AEROSPACE REPORT NO.
TOR-2021-00133

A Class Agnostic Mission Assurance Approach

January 15, 2021

Barbara M. Braun[1], Lisa A. Berenberg[2], Sabrina L. Herrin[1], Riaz S. Musani[1], and Douglas A. Harris[1]
[1]Space Innovation Directorate, Advanced Development and Planning Division
[2]NNSA Nuclear Nonproliferation Programs, NNSA Programs Directorate

Prepared for:

Space and Missile Systems Center
United States Space Force
483 N. Aviation Blvd.
El Segundo, CA 90245-2808

Contract No. FA8802-19-C-0001

Authorized by: Defense Systems Group

Distribution Statement A: Approved for public release; distribution unlimited.

Acknowledgments

The authors would like to thank all the individuals who have contributed to mission management support to the Space Test Program (STP), Space Rapid Capabilities Office (SpRCO), and other space missions supported by the Space Innovations Directorate (SID) over the years and have helped develop the mission management process described in this document. Special thanks go to Peter Chang and Andrew Read for developing the early concepts and heuristics of this approach and to Mark Jelonek, Gayla Walden, and Kara O'Donnell for their review and feedback for improving the document.

Abstract

The space enterprise is rapidly changing - presenting challenges to the Department of Defense, civil and commercial agencies, and ultimately to the Aerospace Corporation as the premiere provider of space enterprise mission assurance. A rapidly growing need for smaller more affordable space missions is testing how Aerospace executes its mission assurance function. As a corporation, we are facing direction to execute many of our missions in two- to three-year time frames, with an increased acceptance of risk and "good enough" performance. Commensurately, there is an increased demand for a mission assurance approach that can deliver best effort within constrained resources and shortened schedules, gracefully accept the resulting risk, and still achieve enough system performance to meet mission objectives or success criteria. The Aerospace Space Innovation Directorate (SID) has, for two decades, practiced a unique approach to mission assurance that can respond to this need.

SID has an excellent track record of success with a framework that has evolved over time and is historically anchored in the traditional Aerospace methodologies. Budget and schedule constraints have traditionally challenged the SID team to, by necessity, adopt a more tailored approach. These practices have continued to evolve with our customers' pursuit of smaller, more affordable, risk tolerant missions. Despite programmatic limitations, this approach has delivered a greater than 95% success rate for these sorts of missions while using 2-3 STE of Aerospace support per mission per year. SID has recently begun to document this capability, calling it a "Class Agnostic Mission Assurance Approach for Constraints-Driven Missions."

Contents

Figures

1. Introduction

1.1 Background

SID has supported more than two dozen missions over the last 20 years servicing the USAF Space Test Program (STP), the Air Force Research Laboratory (AFRL), the Space Rapid Capabilities Office (SpRCO; formerly known as the Operationally Responsive Space office), and other national and civil agencies. In supporting these programs, SID has developed a unique brand of mission assurance for missions ranging from CubeSats to multi-manifest missions flying on Medium and Heavy launch vehicles. Figure 1-1 is a sampling of SID supported missions showing the wide variety of partners over a dozen different launch vehicles at multiple launch sites. This has included mission assurance for payload and bus developments as well as their integration to launch systems. Since 2000, SID has delivered greater than 95% mission success for these smaller risk tolerant missions within highly constrained budgets and schedule, using 2-3 Aerospace STE per year per mission (not including the STE required for Aerospace launch vehicle mission assurance for the NSSL launch vehicles used).

Year	Mission	Launch Vehicle	Launch Site	Type
2000	JAWSAT	Minotaur I	Vandenburg	Launch Integ
	MTI	Taurus	Vandenburg	Launch Integ
	TSX-5	Pegasus	Vandenburg	Launch Integ
	MightySat II.1	Minotaur I	Vandenburg	Launch Integ
2001	Kodiac Star	Athena I	Kodiac	Launch Integ
2002				
2003	Coriolis	Titan II	Vandenburg	Launch Integ
2004	UNSat-2	Delta IV Heavy	Cape Canaveral	Launch Integ
2005	XSS-11	Minotaur I	Vandenburg	Launch Integ
	STP-R1	Minotaur I	Vandenburg	Launch Integ
2006	COSMIC-1	Mintoraur I	Vandenburg	Launch Integ
	STP-H2	Shuttle	Kennedy	Launch Integ
2007	STP-1	Atlas V	Cape Canaveral	Launch Integ
	STPSat-1	"	"	Bus Dev & Integ
2008	C/NOFS	Pegasus	Kwajlien	Launch Integ
2009	HICO / RAIDS	HTV	Japan	Bus & Launch Integ
2010	STPSat-2	Minotaur IV	Kodiac	Bus & Launch Integ
	STP-S26	"	"	Bus & Launch Integ
2011	ORS-1	Minotaur I	Wallops	Bus & Launch Integ
2012				
2013	ORS-3	Minotaur I	Wallops	Bus & Launch Integ
	STPSat-3	"	"	Bus & Launch Integ
	STP-H4	HTV	Japan	Launch Integ
2014	Angels	Delta IV	Cape Canaveral	Bus & Launch Integ
2015	DSCOVR	Falcon 9	Cape Canaveral	Launch Integ
2016				
2017	ORS-5	Minotaur IV	Cape Canaveral	Bus & Launch Integ
	STP-H5	Falcon 9	Kennedy	Launch Integ
	ASETS-II	X-37	Cape Canaveral	Bus & Launch Integ
2018	EAGLE	Atlas V	Cape Canaveral	Bus & Launch Integ
	STPSat-5	Falcon 9	Vandenburg	Bus & Launch Integ
2019	STP-2 (24 SVs)	Falcon Heavy	Cape Canaveral	Bus & Launch Integ
	STP-H6	Falcon 9 / ISS	Cape Canaveral	Launch Integ
2020	STPSat-4	Antares / ISS	Wallops	Bus & Launch Integ

Figure 1-1. Two decades of SID mission support.

During this time, SID has evolved and refined its mission assurance approach. There is no "secret sauce," but there is a thoughtful approach requiring something of a shift in mindset. Although SID's approach is ideal for risk-tolerant (traditionally Risk Class C or D) missions constrained by budget, schedule and other resources, there is value in employing this mindset for Class A and B missions as well. Essentially, the SID approach addresses mission assurance from a system engineering point of view and an agile

mindset that begins with the question “what is it that we are actually trying to do?” The answer to this question is the cornerstone of the whole process upon which subsequent analyses and decisions are made. The approach also calls for highly seasoned systems engineering skills drawing upon lessons learned which is why it is augmented with an active lesson learned program and an “apprenticeship” program to build good, efficient, program office systems engineers, thus deepening the bench essential for continuing and expanding capability.

1.2 Organization of TOR

This TOR begins by exploring the Agile mindset and the concept of requirements versus constraints-driven mission. It then describes how the Class Agnostic Mission Assurance approach borrows Agile concepts for employing mission assurance in uncertain, constraints-driven development environments. Finally, it provides a full description of the Agile Class Agnostic Mission Assurance approach for extension to any mission assurance application across any risk class.

2. Key Concepts

2.1 Introduction

The traditional Class A, B, C, and D mission risk classification scheme can be a useful tool for communicating the risk posture of a mission to stakeholders and bidders, but it does suffer shortcomings. All too often, these classifications are used as a kind of shorthand for the fiscal realities of the mission rather than a true risk posture. They tend to be monolithic, glossing over the fact that a single satellite mission can have a mixture of risk levels – one subsystem can require Class A attention, while for another more robust or less critical subsystem, Class D might be acceptable. Furthermore, once a risk class designation is established, there is typically little to no linkage of that risk posture with the specifics of program execution – there is little guidance given on which risks to mitigate or to accept given the program's resource constraints. The traditional class designation also ignores whether requirements or the constraints drive the mission, and typically isn't flexible to the changing priorities encountered during program execution.

The Space Innovation Directorate has spent a lot of time developing and refining its class agnostic mission assurance model for those missions that do not clearly fit into any one traditional Class A to D construct. In refining this concept, there was a recent recognition of similarities between the SID approach to mission assurance and the Agile Software Development movement. This section briefly reviews Agile concepts drawing analogies between the Agile mindset and that of class agnostic mission assurance. It also attempts to highlight Agile practices that are relevant to mission assurance.

2.2 Agile Mindset and Manifesto

Agile as we know it today was formalized in the early 2000s as a set of principles and mindsets for better software development. On its website, the Agile Alliance describes Agile as "the ability to create and respond to change. It is a way of dealing with, and ultimately succeeding in, an uncertain and turbulent environment." (Agile Alliance, n.d.) Agile software development is an umbrella term for a set of frameworks and practices based on the values and principles expressed in the Manifesto for Agile Software Development and the 12 Principles behind it. Figure 2-1 shows the Agile Manifesto as described by the Agile Alliance website (Agile Alliance, n.d.).

The Agile Manifesto

We are uncovering better ways of developing software by doing it and helping others do it. Through this work we have come to value:

Individuals and interactions over processes and tools
Working software over comprehensive documentation
Customer collaboration over contract negotiation
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Figure 2-1. The Agile Manifesto.

The 12 implementing principles behind the Agile Manifesto can also be found on the Agile Alliance website, and include such statements as:

- Businesspeople and developers must work together daily throughout the project.
- The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.
- Continuous attention to technical excellence and good design enhances agility.
- Simplicity–the art of maximizing the amount of work not done–is essential.
- At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behavior accordingly.

Agile is "really about thinking through how you can understand what's going on in the environment that you're in today, identify what uncertainty you're facing, and figure out how you can adapt to that as you go along." All of these principles can be applied to Mission Assurance. As it says on the Agile Alliance website, "when you think of Agile as a mindset, that mindset can be applied to other activities." (Agile Alliance, n.d.)

2.3 Agile Mindset for Mission Assurance

Class Agnostic Mission Assurance requires much of the same mindset as Agile. Our version of the Agile Manifesto for Mission Assurance is shown in Figure 2-2.

An Agile Mission Assurance Manifesto

We are committed to understanding how best to improve the chances of mission success for all sorts of missions. Through this we have come to value:

Individuals and interactions over processes and tools
Mission success over comprehensive documentation
Customer collaboration over prescriptive requirements and checklists
Responding to change over following a plan

That is, while there is value in the items on the right, we value the items on the left more.

Figure 2-2. Agile Mission Assurance Manifesto.

Scott Ambler of the Agile Alliance further explains the Agile Manifesto for Agile Software developments as listed below. (Ambler, n.d.) The authors of this paper include analogies (also listed below) for describing a similar manifesto for agile mission assurance.

Agile Software: Tools and processes are important, but it is more important to have competent people working together effectively.

Agile Mission Assurance: Mission assurance tools and processes are important, but it is more important to have competent people working together effectively.

Agile Software: Good documentation is useful in helping people to understand how the software is built and how to use it, but the main point of development is to create software, not documentation.

