we define another potential validation metric in Eq. 2.7, consisting of a weighted sum of significance tests, where Δy_i is the variable corresponding to the magnitude of the deviation in the output variable. α refers to the significance level, which the demonstration cases in Chapter 4 will assume is equal to 0.05.

$$V = \frac{1}{n}\sum_{i=1}^{n}(P(\Delta y_i > \mid y_i - \hat{y}_i \mid) > \alpha) \tag{2.7}$$

All of the aforementioned metrics assume that the outputs of the tools are quantitative variables. However, the decisions made during portfolio generation and the satisfaction of

subsystem capabilities in RPO are categorical variables, and so we need to define another effectiveness metric for them. If we consider these variables to be Boolean values (e.g., a correct decision has a value of 1, and an incorrect decision has a value of 0), we can define the variable D_C as the sum of these Boolean variables, and D_T as the total number of Boolean variables (if all the Boolean variables are equal to 1, then $D_C = D_T$). To create a normalized effectiveness metric, we can divide D_C by D_T, as shown in Eq. 2.8.

$$V = \frac{D_C}{D_T} \tag{2.8}$$

Table 2.5 summarizes these effectiveness metrics, their advantages and disadvantages, and their best uses.

2.5.1 RPO Effectiveness Metrics

The variable of interest in an RPO verification test is the set of decisions made in generating a portfolio. These decisions include whether a subsystem is included or excluded, and the number of instances of a particular subsystem should it be included. In the latter case, a decision can be treated as correct if the number of instances selected by the computerized implementation exactly matches the mathematically correct solution (this decision is absent entirely if the subsystem is meant to be excluded). Since the decisions made are Boolean variables, we use the effectiveness metric shown in Eq. 2.8. Eq. 2.9 presents a slight variation of that, where the verification metric consists of a weighted sum of multiple instances of Eq. 2.8. n_p refers to the number of portfolios considered in the verification test.

$$V_R = \frac{1}{n_p} \sum_{\mathrm{i}=1}^{n_p} \frac{D_C}{D_T} \tag{2.9}$$

The validation metric considers both subsystem cost — a quantitative variable — and the satisfaction of subsystem capabilities — a categorical variable. Thus, we frame the validation metric as the average of the submetrics for cost and performance, as shown in Eq. 2.10.

$$R = \frac{P + C}{2} \tag{2.10}$$

Table 2.5. Summary of Candidate Effectiveness Metrics

Metric	Metric Type	Advantages	Disadvantages	Best Uses
Error Measurements [14]	Verification	Accounts for all deviations in a single scalar field	Not normalized	Inappropriate for this work
Coefficient of Determination	Verification, Validation	Normalized; established measurement in statistics	Data points can have outsize influence on the metric; biased yet linear data leads to inaccurate results	Inappropriate for this work
Weighted Sum of Hyperbolic Tangents [14]	Verification, Validation	Normalized; each contribution dependent only on itself	Not applicable for categorical variables; punishes any deviation regardless of size	RPO validation (cost), SDDA verification
Ratio of Sums of Boolean Variables	Verification, Validation	Normalized; applicable to categorical variables	Not applicable for quantitative variables	RPO verification, RPO validation (performance)
Weighted Sum of Significance Tests	Verification, Validation	Normalized; useful if small deviations considered acceptable	Not applicable if uncertainty not given or for categorical variables	SDDA validation

As with the verification metric, the performance validation submetric, shown in Eq. 2.11 consists of a weighted sum of multiple instances of Eq. 2.8, where n is the number of subsys-

tems in a given portfolio, SC_{i} refers to the total number of subsystem capabilities for each subsystem, and $SC_{\mathrm{i},sat}$ refers to the number of those capabilities that are satisfied. Since the subsystem capabilities follow a symmetric uniform uncertainty distribution, we consider a capability satisfied if it falls within the distribution given in the RPO implementation.

$$P = \frac{1}{n}\sum_{\mathrm{i}=1}^{n}\frac{SC_{\mathrm{i},sat}}{SC_{\mathrm{i}}} \tag{2.11}$$

The cost submetric, meanwhile, corresponds to the hyperbolic tangent metric given in Eq. 2.6. To avoid division by zero, as some RPO selections regarding configuration could have zero marginal cost, we add a small constant — \$0.001 for our demonstration cases, cheaper than any real subsystem with a nonzero cost — to the denominator, as shown in Eq. 2.12.

$$C = 1 - \frac{1}{n}\sum_{\mathrm{i}=1}^{n}\tanh\left|\frac{E(C_{\mathrm{i}}) - C_{\mathrm{i}}}{C_{\mathrm{i}} + 0^{+}}\right| \tag{2.12}$$

2.5.2 SDDA Effectiveness Metrics

The variables of interest in an SDDA verification test are the beginning time and the end time of subsystem development. Both of these variables are quantitative, and the requirements for SDDA verification and validation dictate that the effectiveness metrics should place equal weight on them. As such, Eq. 2.13 presents the average of the hyperbolic tangent metrics for the beginning development time and the end development time. By definition, at least one of the beginning times must be equal to zero, so we add an arbitrarily small variable to the denominator (0.000001 should be sufficiently small as to avoid making a large impact on the metric regardless of the time units used). Since a subsystem cannot have a total development time of zero, none of the end times can equal zero, so this metric does not need to add an arbitrarily small constant to the corresponding denominator.

$$V_S = 1 - \frac{1}{2n}\sum_{\mathrm{i}=1}^{n}\tanh\left|\frac{E(t_{b,\mathrm{i}}) - t_{b,\mathrm{i}}}{t_{b,\mathrm{i}} + 0^{+}}\right| - \frac{1}{2n}\sum_{\mathrm{i}=1}^{n}\tanh\left|\frac{E(t_{\mathrm{e},\mathrm{i}}) - t_{\mathrm{e},\mathrm{i}}}{t_{\mathrm{e},\mathrm{i}}}\right| \tag{2.13}$$

In contrast with the verification metric, the validation metric for SDDA explicitly applies to stochastic SDDA, so the metric should take into account whether the beginning and end times for the physical system significantly differ from the expected value. Thus, the validation metric presented in Eq. 2.14 is the average of the stochastic validation metrics shown in Eq. 2.7 for the beginning and end development times. $\Delta t_{b,\mathrm{i}}$ and $\Delta t_{\mathrm{e,i}}$ are variables corresponding to the magnitude of the deviations in the actual beginning and end development times, respectively, from the outputs given by an SDDA validation test.

$$S = \frac{1}{2n}(\sum_{\mathrm{i}=1}^{n}((P(\Delta t_{b,\mathrm{i}} >\mid t_{b,\mathrm{i}} - E(t_{b,\mathrm{i}}) \mid) > \alpha) + (P(\Delta t_{\mathrm{e,i}} >\mid t_{\mathrm{e,i}} - E(t_{\mathrm{e,i}}) \mid) > \alpha)) \tag{2.14}$$

Both the verification and validation metrics for SDDA apply to standard SDDA and $\mathrm{SDDA}_{\mathrm{Max}}$, as the metrics don't take into account the conservatism factor that differentiates the two.

2.6 Comparison of RPO and SDDA Metrics

While the effectiveness metrics for RPO and SDDA share many similarities, they are not interchangeable with one another. Since the outputs of RPO include both categorical and quantitative variables, while the outputs of SDDA are exclusively quantitative, the effectiveness metrics for the two tools rely on different classes of metric: RPO makes use of weighted sums of Boolean variable ratios for its verification metric and performance validation metric; the SDDA validation metric relies on a weighted sum of significance tests; and both tools have effectiveness metrics that rely on hyperbolic tangent relations (the verification metric for SDDA, and the cost validation metric for RPO). As per the requirements given earlier in Chapter 2, the verification and validation metrics for both RPO and SDDA are normalized to take value between 0 and 1, where 1 indicates that the analysis tool functions perfectly. This means that systems engineers developing a hypersonic vehicle with both tools could evaluate the effectiveness metrics for each tool, and determine whether the analysis tools functioned as intended using the same set of numerical standards.

This thesis considers the use of RPO and SDDA individually, and not as part of an overarching design process involving the entire AWB. However, future research involving these effectiveness metrics — particularly if we consider SODA as well — should consider the possibility of developing a unified effectiveness framework for the AWB. Such a framework would take a hypersonic vehicle based on a model analyzed using the AWB, and use a single set of effectiveness metrics to determine if the suite of tools within the AWB can analyze such a system with a high degree of accuracy.

3. APPROACH

In Chapter 2, we developed quantitative measures to evaluate the effectiveness of RPO and SDDA in the verification and validation process. We can now apply these measures to example problems to demonstrate their effectiveness in evaluating hypersonic vehicle architectures. Before we begin, however, we need to discuss how to construct models of hypersonic vehicles within the tools for the verification and validation tests, and to what we will compare their outputs.

3.1 Model Construction

In order to construct a model of a hypersonic vehicle to serve as the demonstration cases for the AWB effectiveness metrics, we need a hypersonic vehicle on which to base the model. Given that most hypersonic vehicles that have been constructed to date have been military projects — funded by the United States, Russia, China, and the European Union, among other national and international governmental bodies [20][21] — publicly available information regarding hypersonic vehicle architectures, costs, and precise development schedules is extremely scarce. In an effort to give engineers and researchers without access to ITAR-restricted data the ability to develop and analyze conceptual designs for hypersonic vehicles, the Air Force Research Laboratory (AFRL) began developing a family of generic hypersonic vehicles (GHVs) in 2012 [22]. These vehicles do not contain any proprietary or sensitive information regarding real-world hypersonic programs, but provide a basic set of mission profiles and top-level subsystems from which we can develop the basic architectures for our demonstration cases.

However, as the name implies, the GHV family of hypersonic vehicles does not have any overarching mission objectives beyond conducting a few basic maneuvers and surviving long enough to descend [22]. Therefore, the demonstration cases in this thesis rely on the system capabilities of the Rockwell X-30, also known as the National Aerospace Plane (NASP). The NASP began development in 1986 — though proposals for a similar hypersonic vehicle began as early as 1958 [23] — as a single-stage-to-orbit spacecraft, with the possibility of carrying crews to and from space. Due to cost concerns, the program was defunded in 1993, and the

NASP was never built; however, a program review published by the RAND Corporation in 1993 lists the intended system capabilities of the proposed vehicle [23]:

- Insert one or more satellites into orbit
- Inspect and retrieve satellites from orbit
- Repair satellites in orbit
- Conduct a resupply mission to a space station
- Carry crew to and from a space station

While the report did conduct a cost analysis for the spacecraft as a whole — focusing specifically on the cost per pound of payload into orbit — it did not break down the costs per subsystem. As such, the costs presented in the demonstration cases are purely illustrative — as are the relative cost differences between subsystems of the same type within RPO — and are not meant to be indicative of the actual subsystem development costs of a hypersonic vehicle. The same applies to the performance specifications of each subsystem, as well as the development timeline.

3.2 RPO Demonstration Cases

Based on the AFRL GHV and the NASP, the RPO validation model for a hypersonic vehicle breaks down the library of top-level subsystems into several distinct groups: propulsion systems, flight computers, thermal protection systems, power systems, sensors, communications systems, aerodynamic bodies, and structural systems. The propulsion systems are further broken down into hypersonic combustors, inlets, nozzles, subsonic propulsion systems, and engine integration systems, while the thermal protection systems are divided into active and passive systems, and the structures are divided into internal structures and skin. Fig. 3.1 displays a list of the propulsion subsystems for the validation case within the RPO input model, as well as their respective costs. As mentioned above, the costs for these demonstration cases are not meant to be indicative of the actual costs of developing a hypersonic vehicle.

Propulsion Systems	Dual-Mode Ramjet-Scramjet Combustor	$7,000,000.00
	Ramjet Combustor	$2,000,000.00
	Scramjet Combustor	$2,500,000.00
	Axisymmetric Inlet	$500,000.00
	Waverider Inlet	$1,000,000.00
	Axisymmetric Nozzle	$500,000.00
	Single Expansion Ramp Nozzle	$1,000,000.00
	Booster Rocket	$1,000,000.00
	Gas Turbine Engine	$750,000.00
	Podded Engine	$500,000.00
	Under-Fuselage Engine	$100,000.00
	Wing-Body Juncture Engine	$750,000.00

Figure 3.1. Propulsion subsystems available for selection by RPO, along with their respective costs, for the validation demonstration case.

The hypersonic vehicle model used for the verification case is similar to that used for the validation case, though with significant simplification. Certain classes of subsystems, such as the propulsion subsystems, have been consolidated (e.g., the hypersonic combustors, inlets, and nozzles have all been combined into hypersonic engine subsystems), and there are no more than two possible selections per type of subsystem. Additionally, the costs, output requirements, capabilities, and uncertainties associated with each subsystem have been defined such that certain selections are obvious at different cost points, in order to assist in generating a conceptual output to which to compare the computerized implementation. The verification demonstration case will also set the conservatism parameter Γ defining limits on acceptable risk to zero, so that RPO will always try to maximize performance within cost constraints without regard to the uncertainties in capability and support requirements for the subsystems.

3.3 SDDA Demonstration Cases

In contrast with the RPO demonstration cases, where the number of subsystems differs between the verification and validation cases, the SDDA models have the same number of subsystems for the verification case and the validation case. The SDDA model consists of the following 12 subsystems:

- Combustor
- Inlet
- Nozzle
- Subsonic Propulsion
- Aerodynamic Body
- Internal Structures
- Skin
- Thermal Protection
- Flight Control Computer
- Sensors
- Communications
- Power

These subsystems align with those in the RPO validation model, but are expressed in generic terms (i.e., the subsystems in the SDDA model do not align with one selection from RPO in particular). Where the verification and validation cases differ is in the connections between the subsystems: the verification case removes a large number of dependencies from the validation case, such that each subsystem is developmentally dependent on no more than two others. Furthermore, the punctuality P_{i} for all subsystems in the verification case has been set to 100, so the relationship between the beginning time of the dependent subsystem and the end time of the input subsystem can be expressed using Eq. 3.1.

$$t_b^{\mathrm{j}} = t_c^{\mathrm{i}} - t_{min}^{\mathrm{j}}(1 - SOD_{\mathrm{ij}}) \tag{3.1}$$

This also means that the only subsystems that develop in more time than the given value of t_{min}^{j} are those that are dependent on two or more other subsystems and begin their

development at the average output beginning time from these relationships, in the case of standard SDDA.

For simplicity, both the verification and validation cases characterize each connection between subsystems with one of three preset values for each parameter: The SOD can equal 0.1, 0.4, or 0.9, and the COD can equal 10, 40, or 90. Additionally, the demonstration cases will be applied to both standard SDDA and SDDA_{Max}, as the effectiveness metrics for verification and validation don't consider the assumption of conservatism (or lack thereof).