Agile Mission Assurance: Good documentation is important in helping people understand risk and risk mitigation, but the main point of mission assurance is to improve the chance of mission success, not to document risks.

Agile Software: A contract is important but is no substitute for working closely with customers to discover what they need.

Agile Mission Assurance: Requirements lists and CDRLs are important but are no substitute for working closely with all partners to discover what the mission needs to do, and how best to help it succeed.

Agile Software: A project plan is important, but it must not be too rigid to accommodate changes in technology or the environment, stakeholders' priorities, and people's understanding of the problem and its solution.

Agile Mission Assurance: A mission assurance plan is important, but it must not be too rigid to accommodate changes in priorities, resources, risk, and people's understanding of the mission's objectives.

2.4 Requirements-Driven and Constraints-Driven Missions

SID's agile class agnostic approach also recognizes that missions are typically either requirements-driven, or constraints-driven. When push comes to shove, missions will either let the requirements drive cost and schedule (adding money and time to meet requirements), or missions will let cost and schedule drive requirements (reducing scope to meet a fixed budget or schedule). The distinction between requirements-driven and constraints-driven missions was first articulated as part of a paper at the 2018 Conference on Small Satellites (Jasper et al., 2018), and later refined in an Institute of Electrical and Electronics Engineers (IEEE) paper (Jasper et al., 2020).

Requirements-driven missions are focused on mission system performance with less emphasis on how that drives budget and schedule.

A Requirements-Driven Mission is:

> *A mission where mission objectives / requirements drive the schedule and budget and where objectives are typically prioritized over schedule and budget* (Jasper et al., 2020, p. 2)

This is not to say that requirements-driven missions do not need to apply due diligence with respect to cost and schedule, but cost and schedule are typically less constrained (e.g., externally constrained launch dates or budget figures) and achieving mission requirements is the focus. Requirements-driven missions have more flexibility for delaying a launch or adding funding to ensure all mission requirements are met. These missions also require a full complement of mission assurance for ensuring the highest probability of successful system performance.

In contrast, constraints-driven missions are highly focused on achieving, within externally constrained budgets and schedule, good enough performance that still satisfies mission objectives.

A Constraints-Driven Mission is:

> *A mission where schedule and budget are equal to, or prioritized above, the objectives/scope. The objectives/scope are traded with, or bounded by, schedule and budget and all three may evolve as the system is defined, designed, tested and operated.* (Jasper et al., 2020, p. 2)

Constraints-driven missions have little to no flexibility for delaying a launch or adding funding to resolve issues that arise. Such missions risk cancellation if they exceed their budget, or risk missing their launch if they exceed schedule. Cost and schedule are the most common constraints, but others may exist as well. For example, there are significant volume and design constraints associated with the CubeSat form factor. "Common bus" or standardized-architecture implementations represent another type of constraint that will likely increase in importance as more programs embrace a production mindset. Standardization requires compromise, and if stakeholders wish to prioritize adherence to a standard interface, they must allow developers the freedom to trade functionality or performance requirements if needed to fit that standard.

For constraints-driven missions, it may be necessary to reduce technical performance and/or accept increased risk to meet program constraints. Such missions must also acknowledge that something less than full mission success (even failure) is possible. Therefore, constraints-driven missions must continually adjust their scope of activity to fit schedule, budget, and resource constraints when addressing emerging risks. In other words, a constraints-driven mission will stay "within the box," while a requirements-driven mission has the freedom to "build its own box."

It is essential for the mission team and supported stakeholders to decide if a mission is requirements-driven or constraints-driven, since this will drive programmatic decisions and trades throughout the mission development. Figure 2-3 shows a spectrum of constraints vs. requirements driven missions with some examples that may help guide the decision.

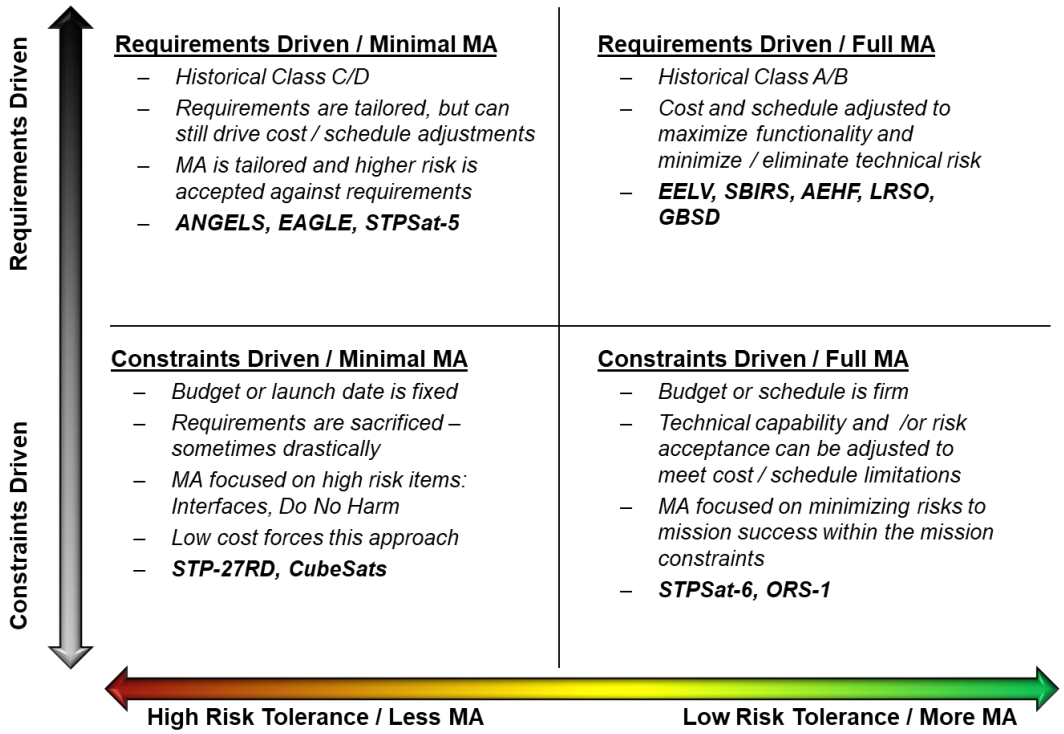


Figure 2-3. Spectrum of Mission Assurance for Constraints and Requirements-driven missions.

2.5 Key Mission Attributes and Concepts

The class agnostic framework is rooted in Agile concepts, and these concepts are applicable to any mission of any size. However, the class agnostic methodology is most applicable to missions with some or all the following attributes.

- Missions that are partly or mostly constraints-driven. The class agnostic heuristic can apply to both constraints-driven and requirements-driven missions, but it is most effective where missions are willing to trade requirements and risk to remain within cost, schedule, or other constraints.

- Missions with small program offices. One of the major tenets of Agile is that it requires small, high-performing teams working closely together. In large program offices with many organizational layers, the sheer size and complexity of the program and its staffing profile makes close communication and coordination difficult. Such missions typically also have the budget to conduct a full independent mission assurance effort, and class agnostic mission assurance may be less appropriate.

- Missions for which less than 100% mission success is an option. A critical launch is a good example of a mission with very few options for less than 100% mission success – it either makes it to orbit, or it doesn't. The heuristic can be applied to missions for which full mission success is the only metric that is "good enough," but in that case it is very similar to traditional Class A mission assurance.

Additionally, the class agnostic mission assurance approach relies on several key concepts.

- Mission assurance is anything that improves the chances of mission success. In constraints-driven missions, risk is often accepted to remain within program constraints, and a guarantee of mission

success is rarely possible. In this context, class agnostic mission assurance seeks to maximize, rather than guarantee, the chances of mission success within the available cost, schedule, and resource constraints.

- Mission assurance is not the purview of any single organization. Everyone involved in the mission – the government program office along with its contractor, SETA, and Aerospace support – makes up the mission assurance function. In most applications of the class agnostic heuristic, mission assurance is the collective effort of the entire team, not one of independent oversight. Mission assurance need not be independent to be objective.

- Efficient mission assurance requires mentorship. When mission assurance efforts are constrained by limited resources, engineers need to have good instincts, so they can spot the issues faster and exercise good judgement in prioritizing activities. Pairing novice engineers with more experienced engineers (like pairing an apprentice with a master) helps generate new seasoned engineers with the right instincts.

- Class agnostic mission assurance tailors "up." Traditional mission assurance approaches start with a "Class A" requirements-driven approach and tailor back. This is less appropriate for highly constrained missions, which start with the most basic "Do No Harm" level of mission assurance (Read et al., 2016) and then add on what is needed and desired by the stakeholders to reach the final risk posture approach. Starting at a "Class A" approach and tailoring back is not only extremely cumbersome for resource-constrained missions, it also provides no fallback position should program realities change.

Appendix A discusses the typical characteristics of constraints-driven missions, and the evolution of the tailored mission assurance concepts in general, in more detail.

3. Class Agnostic Mission Assurance

The Agile Class Agnostic Mission Assurance approach is a heuristic approach that focuses assurance efforts based on specific risks to mission objectives. As summarized above, it borrows concepts from agile software development to scale and manage mission assurance efforts according to the needs of the program. It accommodates changes in mission scope, risk posture, and priorities as the mission development evolves.

The result is a more appropriate application of limited resources across the mission, and a more realistic expectation of mission success that ultimately supports Certification of Flight Readiness (CoFR) at launch.

Figure 3-1 illustrates the overall approach. A more detailed discussion of the steps to this approach follows.

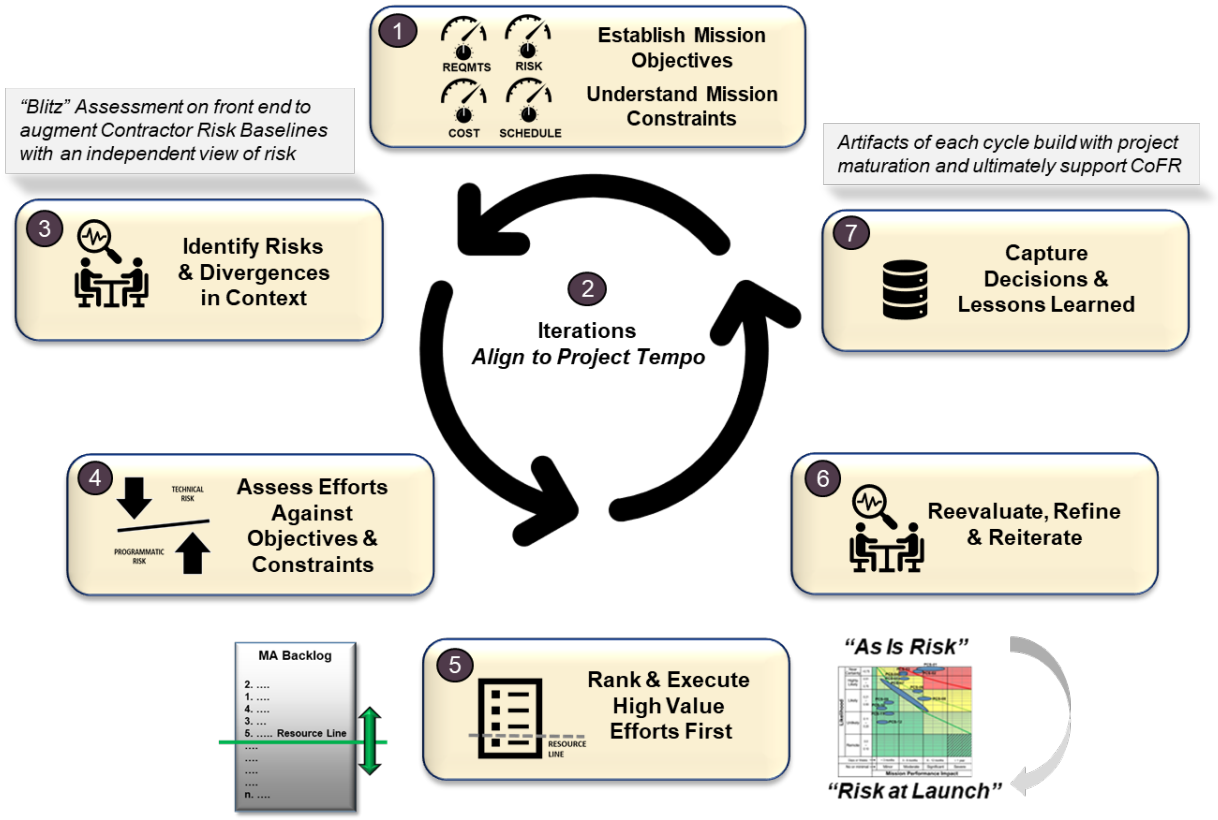


Figure 3-1. Agile Class Agnostic Mission Assurance Approach.

3.1 Step 1: Establish the "Knobs" of the Mission

To successfully apply the class agnostic heuristic programs must understand not only the goals of the mission, but also the constraints (schedule, cost, resources, etc.), and the relationship between them. This includes not only developing an idea of what minimum mission success might look like, but whether the scope of the mission matches this vision, and what level of risk the program is willing to accept, at least initially.

3.1.1 Establish Mission Objectives

The first step is to explore the mission's objectives. On some programs the objectives might be obvious at inception, but on other programs, the situation might be more subtle (or opinions might differ between stakeholders and developers). Many methods for defining and communicating the mission scope are viable; options might include one, or multiple, of the following: metric-based initial success criteria, minimum viable product (i.e., the minimum capability the spacecraft could fly with), use-case and user-story creation, requirements definition and derivation, or experiment plan and concept of operations.

As an example, the STPSat-1 mission team developed a very concise and specific list of minimum, nominal, and goal objectives, as follows:

- Minimum mission success:
  - *Take at least four SHIMMER data collects and download the associated data with a data quality of 99.5%.*
  - *Perform at least 60 hours (not required to be consecutive) of CITRIS operations.*
- Nominal mission success:
  - *Perform four SHIMMER data collects per day, for at least 300 days over the mission year. Download all experiment data with a data quality of 99.5%.*
  - *Operate CITRIS continuously over the mission year. Experience no more than 60 days (not consecutive) of lost experiment time.*
- Maximum mission success:
  - *Perform eight SHIMMER data collects per day, for at least two years. Download all experiment data with a data quality > 99.9% (this was actually achieved).*
  - *Operate CITRIS continuously for two years with no more than five lost days (close, but no cigar).*

Jasper, Braun, and Hunt provide another example – a risk-reduction mission for a high-speed communications system (HSCS) – which establishes the following objectives, in priority order (Jasper et al., 2020):

1. Minimum: File transfer downlink. This is the primary use of the HSCS for the large mission
2. File transfer uplink
3. Full data rate for file transfers
4. Command uplink through HSCS
5. Communication established at multiple ground sites on orbit
6. Goal (not in baseline): Telemetry downlink through HSCS

The Air Force Research Laboratory has several similar examples of success criteria in their configuration processes (AFRL/RV, 2020).

These examples capture the essence of the class agnostic concept of objectives: not so much a requirement list as a story or a set of statements about what the mission is supposed to do. The objectives

can include minimum, baseline, and goal criteria, but should be concise – ideally not more than one briefing chart. For constraints-driven missions, it can be particularly important to define what constitutes "minimum functionality" – i.e., what is "good enough" to launch?

3.1.2 Understand Constraints

Having identified the mission objectives, the team can now begin identifying the mission's constraints, such as cost, schedule, resources, size, etc. These are usually straightforward to list, but harder to match appropriately to the mission scope. The team must make realistic assessments as to whether objectives and constraints match each other. Figure 3-2 (Bitten et al., 2013) and Figure 3-3 illustrate the risk incurred when stakeholders have lofty expectations for a heavily resource-constrained missions. A $1M CubeSat is unlikely to deliver the performance or reliability of a $100M larger satellite, and the program should not expect such miracles unless it is willing to pay for them. One of the major lessons learned from studies of CubeSats is to ensure that missions are scoped appropriately for the vision (Venturini, 2017).

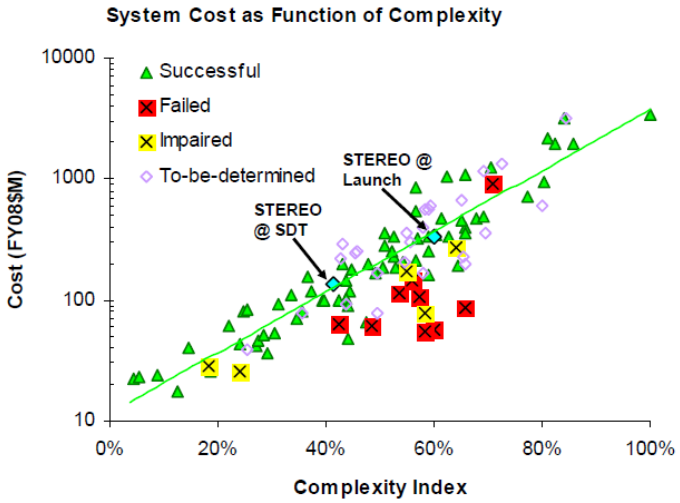


Figure 3-2. System cost as a function of complexity (Bitten et al., 2013).

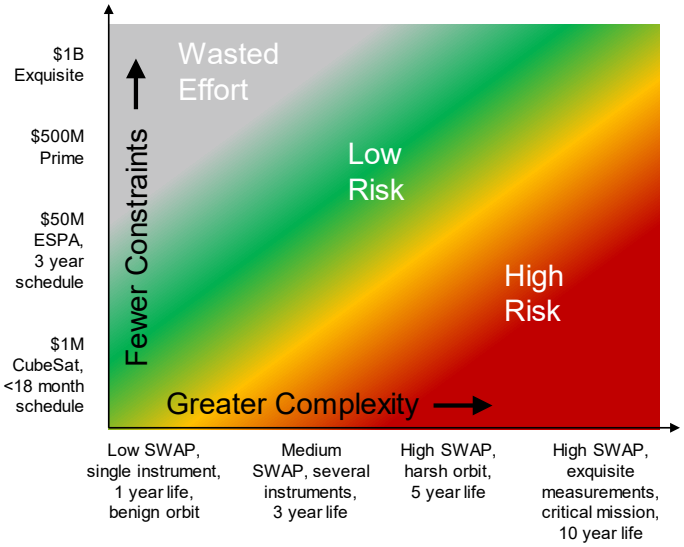


Figure 3-3. Notional risk when matching mission scope to constraints (Venturini, 2017).

3.1.3 Decide Whether Requirements or Constraints Drive the Mission

Once the objectives and constraints of the mission are described, the program and its stakeholders should jointly decide whether the mission is predominantly requirements-driven or constraints-driven. As described earlier, a requirements-driven mission will prioritize meeting mission objectives over staying within cost and schedule (or other mission constraints), while a constraints-driven mission is willing to sacrifice objectives to stay within strict cost, schedule, or other limitations such as those imposed by the CubeSat form factor or other standard architectures.

The requirements-driven / constraints-driven distinction is fundamental to successful communication among stakeholders and developers. Stakeholders and developers should agree up front on whether the mission is requirements-driven or constraints-driven, and this decision should be revisited often – and formally overturned if necessary. Too many missions claim to be constraints-driven when the initial budget is set but become more and more requirements-driven as launch approaches.

3.1.4 Articulate an Initial Risk Posture

Developers and stakeholders should jointly understand the initial risk posture of the mission. Risk can arise from objectives and constraints that do not match each other (as described in Section 3.1.2); it can be part an active decision to attempt something new and unproven; it can arise from programmatic decisions to trade assurance for some other desirable attribute (rapid development, spiral development, etc.); or it can simply be part of the team's mindset. But it should be part of the entire team's mindset since experience from past CubeSats (and small satellite missions) shows that stakeholders and developers will sometimes have different concepts of the risk posture of the mission. (Venturini et al., 2018)

In the past, customers have used traditional Class A/B/C/D designations to denote risk tolerance categories and such designations can still be used (Johnson-Roth & Tosney, 2010). However, few missions truly fit within a single, neat risk category (i.e., all within Class A, B, C or D). For these missions, other methodologies allow tailoring of risk level by subsystem or specialty engineering. This approach allows missions to focus their mission assurance on areas of higher criticality, while accepting more risk in lower-criticality areas. Alternatively, it allows programs to pay more attention to areas that are less mature, and less attention to areas that have heritage (Taylor et al., 2019). Additionally, the traditional designations for the Class A/B/C/D risk posture paradigm may have a risk "floor" that is too high to reap the full benefits of low-cost, risk-tolerant missions. Recent work has provided a sub-class-D taxonomy that provides a useful vocabulary for stakeholders and developers to use when discussing risk posture for highly risk-tolerant missions. Figure 3-4 summarizes this taxonomy, and more details can be found in the referenced paper (Jasper et al., 2018).

Risk Posture	Goals and Implications
Class A, B, C, D (payload performance driven by requirements)	Full Mission Success. Full Functionality
Class D (payload performance driven by constraints)	Full Mission Success. Functionality Limited by Constraints
Minimum Functionality	Min. Mission Success. Mission Recoverable in event of fault: Ex: LEOPS/start up Ex: Maintain Formation
Survival	Not DOA (power + low-rate comm). May have no higher level functionality
Do No Harm	DOA is ok (education and/or fully constrained and not requirement driven)

Figure 3-4. Risk taxonomy for small satellite missions (Jasper et al., 2018).