4. DEMONSTRATION CASES

4.1 Overview

The demonstration cases presented in this thesis are hypersonic vehicle models to which we apply the four effectiveness metrics generated in Chapter 2. The top-level subsystems for the vehicle model are derived from the AFRL GHV [22], a class of vehicles developed to allow engineers to conduct conceptual design analyses on hypersonic vehicles without the restrictions associated with designs subject to ITAR restrictions. These subsystems form the basis of the model used in SDDA analysis, as well as the types of subsystems that can be selected to generate a portfolio in RPO. The cruise phase of the mission profiles for vehicles under the umbrella of the GHV consists of performing basic flight maneuvers, and we wish to generate a more comprehensive set of system capabilities for RPO analysis. These demonstration cases use the mission objectives of the NASP as system capabilities in RPO, which include satellite insertion into orbit, satellite inspection and retrieval, on-orbit satellite repair, space station resupply, and space station crew replacement [23].

4.2 RPO Verification Results

The verification test for RPO uses a simplified library of hypersonic vehicle subsystems to generate portfolios, where each class of subsystem contains two possible candidates, one of which is clearly preferable — either by having a lower cost, better subsystem capabilities, or both — to the other at a particular cost point. For each subsystem class, only one instance need be selected. Table 4.1 displays the eight classes of subsystem used in this demonstration case and the candidate subsystems within them, and indicates which subsystem in each class has a lower cost and a higher performance. For four of the subsystem classes — flight computer, sensors, communication, and structures — the high-performance subsystem also has the lowest cost, and should thus be selected for every generated portfolio. The variation between portfolios, therefore, should be a function of the other four subsystem classes. The demonstration case sets the value of the conservatism parameter Γ to zero, so that the main contributor to the variation between portfolios will be subsystem cost.

Table 4.1. Subsystem Library for the RPO Verification Test

Subsystem Class	Low Performance Option	High Performance Option
Propulsion	Scramjet (Low Cost)	Dual-Mode Ramjet-Scramjet (High Cost)
Flight Computer	Power-Efficient Weak Computer (High Cost)	Power-Inefficient Strong Computer (Low Cost)
Thermal Protection	Flexible TPS Blanket (Low Cost)	Carbon Ceramic Composites (High Cost)
Power	Chemical Fuel Cell (Low Cost)	Gas Turbine APU (High Cost)
Sensors	Power-Efficient Weak Radar (High Cost)	Power-Inefficient Strong Radar (Low Cost)
Communications	Power-Efficient Weak Antenna (High Cost)	Power-Inefficient Strong Antenna (Low Cost)
Aerodynamic Body	Tapered Wing/Wedge Tail (Low Cost)	Waverider (High Cost)
Structures	Steel Structure (High Cost)	Composite Structure (Low Cost)

For the four competitive subsystem classes, the portfolio with the lowest cost would contain the scramjet, flexible TPS blanket, chemical fuel cell, and tapered-wing/wedge tail body. The cost differences between these subsystems and their high-cost counterparts are largest for the aerodynamic body ($3,000,000) and propulsion subsystem ($2,000,000), followed by the thermal protection ($1,000,000) and power ($700,000). Thus, as the cost increases, RPO should select the high-cost options for the power, thermal protection, propulsion, and aerodynamic body, in that order. Additionally, among the four subsystem classes, the propulsion subsystem satisfies the most subsystem capabilities — and thus contributes more to the satisfaction of the overall system capabilities — followed by the aerodynamic body, thermal protection, and power. The demonstration case also defines the support input requirements for the aerodynamic body such that the high-cost option can't be selected without the high-cost option for the propulsion subsystem. Thus, when the cost constraint becomes high enough for the high-cost options of the propulsion subsystem and aerodynamic body to be selected, RPO should switch back to the low-cost options for the thermal protection and power subsystems until the cost constraint increases enough to allow the high-cost options for the latter two to be selected again. The same applies to the power subsystem when the cost constraint increases to the point where RPO can select the high-cost thermal protection,

as the latter satisfies more subsystem capabilities than the former. This results in 12 possible portfolios that RPO should select.

Table 4.2. Portfolio Selection for the RPO Verification Test

Subsystem	**Portfolio (# Instances Selected)**											
	1	2	3	4	5	6	7	8	9	10	11	12
Dual-Mode Ramjet-Scramjet	0	0	0	0	1	1	1	1	1	1	1	1
Scramjet	1	1	1	1	0	0	0	0	0	0	0	0
Power-Efficient Weak Computer	0	0	0	0	0	0	0	0	0	0	0	0
Power-Inefficient Strong Computer	1	1	1	1	1	1	1	1	1	1	1	1
Flexible TPS Blanket	1	1	0	0	1	1	0	0	1	1	0	0
Carbon Ceramic Composites	0	0	1	1	0	0	1	1	0	0	1	1
Chemical Fuel Cell	1	0	1	0	1	0	1	0	1	0	1	0
Gas Turbine APU	0	1	0	1	0	1	0	1	0	1	0	1
Power-Efficient Weak Radar	0	0	0	0	0	0	0	0	0	0	0	0
Power-Inefficient Strong Radar	1	1	1	1	1	1	1	1	1	1	1	1
Power-Efficient Weak Antenna	0	0	0	0	0	0	0	0	0	0	0	0
Power-Inefficient Strong Antenna	1	1	1	1	1	1	1	1	1	1	1	1
Steel Structure	0	0	0	0	0	0	0	0	0	0	0	0
Composite Structure	1	1	1	1	1	1	1	1	1	1	1	1
Total Cost ($100,000)	214	221	224	231	234	241	244	251	264	271	274	281

Table 4.2 lists the generated portfolios for the RPO verification demonstration case. The simulation was run with a conservatism parameter of zero, and a cost interval small enough that RPO wouldn't skip over optimal portfolios at a given cost point. As the table shows, the 12 generated portfolios match the optimal selections discussed above, so all the decisions regarding each subsystem — whether to include it, and how many to include if so — are

correct. Thus, the verification metric $V_R = 1$. Fig. 4.1 visualizes these results in a Pareto frontier.

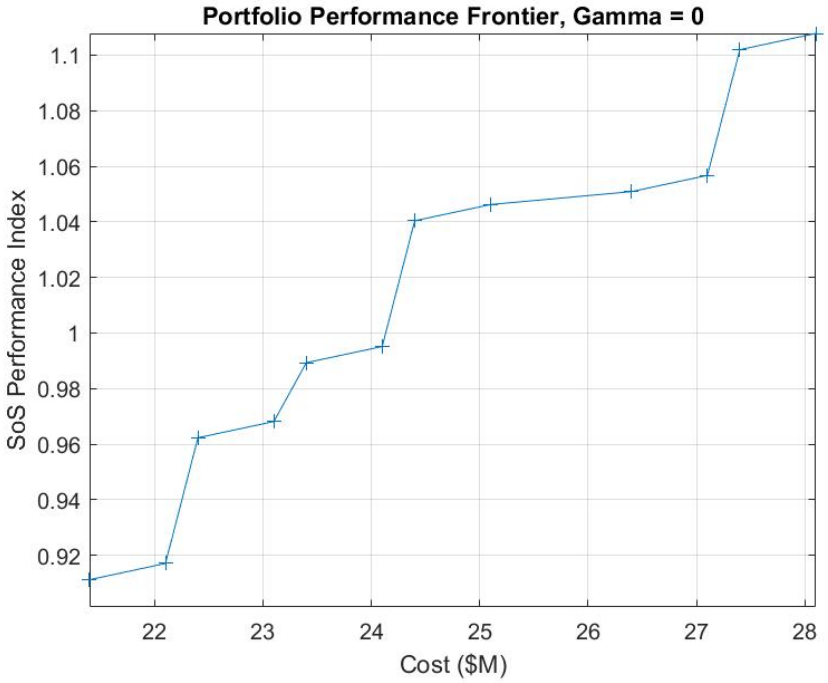

Figure 4.1. Pareto frontier for the RPO verification test, where the conservatism parameter equals zero. The Pareto frontier plots the unitless measure of overall system (or SoS, as RPO was initially designed as an SoS analysis tool) capability against the cost of the system, in millions of dollars.