Starting at the minimum level of functionality of Do No Harm (basically, accomplishing only the assurance required to comply with launch and safety requirements) and tailoring up can be particularly helpful for smaller, more risk-tolerant systems. These lower levels of assurance also provide a fallback position for when the risk "knob" needs to be turned in response to changing circumstances.

Regardless of the specific label used, the overarching goal of the initial risk posture is to communicate, not to rigidly classify. The risk posture of the mission rarely stays fixed throughout a mission. Even requirements-driven missions are sometimes forced to accept risk, and it's a common experience for developers to encounter missions that are constraints-driven at kickoff – but requirements-driven at Launch Readiness Review. Or as one of the authors famously stated, "everyone is willing to tolerate risk, but no one is willing to accept failure."

3.1.5 The "Knobs" Are Not Fixed

The items described above in sections 3.1.2 through 3.1.3 – loosely categorized as technical requirements, risk posture, cost constraints, and schedule constraints – represent "knobs" for mission execution (See Figure 3-5). Depending on the mission objectives and constraints, each of these "knobs" may be fixed or variable – stakeholders may be more willing to adjust cost and schedule to meet requirements or may be more willing to relax requirements and accept risk to meet cost and schedule. The settings may also change during the progression of the program. However, missions cannot expect all four knobs to remain fixed for the duration of the mission. No program is without issues or changes, and the knobs represent the program's flexibility to absorb changes and address issues. Requirements-driven missions will generally aim to solve technical issues keeping a low risk tolerance while accepting cost and schedule changes whereas constraints-driven missions will address technical issues by reducing scope or accepting additional risk to stay within cost and schedule constraints.

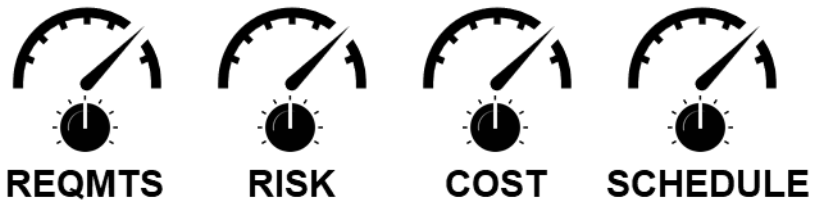


Figure 3-5. Mission design variables.

Early designations of mission risk tolerance before any real knowledge of the technical issues that will arise during development (and the stakeholders' willingness to tolerate those risks) can therefore be baseless. Mission assurance approaches that rely on fixed assumptions are not flexible to the uncertainty in the mission.

3.2 Step 2: Align Iterations to Project Tempo

It is important to recognize that this is an agile and iterative approach. The full cycle is aligned with the project tempo, ideally takes no more than a few weeks, and is continually repeated. This carries through major milestones and ultimately delivers Certification of Flight Readiness (CoFR) at Flight Readiness Review (FRR). Figure 3-6 shows the iterative nature of this agile approach in context of the full mission life cycle.

These iterations optimize mission assurance activity based on the risks to mission success objectives as you go. It also burns down risk to an acceptable level within the overall mission risk tolerance that is agreeable and well understood among stakeholders thus creating more realistic expectation of mission success for supporting CoFR. The first cycle will likely take longer than those following since more time is required for establishing a baseline of risks, divergences, and responses.

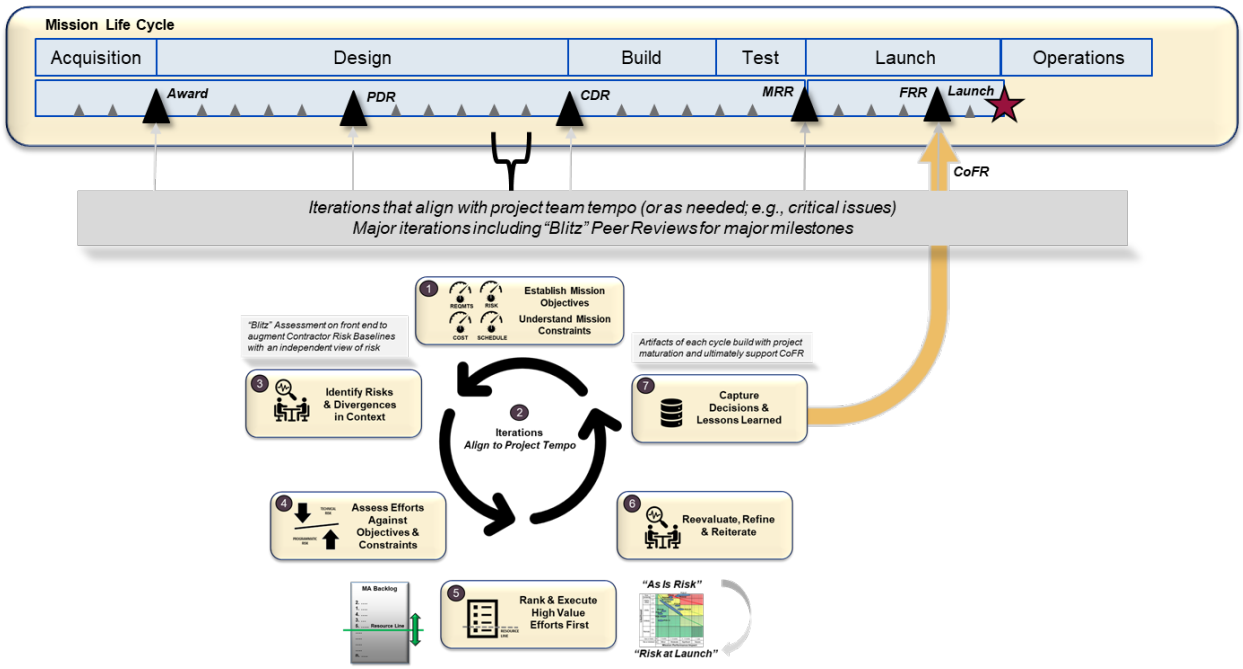


Figure 3-6. Alignment to Project Tempo over mission lifecycle.

3.3 Step 3: Identify Risks and Divergences in Context

Once the mission's initial "knobs" are established, the program can begin to think about risk. Loosely speaking, risks are anything that might cause the program to adjust one of its knobs. When thinking about risk, mission context is important. For example, a lack of battery conditioning might not be a risk for a one-year LEO mission, while it likely poses a significant risk to a ten-year GEO mission. Similarly, contamination can be a significant risk to some missions, and not to others. The orbit, lifetime, and intent of a mission provides the context within which to evaluate risk.

3.3.1 Identify Risks and Divergences

Programs will generally want to conduct an initial assessment of the risks of a mission. There are many tools and frameworks for identifying risk areas and common pitfalls. General examples for guidance are ISO 17666, *Space Systems Risk Management*; the *Risk, Issue, and Opportunity Management Guide for Defense Acquisition Programs*; and NASA NPR 8000.4, *Agency Risk Management Procedural Requirements*. The initial risk assessment is also where a "blitz" of Aerospace attention from subject-matter experts (if the program can afford it) can be especially effective.

Approaches to identifying high-risk areas of a mission such as those described in Section 3.1.4 can also provide an initial idea of where a program needs to focus its attention. Areas to consider include:

- Generally-agreed-upon critical areas (e.g., power, communication, Do No Harm, safe modes, interfaces)
- Areas of specific concern for this mission (contamination, EMI, radiation, non-space parts, previous failures)
- New items (first flight, changes from last time)
- Items from the Mission Assurance Baseline (MAB) or tailored MAB
- Lessons learned / "gold rules" (software, polarity, test like you fly)
- Expert opinion and experience

The risks that emerge from this initial pass typically represent known and "known unknown" risks. The "unknown unknowns" will generally emerge during the execution of the program. The mission assurance approach needs to be flexible enough to adjust.

The team should always keep the context of the mission in mind. Something that is a risk for a system with a 10-year lifetime might not be a risk for a mission with a lifetime of one year. Remember that success is not always binary. Limited or degraded system performance may still achieve mission objectives. There is always a range of risk trades and dispositions that can be exercised to achieve the optimal operational performance within the mission constraints. It may be possible to accept risk that only degrades performance or threatens performance that is marginally important to the mission objectives.

The result of the initial assessment is often a full laundry list of issues, deficiencies, and risks from various disciplines, but this initial assessment is just the beginning. It is important to keep in mind that risk lists (like requirements lists) are products of the human imagination and thus never guaranteed to be complete, self-consistent, or fully accurate. Programs must revisit the risk assessment process frequently, as part of agile design, build, and test iterations, preferably at the start of each iteration.

3.3.2 Identify Potential Reduction Efforts

Once risks are identified, the team should consider both specific and overarching risk reduction efforts. These can be specific actions, like the following:

- Investigate GPS dropouts over the poles and recommend firmware updates
- Add a modal survey test to the test campaign
- Add a redundant receiver to the design
- Add Reed-Solomon encoding to the downlink
- Test all COTS parts upon receipt

They can also be more overarching, general activities, like the following:

- Analyze the thermal performance
- Review all ICDs for discrepancies
- Witness testing
- Validate Do No Harm artifacts
- Add a review or a standing meeting with high-risk mission element representatives
- Plan for a two-week review of some mission deliverables in the mission schedule to allow for subject-matter expert input as necessary

Missions should start with broad concepts and refine as necessary. Early in the program, risks are generally broader and based on general principles; later, risks become more specific and are more often related to observed failures or deficiencies. The team should tie reduction efforts to specific risks where possible, but this doesn't have to be a one-to-one mapping. While the team should take care to make the risk statements crisp and actionable, the team should remain flexible and avoid getting hung up on specific details. The goal is to understand if constraints will be violated or objectives may not be met (e.g. mission success), not to achieve exquisite risk documentation.

The first risk list is never complete; indeed, the latest risk list is also never complete. The power of agile iterations is that it allows risks that weren't identified in earlier iterations to be uncovered in later iterations, and the program evolves toward completion.

3.3.3 The Role of Peer Reviews

Although not required by the class agnostic heuristic, many programs find peer reviews helpful in the risk identification step. Peer review serves two primary purposes: (1) to provide technical input for the team on their design, and (2) understand risks as they emerge. These reviews can be small one-on-one meetings or subject matter expert / team interactions, but the key is that they serve to provide the engineers actionable feedback. Ideally some external reviewers participate to provide perspective and reduce groupthink.

The team can use these peer reviews to adjust their designs as necessary and to identify potential divergences from the current scope or constraints. These possible divergences should be actionable, tradeable risks – not "paranoia" or "what-if" exercises.

3.4 Step 5: Assess Efforts against Objectives and Constraints

Once the team has the initial – or most recent iteration – of the risk list, it's time to estimate the level of each risk and the amount of reduction possible, as well as the effort required to conduct each risk reduction action. The amount of risk reduction expected for each mitigation action, and the effort estimated for each action, help the team estimate the "bang for the buck" of each action – an estimation central to the class agnostic concept.