4.3 RPO Validation Results

The demonstration case for RPO validation assumes that, after running an RPO analysis for the top-level subsystem model of the hypersonic vehicle, the design engineers selected the portfolio generated with a cost of $23,100,000 for further development. To explore how deviations in subsystem cost and performance affect the validation metric, we examine two scenarios regarding the final design of the hypersonic vehicle: one with low deviations from the RPO-predicted values, and another with high deviations. For the high-deviation vehicle, the deviations in the subsystem cost and subsystem capabilities will be exactly double those of the low-deviation vehicle, so the effects of these deviations on the validation metric will

be evident. Table 4.3 lists the subsystems within the selected portfolio, as well as their predicted and actual costs.

Table 4.3. Generated Portfolio and Subsystem Costs for the RPO Validation Test

Subsystem Type	**Selected Subsystem**	**Predicted Cost ($)**	**Actual Cost (Low Deviation, $)**	**Actual Cost (High Deviation, $)**
Hypersonic Combustor	Dual-Mode Ramjet-Scramjet	7,000,000	8,120,000	9,240,000
Inlet	Axisymmetric Inlet	500,000	480,000	460,000
Nozzle	Axisymmetric Nozzle	500,000	510,000	520,000
Subsonic Propulsion	Booster Rocket	1,000,000	1,140,000	1,280,000
Engine Integration	Wing-Body Juncture	750,000	830,000	910,000
Flight Computer	Power-Inefficient Strong Computer	300,000	260,000	220,000
Passive Thermal Protection	Dielectric Composites	1,000,000	970,000	940,000
Active Thermal Protection	Transpiration Cooling	2,000,000	2,350,000	2,700,000
Power	Chemical Fuel Cell	800,000	810,000	820,000
Sensors	Power-Inefficient Strong Radar	300,000	340,0000	380,000
Communications	Power-Inefficient Strong Antenna	150,000	140,000	130,000
Aerodynamic Body	Delta Wing, Forward Canards	8,000,000	8,550,000	9,100,000
Internal Structures	Steel Structure	500,000	470,000	440,000
Skin	Superalloy Skin	300,000	320,000	340,000
Total		**23,100,000**	**25,290,000**	**27,480,000**

Since the Bertsimas-Sim formulation of RPO does not assume any uncertainty in subsystem cost [17], any deviation in cost from the value given in RPO, no matter how small, will affect the cost validation submetric. This is not the case for the subsystem capabilities, where each capability is characterized by a symmetric, uniform uncertainty distribution (it's important to note that this uncertainty is treated as a confidence interval, rather than a for-

mal probability distribution function). For this demonstration case, we set the uncertainties for each capability to be either 20% or 30% of the capability's value. Tables 4.4-4.17 list the capabilities that apply to each subsystem, as well as their predicted values, uncertainties, and actual values.

Table 4.4. Subsystem Capabilities for the Hypersonic Combustor

Capability (Parameter, Unit)	**Predicted Value**	**Actual Value (Low Deviation)**	**Actual Value (High Deviation)**
Maintain Hypersonic Speeds (Thrust, kN)	6 ± 1.2	7	8
Limit Temperature Increase of Subsystems (Temperature Reduction, K)	50 ± 10	42	34
Provide Electrical Power (Power, W)	7500 ± 1500	7000	6500

Table 4.5. Subsystem Capabilities for the Inlet

Capability (Parameter, Unit)	**Predicted Value**	**Actual Value (Low Deviation)**	**Actual Value (High Deviation)**
Maintain Hypersonic Speeds (Thrust, kN)	0.1 ± 0.03	0.08	0.06

Table 4.6. Subsystem Capabilities for the Nozzle

Capability (Parameter, Unit)	**Predicted Value**	**Actual Value (Low Deviation)**	**Actual Value (High Deviation)**
Maintain Hypersonic Speeds (Thrust, kN)	0.1 ± 0.03	0.11	0.12

Table 4.7. Subsystem Capabilities for the Subsonic Propulsion System

Capability (Parameter, Unit)	**Predicted Value**	**Actual Value (Low Deviation)**	**Actual Value (High Deviation)**
Thrust to Hypersonic Speeds (Thrust, kN)	90 ± 18	79	68

Table 4.8. Subsystem Capabilities for the Engine Integration

Capability (Parameter, Unit)	Predicted Value	Actual Value (Low Deviation)	Actual Value (High Deviation)
Maintain Hypersonic Speeds (Thrust, kN)	0.15 ± 0.03	0.13	0.11

Table 4.9. Subsystem Capabilities for the Flight Computer

Capability (Parameter, Unit)	Predicted Value	Actual Value (Low Deviation)	Actual Value (High Deviation)
Determine Vehicle Position and Motion (Performance Index, Unitless)	10 ± 2	9.1	8.2
Calculate Optimal Trajectory (Performance Index, Unitless)	10 ± 2	8.9	7.8

Table 4.10. Subsystem Capabilities for the Passive Thermal Protection

Capability (Parameter, Unit)	Predicted Value	Actual Value (Low Deviation)	Actual Value (High Deviation)
Limit Temperature Increase of Subsystems (Temperature Reduction, K)	375 ± 75	390	405
Protect Payload (Performance Index, Unitless)	10 ± 2	9.2	8.4

Table 4.11. Subsystem Capabilities for the Active Thermal Protection

Capability (Parameter, Unit)	Predicted Value	Actual Value (Low Deviation)	Actual Value (High Deviation)
Limit Temperature Increase of Subsystems (Temperature Reduction, K)	50 ± 15	40	30
Protect Payload (Performance Index, Unitless)	7 ± 2.1	5.8	4.6

Table 4.12. Subsystem Capabilities for the Power Subsystem

Capability (Parameter, Unit)	Predicted Value	Actual Value (Low Deviation)	Actual Value (High Deviation)
Provide Electrical Power (Power, W)	750 ± 225	720	690

Table 4.13. Subsystem Capabilities for the Sensors

Capability (Parameter, Unit)	Predicted Value	Actual Value (Low Deviation)	Actual Value (High Deviation)
Detect Target (Performance Index, Unitless)	10 ± 2	9.6	9.2
Determine Target Position (Performance Index, Unitless)	10 ± 2	8.6	7.2

Table 4.14. Subsystem Capabilities for the Communications Subsystem

Capability (Parameter, Unit)	Predicted Value	Actual Value (Low Deviation)	Actual Value (High Deviation)
Broadcast Vehicle Telemetry (Antenna Gain, dB)	12 ± 2.4	12.3	12.6
Receive Ground Commands (Antenna Gain, dB)	12 ± 2.4	12.3	12.6

Table 4.15. Subsystem Capabilities for the Aerodynamic Body

Capability (Parameter, Unit)	Predicted Value	Actual Value (Low Deviation)	Actual Value (High Deviation)
Aerodynamic Stability (Performance Index, Unitless)	4 ± 1.2	5.4	6.8
Vehicle Maneuverability (Performance Index, Unitless)	4 ± 1.2	4.5	5

Table 4.16. Subsystem Capabilities for the Internal Structures

Capability (Parameter, Unit)	Predicted Value	Actual Value (Low Deviation)	Actual Value (High Deviation)
Sustain Aerodynamic Loads (Yield Stress, MPa)	700 ± 210	680	660
Protect Payload (Performance Index, Unitless)	7 ± 2.1	6.7	6.4

Table 4.17. Subsystem Capabilities for the Vehicle Skin

Capability (Parameter, Unit)	Predicted Value	Actual Value (Low Deviation)	Actual Value (High Deviation)
Sustain Aerodynamic Loads (Yield Stress, MPa)	1000 ± 200	880	760
Protect Payload (Performance Index, Unitless)	10 ± 2	8.9	7.8

All 14 subsystem capabilities appear at least once in the generated portfolio, and the 14 selected subsystems contain a total of 24 instances of these subsystem capabilities. Of these 24, only one falls outside the uncertainty bounds for the low-deviation model, while 12 fall outside the bounds for the high-deviation model. Thus, the performance validation submetric $P = 0.958$ for the low-deviation demonstration case, while $P = 0.5$ for the high-deviation demonstration case. Even though the magnitude of the deviations is only twice as high for the high-deviation case, the performance validation submetric drops off dramatically as the subsystem capabilities fall outside acceptable uncertainty bounds. The cost validation submetric, meanwhile, is $C = 0.919$ for the low-deviation demonstration case, and $C = 0.845$ for the high-deviation demonstration case. While the difference between the two is closer to the difference between the low-deviation cost submetric and 1 than those of the performance submetric, the differences are not exactly the same, as the cost submetric uses the hyperbolic tangent of the error fraction rather than the error fraction directly. The validation metric is therefore $R = 0.939$ for the low-deviation case, and $R = 0.672$ for the high-deviation case.

4.4 SDDA Verification Results

The verification test for SDDA uses a top-level model of a hypersonic vehicle, where the dependencies between the subsystems are simplified enough that a systems engineer can easily calculate the beginning and end times for each subsystem by hand with relatively little effort. Fig. 4.2 plots the dependency network for the hypersonic vehicle model used in this verification test, where the arrows point from subsystems that impact other subsystems' development to the subsystems whose development is impacted. Each subsystem is dependent on no more than two other subsystems, so the beginning times for standard SDDA can be easily calculated while still being distinguishable from SDDA_{Max} (we'll consider both for this example).

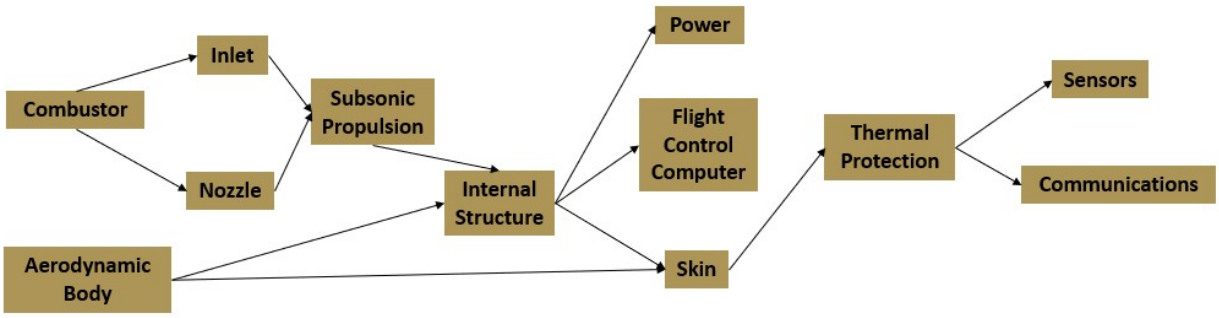

Figure 4.2. SDDA dependency network for a hypersonic vehicle, simplified for a verification test.

Because we've set the punctuality of all subsystems to 100, each subsystem should take the minimum amount of time to develop. For this example, all subsystems have a minimum development time of 10 weeks, so the end time should be 10 weeks after the most conservative beginning time. We'll use deterministic SDDA for this example, since the verification metric does not take uncertainty into account. For subsystems dependent on only one other subsystem, the beginning times will be the same for SDDA and SDDA_{Max}, and will be equal to the end time minus $(1 - SOD_{\text{ij}})$ times the minimum development time. However, for subsystems dependent on two or more subsystems, we'll need to consider multiple beginning times calculated using the aforementioned formula: For SDDA, the beginning development time will equal the average of the beginning times for each input subsystem; for SDDA_{Max},

the beginning time will equal the latest of all the beginning times calculated for each input subsystem. For example, the internal structures are dependent on the subsonic propulsion subsystem ($t_e = 12$ weeks, $SOD = 0.4$) and the aerodynamic body ($t_e = 10$ weeks, $SOD = 0.9$). The beginning time calculated from the subsonic propulsion system is thus $12-(1-0.4)*10 = 6$ weeks, while the beginning time calculated from the aerodynamic body is $10-(1-0.9)*10 = 9$ weeks, the average of which is 9.5 weeks. Table 4.18 and Table 4.19 show the beginning and end times output from SDDA and SDDA_{Max}, respectively, and the values predicted from the conceptual model. In both cases, the predicted values match the outputs exactly, so $V_S = 1$.

Table 4.18. Beginning and End Times for SDDA Verification (Standard SDDA)

Subsystem	**Predicted** t_b **(Weeks)**	**Output** t_b **(Weeks)**	**Predicted** t_e **(Weeks)**	**Output** t_e **(Weeks)**
Combustor	0	0	10	10
Inlet	1	1	11	11
Nozzle	1	1	11	11
Subsonic Propulsion	2	2	12	12
Aerodynamic Body	0	0	10	10
Internal Structures	7.5	7.5	19	19
Skin	13.5	13.5	28	28
Thermal Protection	27	27	37	37
Flight Control Computer	10	10	20	20
Sensors	28	28	38	38
Communications	31	31	41	41
Power	13	13	23	23

Table 4.19. Beginning and End Times for SDDA Verification ($SDDA_{Max}$)

Subsystem	**Predicted** t_b **(Weeks)**	**Output** t_b **(Weeks)**	**Predicted** t_e **(Weeks)**	**Output** t_e **(Weeks)**
Combustor	0	0	10	10
Inlet	1	1	11	11
Nozzle	1	1	11	11
Subsonic Propulsion	2	2	12	12
Aerodynamic Body	0	0	10	10
Internal Structures	9	9	19	19
Skin	18	18	28	28
Thermal Protection	27	27	37	37
Flight Control Computer	10	10	20	20
Sensors	28	28	38	38
Communications	31	31	41	41
Power	13	13	23	23

4.5 SDDA Validation Results

The hypersonic vehicle model used for the validation test contains the same set of 12 subsystems as the model used in the verification test, but with more connections included between subsystems. The hypersonic combustor, for instance, impacts the development of seven other subsystems, while the power subsystem has its development impacted by five others. Additionally, the minimum and maximum development times are no longer standardized across subsystems — on the short end, the development time of the sensors ranges between 4 and 8 weeks, while that of the subsonic propulsion ranges between 20 and 50 weeks — and the punctuality of each subsystem ranges between 50 and 90. Fig. 4.3 plots the dependency network for the hypersonic vehicle model used for the validation test.