While several frameworks are discussed here, teams should keep the Agile mindset in mind. The goal of assessing risks and efforts is to apply resources where they will do the most good – not to produce perfect estimates, or perfect risk analysis charts. In many missions, the whole effort is rather informal. The estimates are made to inform decision-making, not to make statistical predictions. Documentation should be focused on providing the rationale behind the decisions made, rather than on achieving a certain "magic number" of risks or an exact estimation of "bang for the buck."

In all cases, keep the context in mind – the program's objectives, constraints, and risk posture will drive the estimation process. A high risk on a Requirements-driven mission might be a baseline risk on a Constraints-driven mission.

3.4.1 Estimating Risk and Risk Reduction

Teams frequently get hung up in quantifying risk. The standard 5x5 risk matrix shown in Figure 3-7 provides a useful tool for identifying risk levels by likelihood and consequence, but has pitfalls.

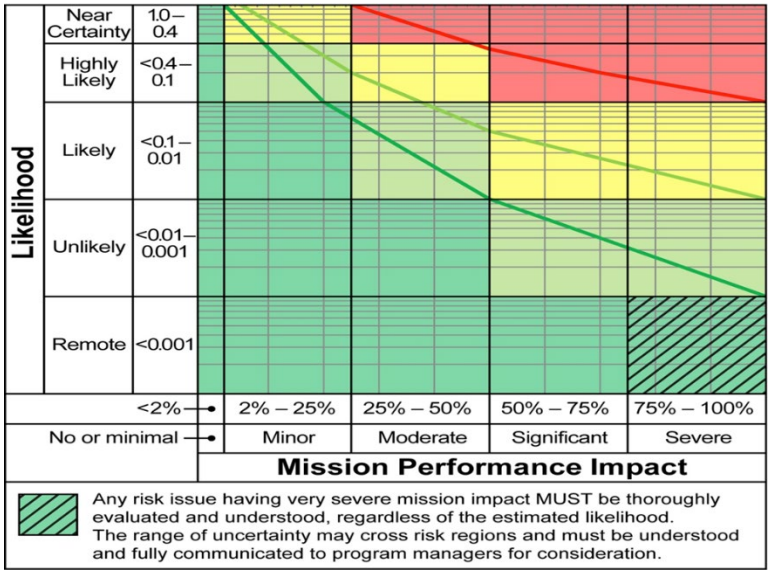


Figure 3-7. The standard 5x5 risk matrix (Guarro, 2011).

Chief among these pitfalls is the tendency to view the numerical numbers on the axes as statistically significant, data-driven values. Some risks can be quantified in this manner, but on one-of-a-kind missions, the team is unlikely to have enough real data to be able to give a statistically solid number for the true likelihood of an event. This is especially true on constraints-driven missions, and research and

development missions, for which the true likelihood of failure is unknown. Another pitfall is the tendency for risks to span categories across the 5x5 risk matrix. A single issue – for example, the failure to conduct a true "week in the life" test – may be relatively likely to result in minor issues, but relatively unlikely to result in severe issues. Real risks don't adhere to simple categories and trying to force-fit them into their exact box usually wastes a lot of time and effort that could instead be focused on resolving issues.

Instead, the class agnostic heuristic recommends adopting a more Agile mindset to risk estimation. Teams may choose to use the 5x5 risk matrix, but except for very clear-cut, quantitative risks, teams should consider the numerical values in the risk matrix as guidelines. Different stakeholders may have different interpretations as to what each of the "likelihood" categories mean; one may consider something likely if it occurs about 10% of the time; another might consider something likely only if it occurs more than 50% of the time. The use of numerical values helps define what each of the likelihood categories means in a way that can be understood by all stakeholders. In most cases, however, they should not be interpreted as a statistically evaluated numerical estimation.

Some programs – especially smaller, constrained programs – might consider a simplified 2x2 risk matrix, instead, as shown in Figure 3-8. While this might seem oversimplified, it helps programs quickly categorize major risks and focus their efforts on mitigating – rather than categorizing – them.

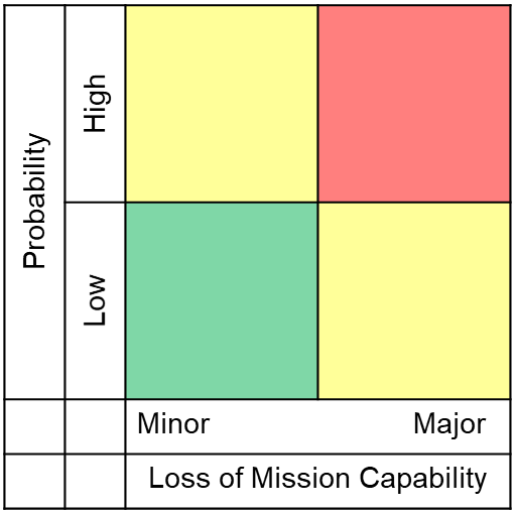


Figure 3-8. Simplified mission risk matrix.

Teams may also consider using other Agile methods like "planning poker" to estimate broad risk levels – whether a given risk is high, medium, or low – or can be used to estimate where a risk should fall on each axis the 5x5 risk matrix. Planning poker offers an informal less structured method that uses a game-like format to avoid bogging down while leveraging the full knowledge of the team. (Agile Alliance, n.d.)

Whatever method is chosen, teams must not only estimate the current risk level, but also the risk reduction made possible by each proposed mitigation action. This can be accomplished by showing the risk reduction on the 5x5 risk matrix as shown in Figure 3-7, or by playing planning poker again on while envisioning the end state. The difference between the "before" and "after" risk levels provides an estimate of risk reduction.

3.4.2 Estimating Efforts

Teams will also need to estimate how much effort is required to conduct each proposed risk mitigation action. These include broad actions, such as reviewing ICDs and witnessing testing, and specific actions, such as adding extra tests or making software / hardware changes.

One of the drawbacks of the 5x5 risk matrix is that it focuses only on the risk, not on the effort required to mitigate that risk. Especially for resource-constrained missions, risk reduction must take place in the context of the effort required. The most effective risk-reduction method is worthless if it results in program cancellation due to lack of funds.

Estimating the effort involved can include a simple guess at the number of staff-days or staff-weeks and cost involved, or it can involve another round of Agile "planning poker," where the focus is on the amount of effort it will take to implement the proposed mitigation action. (Indeed, Agile planning poker is most commonly used for effort estimation.)

3.5 Step 5: Rank and Execute High Value Efforts First

Once risks and actions have been defined, and the effort to accomplish each estimated, the team evaluates the "bang for the buck" for each risk / action set and ranks them in a method that allows the team to decide how to apply its mission assurance efforts.

It can do this by evaluating the ratio between technical risk reduction and programmatic risk increase for doing any given action. Technical risk is defined as risk against the already-established scope, as estimated in Step 3; programmatic risk is defined as risk against the cost, schedule, and resource restrictions already defined. In some heavily resource-constrained missions, the ratio might be inverted, and the team might instead consider the programmatic (cost or schedule) risk reduced for the technical risk incurred. For example, a typical trade in a constraints-driven program is to remove a secondary mission objective in order to maintain cost and schedule.

With the ratios defined, the team now ranks risk reduction, mission assurance, or even design efforts in order from most "bang for the buck" to least "bang for the buck". Note that the elements are not ranked in order from "highest risk" to "lowest risk," or even from "highest risk reduction" to "lowest risk reduction." The effort required to implement an objective or reduce risk is part of the assessment. "Low hanging fruit" may fall higher on the list than more serious risks that are harder to mitigate or more interesting objectives that consume more resources. The sum of the effort required to implement these actions (the sum of the programmatic risk) tells the program how many resources are required. Constrained programs will need to draw the line at the limit of their resources; items that fall below the line are not addressed unless more resources become available. Then the team executes the efforts roughly in order.

3.5.1 Visualization Approaches

One way to visualize this would be in a table of risk reduction efforts, listing the estimated amount of risk reduction, the confidence in that estimate, the estimated cost or effort required, and the confidence in that estimate. Figure 3-9 is a notional example.

ID	Item (Risk / Opportunity / Issue / Trade)	Description / Story	Response / Action	Benefit*	Benefit Confidence	Cost Margin Impact	Cost Confidence	Sched Margin Impact (weeks)	Schedule Confidence	Execute?
R-001a	Software Risk #1	Potential delay to schedule	Add to WITL with time to follow-up	20	High	$ 150,000	Medium	2	High	Yes
R-001b	Software Risk #1	Potential delay to schedule	Add Software peer review	40	High	$ 35,000	High	1	High	Yes
I-001a	IR Focal Plane Weld Cracks	IRFPA design welds show cracking	Correct Welding Process	40	Medium	$ 30,000	High	8	Low	No
I-001b	IR Focal Plane Weld Cracks	IRFPA design welds show cracking	Inspect Welds	20	High	$ 25,000	High	14	Medium	No
I-001c	IR Focal Plane Weld Cracks	IRFPA design welds show cracking	Mechanical Clamping	8	High	$ 30,000	High	2	High	Yes
I-002a	Finite Element Model	FEM not correlated; potential for structural failure	Add modal survey test	40	High	$ 250,000	High	4	High	Yes
I-002b	Finite Element Model	FEM not correlated; potential for structural failure	Add extra accelerometers to sine vibe testing	8	Medium	$ 25,000	High	1	Medium	No
O-001a	New OS for Software Opportunity	Could boost performance by 20%	Build portability to new OS	3	Medium	$ 50,000	Medium	2	Medium	No
					Margin Used =	93%	Margin Used =	75%		

Figure 3-9. Visualization example for space mission risk reduction trades.

Missions could use similar charts to show how their reduction efforts reduce their risk to match their overall risk posture – or where the cost of such efforts run up against the "hard line" of the resources available.

This is just one way to rank and execute risk reduction efforts. Missions may choose instead to do something simpler, like plotting risks along a programmatic / technical risk matrix as shown in Figure 3-10. Missions can then determine which actions to prioritize based on where they fall along green / yellow / red boundaries the team has drawn itself.

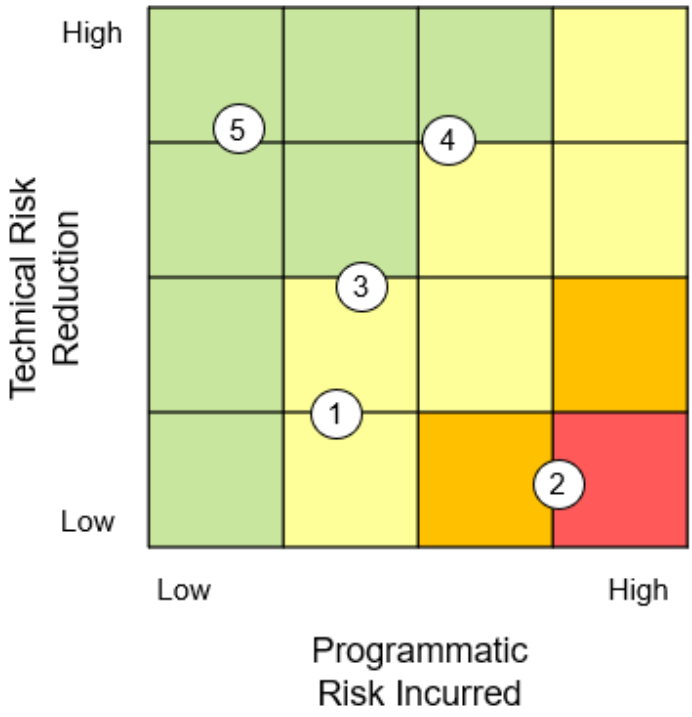


Figure 3-10. Visualizing risk trades with a programmatic/technical risk matrix.