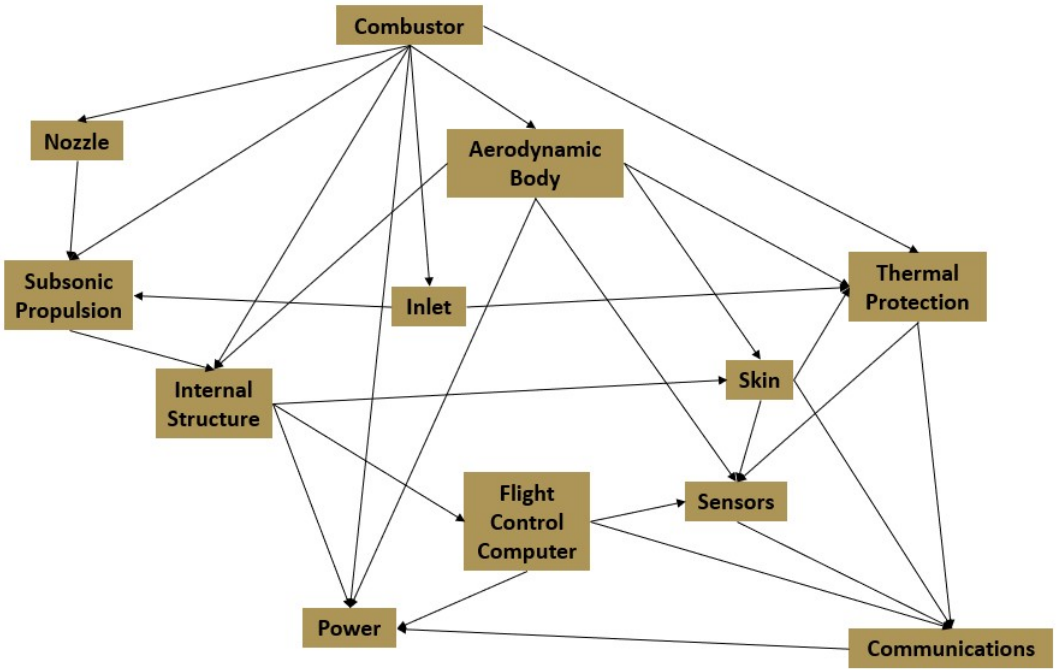

Figure 4.3. SDDA dependency network for a hypersonic vehicle.

Fig. 4.4 plots the Gantt chart for a run of stochastic SDDA on the validation example. Here, the difference between the beginning and end times for standard SDDA and SDDA_{Max} is stark, with the vehicle finishing its predicted development schedule over 30 weeks later for the latter when compared to the former. Additionally, for both forms of SDDA the uncertainty propagates as development progresses, with the distributions of each beginning and end time growing visibly wider.

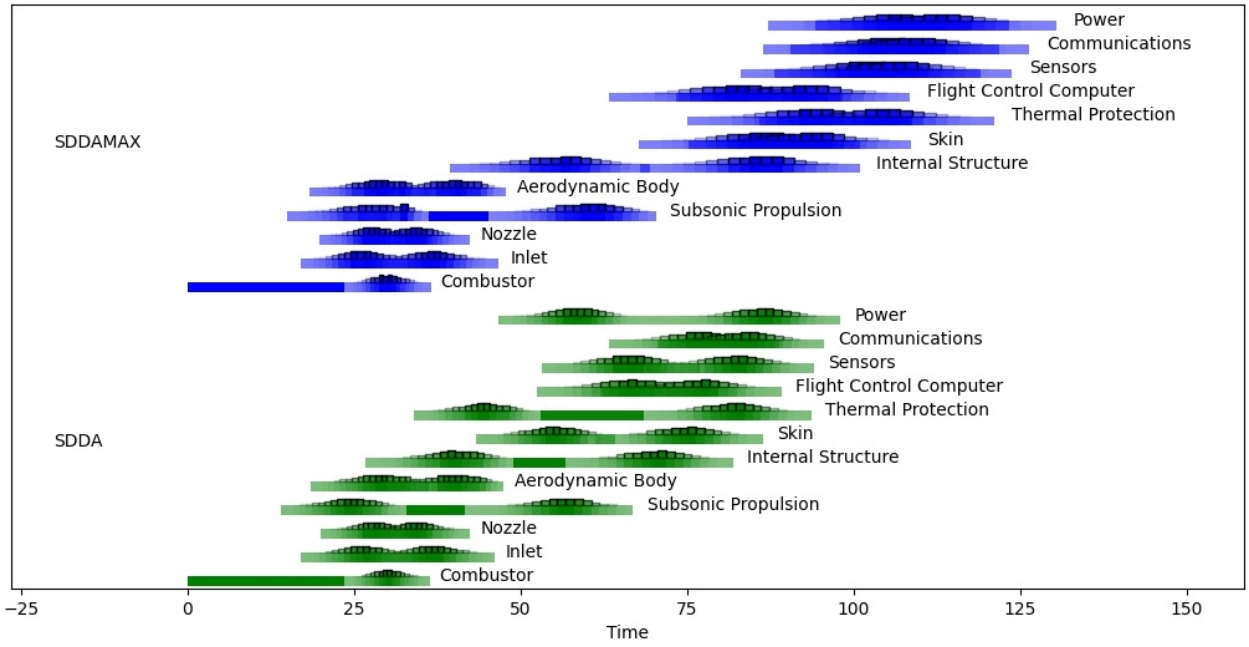

Figure 4.4. Gantt chart for a hypersonic vehicle development timeline, with standard SDDA in green and SDDA_{Max} in blue. The vertical bars above each horizontal bar represent the probability distributions for the beginning and end times of each subsystem's development. Time is measured in weeks.

In this example, we compare the outputs of standard SDDA and SDDA_{Max} with a hypothetical hypersonic vehicle that has already finished development, and thus has a known development timeline. The work presented in this thesis will use the same known development schedule for standard SDDA and SDDA_{Max}, as this allows comparison between the two levels of conservatism. Our validation metric S considers the number of subsystem beginning and end development times that fall outside a $(1-\alpha)\times 100\%$ confidence interval, and for this example we set $\alpha = 0.05$. Table 4.20 and Table 4.21 show the confidence intervals for the beginning and end times output from SDDA and SDDA_{Max}, respectively, alongside the actual beginning and end times for the hypothetical hypersonic vehicle. Of the 12 beginning and 12 end times, two beginning times fall outside the confidence intervals predicted by standard SDDA, while seven beginning and seven end times fall outside those predicted by SDDA_{Max}. Thus, $S = 0.917$ for standard SDDA, and $S = 0.417$ for SDDA_{Max}, implying that a low-conservatism implementation of SDDA more accurately predicts the development timeline of this hypothetical hypersonic vehicle than a high-conservatism implementation.