3.5.2 Execute Efforts

With risk and risk-reduction efforts identified, ranked in order from most "bang for the buck" to least "bang for the buck," and discussed within the program, the team can now execute the efforts they have agreed provide the best mission assurance value.

3.5.3 A Word About Messiness

Mission assurance and risk management are rarely simple and elegant. The ranked risk list is a nice theory, but messier in practice – it is unlikely the mission will have a clean list of tasks in a neat order. In fact, like many of the other steps in the class agnostic heuristic, most of the authors have never actually produced an official ranked risk list but have instead operated on an intuitive feel for where assurance efforts provide the most bang for the buck.

Agile mission assurance is an art as much as it is a science. Teams should remember the Agile mindset, and that the main point of mission assurance is to improve the chances of mission success, not to document risks. Agile values individuals and interactions over processes and tools, and customer collaboration over checklists. If the tool or process doesn't work for your mission, use a different one!

3.6 Step 6: Reevaluate, Refine and Reiterate

The most critical part of the class agnostic mission assurance process, and the most related to Agile principles, is Step 6: Re-evaluate and Refine. As described in previous sections, early estimates of risk are usually the least accurate, and correspondingly, mission assurance plans developed early in mission execution are rarely applicable throughout the mission lifecycle. As a mission progresses, priorities will change, and new risks and issues will emerge. Some efforts will take more resources than expected, and some efforts will take fewer resources than expected. On a regular basis – whether that be after a two-week "sprint," monthly, or quarterly – the team meets to determine what has been done, what remains to be done, whether priorities have changed, or whether new information has emerged that might cause the team to re-direct mission assurance activities. If the mission is still operating within the agreed-upon cost, schedule, requirements, and risk posture "knob values," this can be done internally to the team. Having agreed upon the next set of priorities (or that the current efforts should continue), the team embarks on the next iteration through the cycle.

If enough issues have arisen that one of the "knobs" needs to be adjusted, the government, contractor, Aerospace team should engage leadership and, if necessary, key stakeholders. Leadership may, in discussion with stakeholders, adjust the constraints of the mission, adding funds and schedule or reducing scope and accepting more risk, in order to absorb the changes that arise. This is where communication with stakeholders can be key. Programs may delegate decision authorities differently depending on the size and criticality of the mission, but in general, only leadership and / or mission stakeholders can change one of the "knobs" of a mission. At a minimum, programs will need to inform leadership and key stakeholders if adjustments to the knobs are necessary.

Teams may use the Agile class agnostic cycle in the context of mission milestones. For example, a team might make use of peer reviews after every few sprints to identify areas of concern and point out where design and mission assurance efforts might be overlooking key risks. A larger examination of the overall "knobs" of the mission might occur at major programmatic reviews, when developers, leadership, and stakeholders can all meet to review program status and decide if any of the "knobs" need to be adjusted.

3.7 Step 7: Capture Decisions and Lessons Learned

During the execution of the class agnostic mission assurance cycle, teams should look both inward and outward. Within the mission, teams should document all decision-making. As decisions, trades, and adjustments to the "knobs" of the mission are made, it is critical that teams capture the rationale behind these changes. Not only does this help maintain continuity across personnel and leadership changes, it helps prevent "risk aversion creep" by keeping the entire team aware of what trades have already been made between cost, schedule, risk, and requirements – and why. This history is particularly important as

the mission approaches launch, when cost and schedule risks are all in the rear-view mirror, and leadership is most concerned with technical risk.

Additionally, missions should be outward-looking, and should document lessons learned frequently during mission execution. Capturing lessons learned after a certain number of "sprints," or at a minimum at each major milestone, ensures that lessons are documented while they are still fresh. Reviewing key lessons learned from past missions at each milestone can also help the team anticipate problems that might occur during the next phase. Lessons-learned become sources for evaluating risks for future missions (see section 3.3.1).

While process documentation, lessons learned, and reference material are important, truly efficient mission assurance requires apprenticeship. Documents and "how to" manuals have their place but are no substitute for experience. Young engineers should be paired with experienced engineers to help them develop good instincts.

4. Conclusion

Like all Agile approaches, the class agnostic mission assurance heuristic is a living process, and the approach outlined here is not meant to be prescriptive. Appendix B illustrates how variations of this approach have been tailored to specific customers and programs, and future programs considering a more agile approach to mission assurance should consider how the class agnostic approach might best apply to them.

Even within the execution of the class agnostic mission assurance cycle, program managers and the mission assurance team will need to continuously evaluate what is working for them, and where improvement is needed. Rarely will any given mission follow the process outlined here in its entirety, with formal risk evaluations and a formal list of efforts to be executed in a strict order. Teams are encouraged to "embrace the messiness" of the process, and recognize that good mission assurance is an art, not a science (or a checklist). The Agile mindset encourages teams to re-evaluate processes regularly and discard those that don't work, and this approach is no exception.

It is the hope of the authors that the class agnostic heuristic presented here will encourage the mission assurance community to think outside the Class A-D box for holistically addressing technical risks as we strive towards the goal of properly balancing risk mitigation within a programmatically constrained environment. The authors also hope that mission teams will take on the challenge of adapting and evolving this approach, developing their own tools and approaches, and sharing their findings with the rest of the community.

5. References

[1] Agile Alliance. (n.d.). *Agile 101 - What is Agile Software Development?* Retrieved October 13, 2020, from https://www.agilealliance.org/agile101/

[2] Ambler, Scott. (n.d.). *Examining the Agile Manifesto*. Retrieved October 13, 2020, from http://www.ambysoft.com/essays/agileManifesto.html

[3] Bitten, R., Mahr, E., Kellogg, R. (2013). *Cost Estimating of Space Science Missions* (Report No. ATR-2013-00108). The Aerospace Corporation.

[4] Department of Defense. (2017). *Risk, Issue, and Opportunity Management Guide for Defense Acquisition Programs*. http://acqnotes.com/wp-content/uploads/2017/07/DoD-Risk-Issue-and-Opportunity-Management-Guide-Jan-2017.pdf

[5] Guarro, S. B. (2011). *Mission Risk Assessment Process and Techniques for APR* (Report No. ATR-2012(9012)-1). The Aerospace Corporation.

[6] International Organization for Standardization. (2016). *Space Systems - Risk Management* (ISO Standard No. 17666:2016). https://www.iso.org/standard/33149.html

[7] Jasper, L., Hunt, L., Voss, D., and Jacka, C. (2018, August 4-9). *Defining a New Mission Assurance Philosophy for Small Satellites* [Paper No. SSC18-WKII-05]. 32nd Annual AIAA/USU Conference on Small Satellites, Logan, UT, USA. https://digitalcommons.usu.edu/smallsat/2018/all2018/431/

[8] Jasper, L., Braun, B., and Hunt, L. (2020, March 8-13). *New Constraint-Driven Mission Construct for Small Satellites and Constrained Missions* [Paper presentation]. IEEE Aerospace Conference, Big Sky, MT, USA.

[9] Johnson-Roth, G., Tosney, W. (2010). *Mission Risk Planning and Acquisition Tailoring Guidelines for* National *Security Space Vehicles* (Report No. TOR-2011(8591)-5). The Aerospace Corporation.

[10] National Aeronautics and Space Administration (NASA). (2017) *Agency Risk Management Procedural Requirements*. (NPR 8000.4B). https://nodis3.gsfc.nasa.gov/displayDir.cfm?t=NPR&c=8000&s=4B

[11] Read, A., Chang, P., Braun, B. Voelkel, D. (2016). *Rideshare Mission Assurance and the Do No Harm Process* (Report No. TOR-2016-02946). The Aerospace Corporation.

[12] Taylor, A., Becklund D., Rast S., Ayers J. (2019). *Adaptive Mission Assurance Strategy for Pre-Acquisition: Phase 1* (Report No. TOR-2019-01781). The Aerospace Corporation.

[13] US Air Force Research Laboratory Space Vehicles Directorate AFRL/RV (2020). Configuration Management (CM) and Configuration Control Board (CCB) Process Version 4.0.

[14] Venturini, C. (2017). *Improving Mission Success of CubeSats* (Report No. TOR-2017-01689). The Aerospace Corporation.

[15] Venturini, C., Braun, B., Hinkley, D., Berg, G. (2018, August 4-9). *Improving Mission Success of CubeSats* [Paper No. SSC18-IV-02]. 32nd Annual AIAA/USU Conference on Small Satellites, Logan, UT, USA. https://digitalcommons.usu.edu/smallsat/2018/all2018/431/

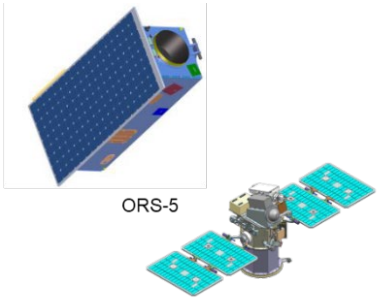
Appendix A. Constraints-Driven Mission Attributes and the Evolution of the Space Innovation Directorate's Class Agnostic Mission Assurance Approach

A.1 Constraints-Driven Mission Attributes

A.1.1 Program Attributes

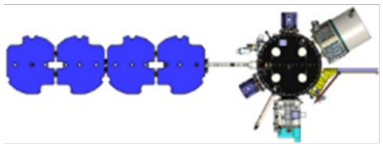
Small, risk-tolerant, constraints-driven missions are distinct from ACAT 1, Risk Class A Missions that demand greater system reliabilities for operational availability and resilience. It is these distinctions that, when leveraged, enable the efficiencies necessary for meeting mission success objectives within constrained resources and schedule.

Programmatically, these missions are typically smaller systems for demonstration, technology maturation, and / or deploying a small component of an overall capability where resilience is delivered at the architecture level. They possess fewer requirements, condensed management teams, and a necessity to trade system performance requirements and risk for a return in cost and schedule benefit. Customer mission managers and their leadership must be willing to accept additional risk. Most importantly, the mission team must have the freedom to fail.