Table 4.20. Beginning and End Times for SDDA Validation (Standard SDDA)

Subsystem	t_b **95% CI (Weeks)**	**Actual** t_b **(Weeks)**	t_e **95% CI (Weeks)**	**Actual** t_e **(Weeks)**
Combustor	0.0 - 0.0	0	25.8 - 34.3	30
Inlet	20.4 - 32.6	22	30.8 - 43.2	35
Nozzle	22.8 - 33.3	23	29.0 - 39.6	36
Subsonic Propulsion	17.7 - 30.5	32	48.8 - 63.7	60
Aerodynamic Body	22.2 - 34.3	23	33.1 - 45.4	44
Internal Structures	32.6 - 46.4	35	62.2 - 77.1	65
Skin	47.8 - 60.9	55	66.7 - 81.6	80
Thermal Protection	37.9 - 50.1	49	73.9 - 88.8	82
Flight Control Computer	58.1 - 74.1	60	68.3 - 84.3	69
Sensors	58.6 - 72.7	62	74.3 - 89.2	76
Communications	69.0 - 84.1	71	75.9 - 90.8	77
Power	51.3 - 64.3	68	78.3 - 93.2	90

Table 4.21. Beginning and End Times for SDDA Validation (SDDA_{Max})

Subsystem	t_b **95% CI (Weeks)**	**Actual** t_b **(Weeks)**	t_e **95% CI (Weeks)**	**Actual** t_e **(Weeks)**
Combustor	0.0 - 0.0	0	25.7 - 34.2	30
Inlet	20.2 - 32.5	22	30.7 - 43.0	35
Nozzle	22.6 - 33.2	23	28.9 - 39.5	36
Subsonic Propulsion	19.6 - 34.2	32	50.8 - 67.5	60
Aerodynamic Body	21.9 - 34.2	23	33.0 - 45.3	44
Internal Structures	45.3 - 65.2	35	75.1 - 95.6	65
Skin	75.0 - 95.5	55	82.4 - 103.2	80
Thermal Protection	82.3 - 103.1	49	93.2 - 114.2	82
Flight Control Computer	71.3 - 92.5	60	81.4 - 102.7	69
Sensors	90.6 - 111.6	62	95.4 - 116.4	76
Communications	93.5 - 114.6	71	97.9 - 118.9	77
Power	94.6 - 115.7	68	101.7 - 122.8	90

5. CONCLUSIONS

5.1 Summary

The motivation for this thesis is the necessity of ensuring that the tools systems engineers use for design analysis carry out their intended functions with a high degree of accuracy. The AWB is one such suite of tools, designed to address complex interdependencies between constituent systems in an SoS or a highly-integrated system in order to more effectively generate system architectures and characterize developmental and operational risks in an MBSE environment. One class of engineered system with a high degree of interdependence between subsystems is that of hypersonic flight vehicles, as the aerodynamic characteristics of hypersonic speeds and the necessity for multiple types of propulsion systems creates numerous constraints on the vehicle's design and results in a narrow performance envelope. As such, the AWB and its constituent tools — including RPO and SDDA — could be useful in generating conceptual designs for hypersonic vehicles, though design engineers would need a way to verify and validate the AWB tools for this class of problem. To this end, this thesis addressed the following research question: *What quantitative measures can we use to evaluate the effectiveness of Analytic Workbench tools in generating and evaluating hypersonic vehicle architectures?* This chapter summarizes the findings pertaining to this question as applied to RPO and SDDA.

This thesis presents four effectiveness metrics that can be applied to the AWB tools, one each for RPO verification, RPO validation, SDDA verification, and SDDA validation. These metrics were partially inspired by verification and validation metrics for computational fluid dynamics software, in particular the work of Oberkampf and Trucano [14]. Each effectiveness metric takes into account the quantifiable outputs of the tool to which it applies, and how it compares to the results of either a conceptual analysis in the case of verification, or a physical system in the case of validation. For RPO, these outputs include the system portfolios generated at different cost points, the decisions that RPO made in generating them, and the subsystem cost and capabilities. The SDDA metrics account for each subsystems beginning time of development and end time of development.

We selected the four effectiveness metrics from a list of five candidate metrics, applying each candidate metric to the effectiveness metrics whose requirements it satisfied:

- Error Measurements: Did not satisfy the requirements for any of the four effectiveness metrics
- Coefficient of Determination: Did not satisfy the requirements for any of the four effectiveness metrics
- Weighted Sum of Hyperbolic Tangents: Satisfied the requirements for RPO validation (cost submetric) and SDDA verification
- Ratio of Sums of Boolean Variables: Satisfied the requirements for RPO verification and RPO validation (performance submetric)
- Weighted Sum of Significance Tests: Satisfied the requirements for SDDA validation

To demonstrate the application of these metrics to a physical system, this thesis created a demonstration case for each one in the form of a hypothetical hypersonic vehicle, with a system architecture based on the GHV [22] — a class of hypersonic vehicle conceptual designs developed by the AFRL — and system capabilities derived from the NASP [23] — a proposed hypersonic vehicle in the 1980s and 1990s designed as a single-stage-to-orbit vehicle. The corresponding RPO and SDDA models for the verification test were simple enough that the outputs generated from a conceptual analysis could easily be calculated by hand, such that the verification metrics were always equal to 1. The RPO and SDDA models for the validation test were constructed such that the outputs deviated somewhat from the hypothetical physical systems, so that the resulting loss of accuracy could be clearly seen in the validation metrics. For SDDA in particular, the hypothetical physical system aligned more with the level of conservatism predicted by standard SDDA than SDDA_{Max}, so the validation metric was higher for the former than the latter. Since the verification metrics for RPO and SDDA — the implementations of which had been extensively developed and tested — both equaled 1, and the validation metrics clearly decreased as the deviation in the outputs from the demonstration case vehicles increased, we can conclude that the selected metrics were appropriate for the verification and validation of these AWB tools.

5.2 Future Work

We can explore more ways to apply verification and validation effectiveness metrics to RPO and SDDA:

- Other RPO Formulations: The current effectiveness metrics assume that RPO uses the Bertsimas-Sim formulation of risk [16][17]. We can reevaluate our requirements for RPO effectiveness metrics to generate measures that apply to the robust mean-variance and CVaR formulations of RPO.

- Cost and Performance Weighting: In our requirements in Chapter 2, we specify that the RPO validation metric must weight cost and performance equally, as we can't assume that RPO places a higher importance on one than the other. However, subsequent explorations of the RPO validation metrics could test this assumption by placing different weights on cost and performance, and examining how different weightings affect the validation process.

- Beginning and End Time Weighting: Similarly, the requirements for SDDA verification and validation specify that their respective effectiveness metrics must place equal weight on the beginning and end development times. However, since the end development time of each subsystem depends, by definition, on the beginning development time, and SDDA_{Max} differs from standard SDDA exclusively in the formulation of the beginning time, it's possible that the effectiveness metrics would be more useful with a higher importance placed on beginning time. Subsequent explorations could test this assumption by examining how different weightings affect the verification and validation process.

- Significance Level Selection: In Chapter 2, we specified that the demonstration cases would assume a significance level of 0.05, which is commonly used in statistics. Subsequent explorations of the SDDA validation metric could apply it using other commonly used values of α, to see how it affects the validation of SDDA given deviations of various magnitudes from the demonstration case development schedule.

- SDDA Conservatism: The effectiveness metrics for SDDA make no distinction between standard SDDA and SDDA_{Max}, and does not consider any other levels of conservatism besides these two. Further explorations of these metrics could include conservatism as a variable, in addition to beginning and end development times, and we could examine ways to quantify conservatism to possibly create a "middle ground" between SDDA and SDDA_{Max}.

- Expansion to Other Problems: In Chapter 2, we developed the requirements for the verification and validation metrics within the context of a hypersonic vehicle design problem, where the portfolios and networks consisted of top-level subsystems. If we were to apply these effectiveness metrics to a system-of-systems problem, or a design involving both top-level and lower level subsystems, we may need to reevaluate the requirements to ensure that they still apply. If not, we would need to develop a different set of effectiveness metrics for that kind of problem.