ORS-5

STPSat-5



EAGLE and Mycroft

STP-2 Integrated Payload Stack

Contractually, efficiencies are needed for remaining within constrained budgets and schedules. Efficiencies are realized by leveraging developer processes, tailoring statements of work and limiting mission assurance deliverables to only that which is critical to achieving good enough performance that still meets mission objectives. A low risk tolerant mission team pursuing full system performance could never execute "Class A" mission assurance without contracting for the appropriate developer support and deliverables. Similarly, a mission team operating on a shoestring budget and accelerated schedule cannot contract for a full complement of mission assurance support and deliverables. Since mission success is defined by objectives rather than system performance, it would be inefficient to purchase support and deliverables that provide only diminishing returns when good enough performance is adequate. Finally, smaller mission teams will rarely have the additional time or people to properly assess a full complement of support and deliverables anyway.

A.1.2 Technical Attributes

Looking at system design, the small, risk tolerant space vehicles are usually single string systems designed for a one to three-year design life with acceptable mission reliabilities ranging between 40% and 90% over a one-year mission life. These systems can, most of the time, tolerate operational down times but also have a built-in capability for failing gracefully into a known state for recontact, troubleshooting, anomaly resolution, and reboot of the system. This is why simply trying to tailor Class A standards and prescriptive requirements for an existing Class A designed system will not result in the same cost and time savings as for starting with a streamlined design in the beginning.

For manufacturing, developers tend toward a higher usage of commercial parts. There is rarely a spares concept beyond that of sparing at the system level; the space vehicle itself. In the case of large constellations of smaller satellites, the satellites themselves are the sparing concept since capability is delivered at the architectural level. This is the case for CPA which is discussed later. Small satellite ground systems, typically servicing multiple satellites at a time, will usually have a sparing concept for ensuring availability.

A.2 Constraints-Driven Risk Tolerance

Constraints-driven missions must accept risk and manage that risk along with stakeholder expectations of mission success. There is almost always a willingness to accept risk at the start of a mission development that gives way to an unwillingness to accept mission compromise or failure at launch. Constraints-driven missions still rigorously manage risk, but they accept the reality of significant residual risks given constraints on budget and schedule. This includes the possibility of accepting medium to high residual risks at launch. It may make more sense to launch with a 50% chance of success today than to accept a 0% chance if the team misses a rideshare opportunity or if mitigation costs threaten mission cancellation. Shown here are two missions and their risk matrices at launch.

ORS-1 is a great example being schedule constrained due to a warfighter urgent need. In Figure A-1, one can see that there were significant risks, including red & yellow risks, at launch which indicates the increased risk tolerance of the ORS-1 program. It was better to launch with a red risk than deny an urgent capability to the warfighter. In fact, the red risk identified was accepted at the beginning and despite this, ORS-1 launched after a three-year development and was successful on orbit.

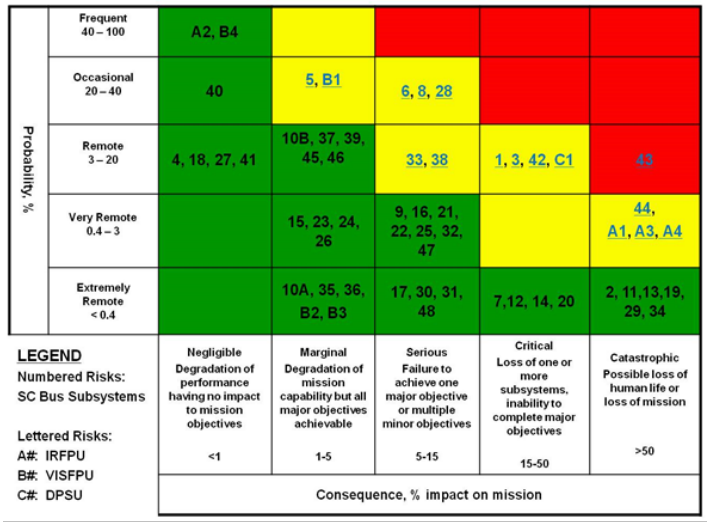


Figure A-1. ORS-1 mission risk assessment at launch.

STPSat-3 is another example being budget constrained. In Figure A-2, one can see that there were many risks, including yellow risks at launch also indicating an increased risk tolerance. This mission also launched after a three-year development and was successful on orbit.

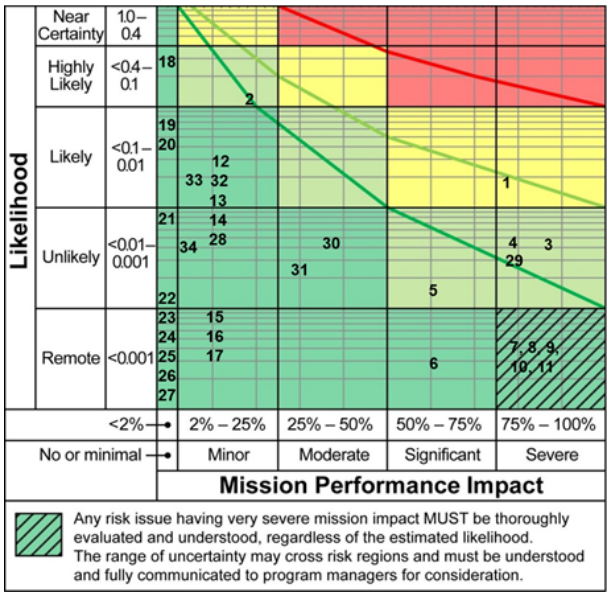


Figure A-2. STP-Sat-3 mission risk assessment at launch.

A.3 Spectrums of Mission Assurance

Figure A-3 is a quad chart that shows the spectrum of constraints versus requirements driven missions along examples for those types of missions.

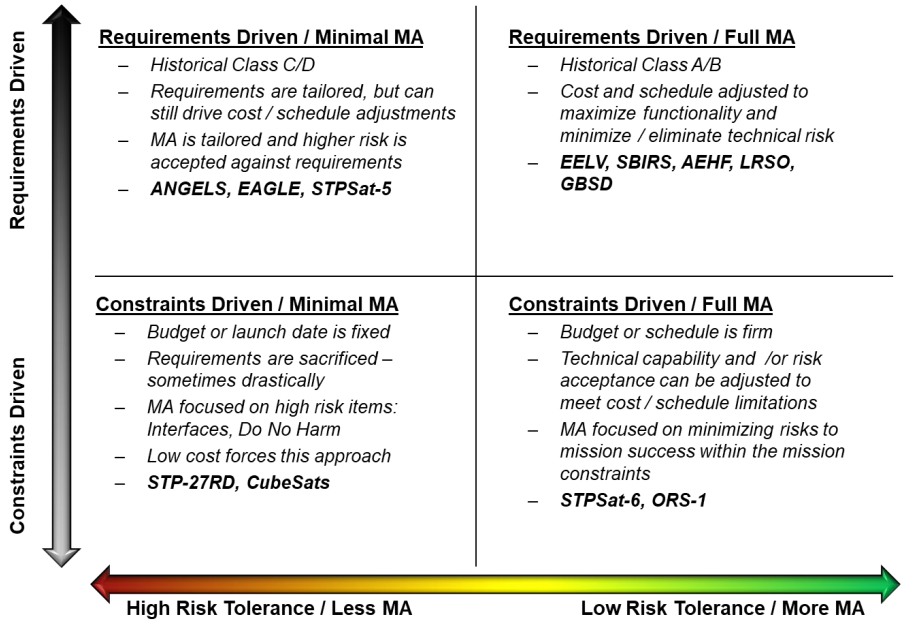


Figure A-3. Spectrum of mission assurance for constraints and requirements-driven missions.

In the upper right, there are traditional Class A/B missions, where requirements drive the mission. Though everyone aims to launch on time and on budget, in the end, for these types of missions, the requirements are not compromised.

At the other extreme are the many typical CubeSats. Usually built by universities, CubeSats often rideshare along with other primary missions, and therefore have a fixed launch date. Most CubeSat builders will aggressively descope their missions as needed to meet this fixed launch date. As on put it, "we'd rather take a 5% chance of it working on orbit, than a 0% chance of it ever launching." STP has recently launched a similar mission (STPSat-5) that was manifested on a commercial rideshare, and could not therefore influence the launch schedule. The mission team accepted data quality issues and operational constraints to stay within budget and launch on schedule. They have had issues on orbit and have had to take time during LEOP to solve them, but they will eventually get to operational status for achieving some degree of mission success.

In the other corners are missions more typical of STP, such as STPSat-2 or STPSat-3, which although they were more fundamentally requirements driven, still aimed to meet requirements with streamlined mission assurance. And ORS-1 had a fixed schedule sacrificing technical performance where needed to maintain that schedule. But ORS-1 applied a more traditional mission assurance approach which is more comprehensive than one would apply to a typical CubeSat or STPSat-5.

Figure A-4 shows that SID supported missions have, over the decades, migrated becoming more constraints-driven missions with minimal mission assurance (lower left corner). This trend has further evolved how SID approaches mission assurance; specifically, an agile constraints-driven approach that is agnostic to a specific risk class.

Figure A-4. Migration of SID missions to constraints-driven mission assurance.

A.4 Evolution of Class Agnostic Mission Assurance

The move toward more constraints-driven, minimal-mission-assurance missions has caused the SID mission assurance approach to evolve from a more traditional mission assurance approach adapted for constraints-driven missions to an agile risk class agnostic version that is used today. Importantly, each of these models remains relevant (i.e., tools in the SID toolbox) for use depending on the mission need. Figure A-5 depicts the evolution to date.

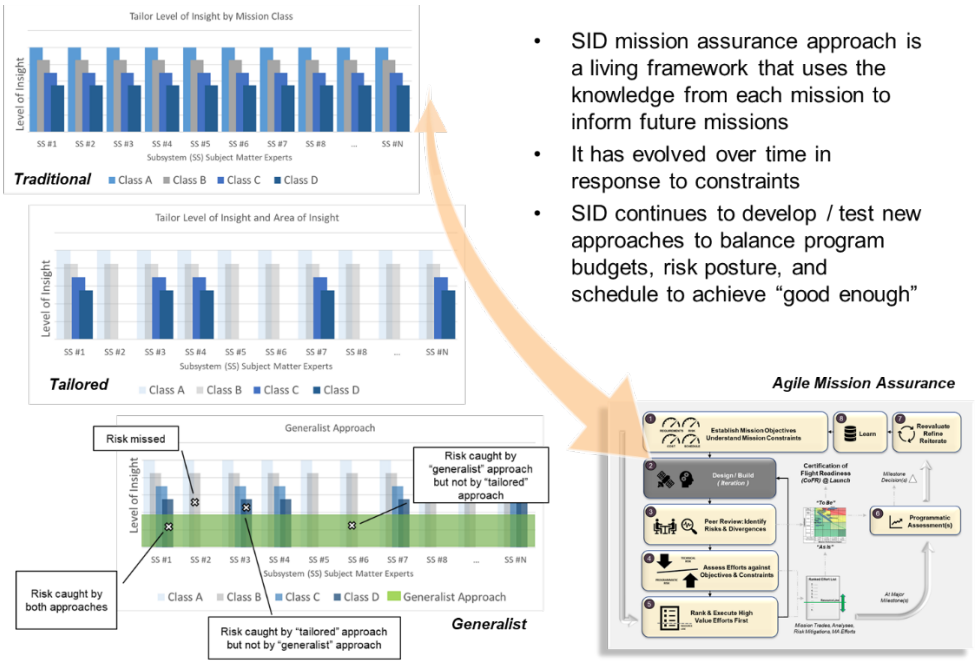


Figure A-5. Evolution of mission assurance for constraints-driven missions.