Additionally, we would need to generate a larger suite of verification test cases to ensure that the verification metrics applied to all of them, and subsequent applications of the validation metrics should use an actual physical system, rather than the hypothetical system used in Chapter 4. Nevertheless, these effectiveness metrics — and the example problems to which we applied them in this thesis — provide a good starting point for further examination.

The next step in developing effectiveness metrics for the analysis tools within the AWB would be to develop verification and validation metrics for SODA, which — as explained in Chapter 1 — was beyond the scope of this thesis. However, generating metrics for SODA presents a problem not present in the development of effectiveness metrics for RPO and SDDA: The outputs of RPO (generated portfolios, subsystem capabilities, and subsystem costs) and SDDA (beginning and end development time) have inherent physical meaning, and correspond to quantities that are unambiguously measurable in a physical system. However, the outputs of SODA — subsystem operability and total system operability — don't correspond with physical quantities, and neither does the subsystem self-effectiveness [9]. For example, what does it mean for a hypersonic combustor to have an operability of 50? Does that mean that the combustor is operating as intended, just not at its optimal condi-

tions, or that it's experiencing one or more mechanical failures? As a consequence, one could potentially change the definition of operability depending on the problem in order to make the SODA analysis appear more effective than it actually is, thus weakening the applicability of the validation metrics. As such, any exploration of effectiveness metrics for SODA should carefully define operability for each class of problem *a priori*, to prevent the possibility of "gaming" the outcome.

REFERENCES

[1] P. Hagseth, "Hypersonic Vehicle Design Short Course: Air-Breathing Hypersonic Propulsion," *American Institute of Aeronautics and Astronautics*, 2020.

[2] K. Bowcutt, "Hypersonic Vehicle Design Short Course: Hypersonic Vehicle Design Challenges," *American Institute of Aeronautics and Astronautics*, 2020.

[3] L. Hart, "Model-Based Systems Engineering: An Emerging Approach for Modern Systems," *Delaware Valley INCOSE Chapter Meeting*, 2015. [Online]. Available: https://www.incose.org/docs/default-source/delaware-valley/mbse-overview-incose-30-july-2015.pdf.

[4] J. Dahmann, "System of Systems Pain Points," *INCOSE International Symposium*, vol. 24, no. 1, pp. 108–121, 2014. DOI: 10.1002/j.2334-5837.2014.tb03138.x.

[5] N. Davendralingam, D. DeLaurentis, Z. Fang, C. Guariniello, S. Y. Han, K. Marais, A. Mour, and P. Uday, "An analytic workbench perspective to evolution of system of systems architectures," *Procedia Computer Science*, vol. 28, pp. 702–710, 2014, 2014 Conference on Systems Engineering Research, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2014.03.084.

[6] M. W. Maier, "Architecting Principles for Systems-of-Systems," *Systems Engineering*, vol. 1, no. 4, pp. 267–284, 1998. DOI: 10.1002/(SICI)1520-6858(1998)1:4<267::AID-SYS3>3.0.CO;2-D.

[7] N. Daventralingam, M. Mane, and D. A. DeLaurentis, "Capability and Development Risk Management in System of Systems Architectures: A Portfolio Approach to Decision Making," *Defense Technical Information Center*, 2012. [Online]. Available: https://apps.dtic.mil/sti/citations/ADA563286.

[8] W. J. O'Neill, D. A. DeLaurentis, and N. Davendralingam, "The price of robustness," *Proceedings of the 69th International Astronautical Congress*, 2018. [Online]. Available: https://iafastro.directory/iac/paper/id/45149/abstract-pdf/IAC-18,D1,4B,10,x45149.brief.pdf?2018-03-29.11:44:22.

[9] C. Guariniello and D. A. DeLaurentis, "Supporting design via the System Operational Dependency Analysis methodology," *Research in Engineering Design*, vol. 28, Jan. 2017. DOI: 10.1007/s00163-016-0229-0.

[10] C. Guariniello and D. A. DeLaurentis, "Dependency Analysis of System-of-Systems Operational and Development Networks," *Procedia Computer Science*, vol. 16, pp. 265–274, 2013, 2013 Conference on Systems Engineering Research, ISSN: 1877-0509. DOI: 10.1016/j.procs.2013.01.028.

[11] E. V. Sitchin, B. Smith, W. W. Stahlschmidt, A. K. Raz, and D. A. DeLaurentis, "Modeling Hypersonic Vehicle Interdependencies at the Subsystem Level," in *AIAA Aviation 2021 Forum*. 2021. DOI: 10.2514/6.2021-2456. eprint: https://arc.aiaa.org/doi/pdf/10.2514/6.2021-2456. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/6.2021-2456.

[12] L. J. Gullo and J. Dixon, *Design for Safety*, eng, ser. Wiley Series in Quality & Reliability Engineering. Chichester, West Sussex: Wiley, 2018, ISBN: 978-1-118-97429-2.

[13] G. Shea, *NASA Systems Engineering Handbook Revision 2*, Text, 2017. [Online]. Available: https://www.nasa.gov/connect/ebooks/nasa-systems-engineering-handbook.

[14] W. L. Oberkampf and T. G. Trucano, "Verification and Validation in Computational Fluid Dynamics," *Progress in Aerospace Sciences*, vol. 38, no. 3, pp. 209–272, 2002, ISSN: 0376-0421. DOI: 10.1016/S0376-0421(02)00005-2. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0376042102000052.

[15] D. D. Walden, G. J. Roedler, K. Forsberg, R. D. Hamelin, and T. M. Shortell, Eds., *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, 4th ed. Hoboken, NJ: Wiley, 2015, ISBN: 978-1-118-99940-0.

[16] D. Bertsimas and M. Sim, "The price of robustness," *Operations research*, vol. 52, no. 1, pp. 35–53, 2004. DOI: 10.1287/opre.1030.0065.

[17] N. Davendralingam and D. DeLaurentis, "A robust optimization framework to architecting system of systems," *Procedia Computer Science*, vol. 16, pp. 255–264, 2013, 2013 Conference on Systems Engineering Research, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2013.01.027.

[18] N. Davendralingam and D. A. DeLaurentis, "An Analytic Portfolio Approach to System of Systems Evolutions," *Procedia Computer Science*, vol. 28, pp. 711–719, 2014, 2014 Conference on Systems Engineering Research, ISSN: 1877-0509. DOI: 10.1016/j.procs.2014.03.085.

[19] "Guide: Guide for the Verification and Validation of Computational Fluid Dynamics Simulations (AIAA G-077-1998(2002))," in *Guide: Guide for the Verification and Validation of Computational Fluid Dynamics Simulations (AIAA G-077-1998(2002))*. 1998. DOI: 10.2514/4.472855.001. eprint: https://arc.aiaa.org/doi/pdf/10.2514/4.472855.001. [Online]. Available: https://arc.aiaa.org/doi/abs/10.2514/4.472855.001.

[20] P. Hagseth, "Hypersonic Vehicle Design Short Course: Historical Review," *American Institute of Aeronautics and Astronautics*, 2020.

[21] D. Wilson, "Hypersonic Vehicle Design Short Course: Review of Current International Programs," *American Institute of Aeronautics and Astronautics*, 2020.

[22] B. Ruttle, J. Stork, and G. Liston, *Generic Hypersonic Vehicles for Conceptual Design Analyses*, Text, 2012.

[23] B. W. Augenstein and E. D. Harris, *The National Aerospace Plane (NASP): Development Issues for the Follow-on Vehicle*, Text, 1993. [Online]. Available: https://apps.dtic.mil/sti/pdfs/ADA278417.pdf.