In the top left corner, there is the "traditional" Class A-D spectrum of mission assurance approaches that tailored level of insight based on the mission risk classification (A to D). This assumes that Class A requirements-driven missions (i.e., Class A/B) will have the resources available to dig down into each mission area for the full depth of insight, while constraints-driven missions (i.e., Class C/D) will be willing to accept the additional risk associated with the lower level of insight. This traditional approach assesses all aspects of the mission (e.g., all SV subsystems, launch segment, ground segment, etc.), reaching back for all applicable SMEs. It provides assurance that the space system will operate properly on-orbit by verifying requirements, ensuring adequate environmental testing, and providing independent risk assessments from appropriate experts.

As resources become constrained, the mission can tailor further by concentrating limited mission assurance efforts into specific areas. This "Tailored" mission assurance model "tailors" both the breadth and depth of insight (i.e., which specific subsystems or areas will be evaluated). Subsystems or segments that are deemed lower risk (e.g., a production line, use of heritage components or more extensive testing) are tailored out for specific evaluation by dedicated SMEs. This frees up scarce resources to assess the higher risk subsystems or segments (e.g., modifications to a production run, changes in components, or "Do No Harm" verifications). This more focused expertise leverages what is already known to provide reasonable assurance that the system will operate properly on-orbit.

But if broad coverage of subsystems is desirable and resources are insufficient to achieve the most minimal level of insight across all SMEs and experts there is the "generalist" approach. The generalist approach offers a better opportunity for identifying risks that the tailored approach could miss without having to expend the resources required of the traditional approach. This approach replaces SME expertise in every area with three to four generalist system engineers who possess seasoned experience (or "Spidey Senses") to screen for problems requesting additional SMEs and experts as needed.

This was first initiated on STPSat-5 employing four individuals from different areas of the company. This can be scaled down to accommodate an extremely low level of support, for example, just one full time or

perhaps a half time generalist. This approach provides a limited assurance that the system will operate properly on-orbit.

The most recent development that SID has recently begun to productize is the “Class Agnostic” mission assurance model. This model is not new but rather an outgrowth of mission assurance evolutions to date. This model is described in the main body of the TOR.

Appendix B. Other Applications and Illustrations of the Class Agnostic Concept

B.1 Early Class Agnostic Diagrams and Approaches

The initial Class Agnostic approach diagram in Figure B-1 was developed by Peter Chang and Andrew Read, and emphasizes the "knobs" of the mission and the need to adjust your risk management in accordance with your resources. It also recommends an up-front "blitz" of evaluation by subject-matter experts to help determine where risk mitigation efforts should be focused.

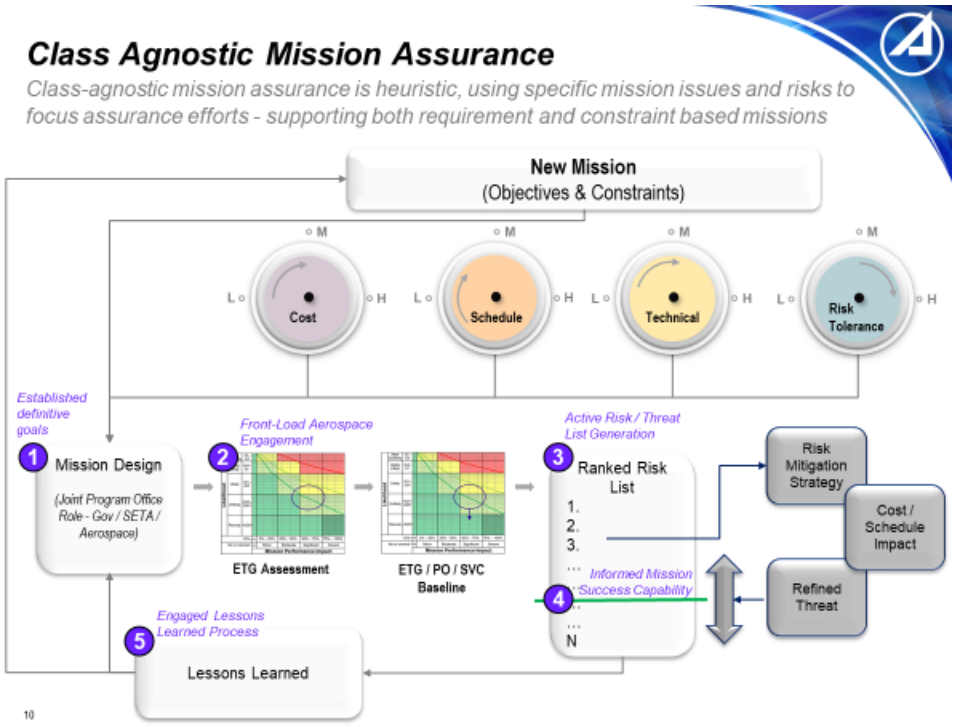


Figure B-1. Initial class agnostic approach diagram.

The initial presentation of this approach (see Figure B-2 and Figure B-3) also included a discussion of the applicability of Class Agnostic Mission Assurance to Continuous Production Agility.

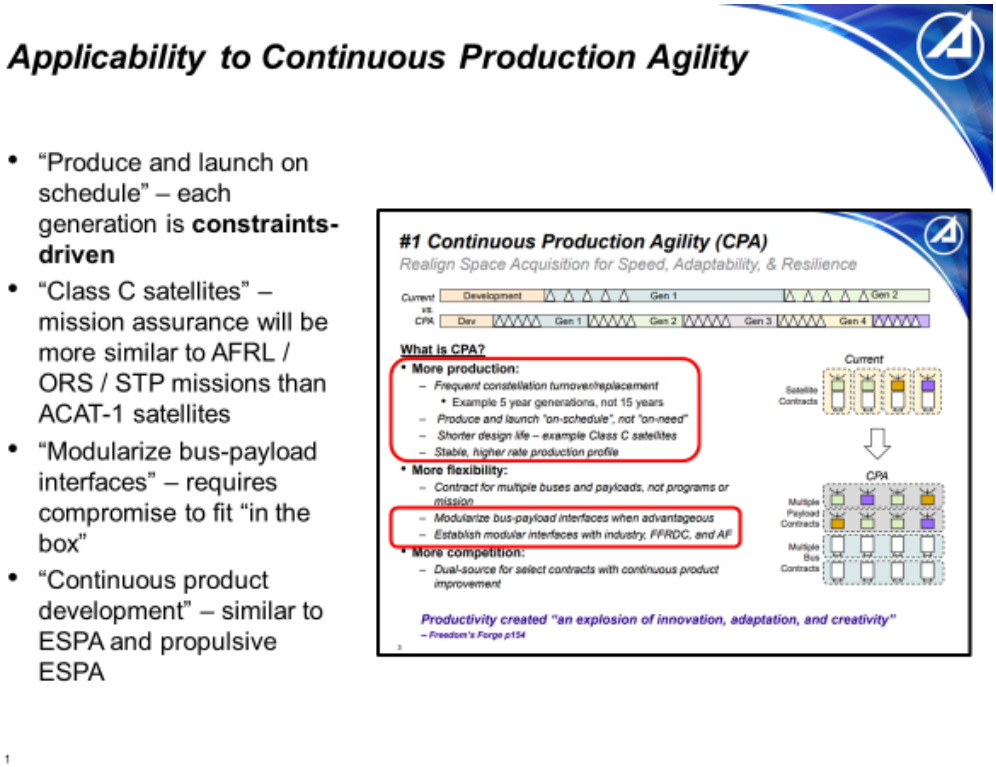


Figure B-2. Applicability to Continuous Production Agility.

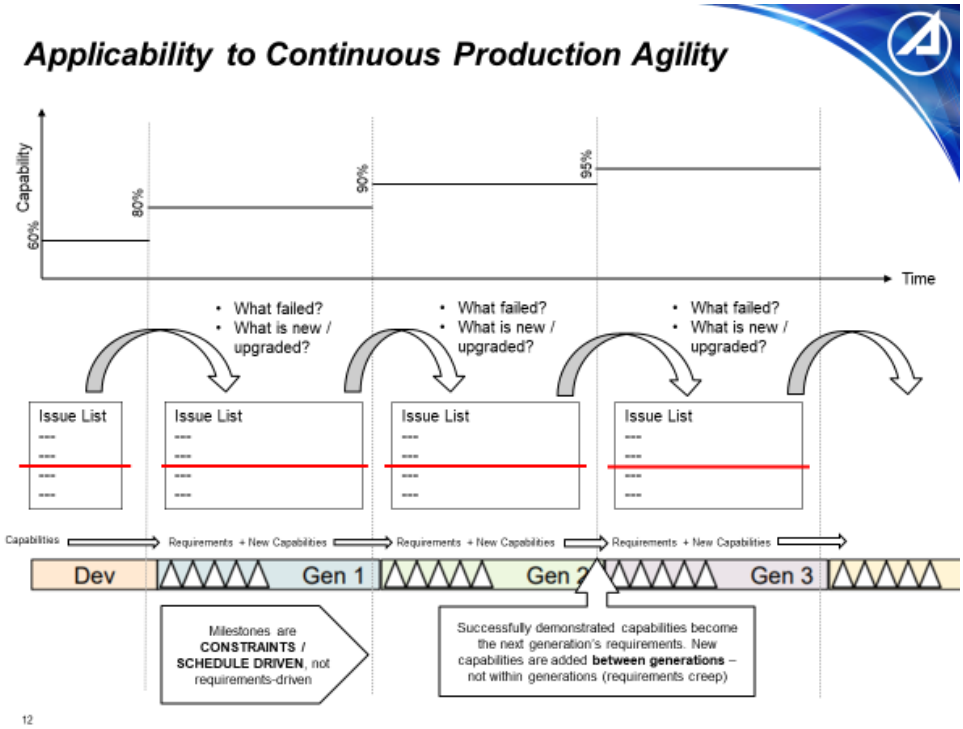


Figure B-3. Continuous Production Agility flow using class agnostic mission assurance.

Later evolutions of the same paradigm changed the diagram slightly (See Figure B-4) to streamline it and to more clearly label the steps.

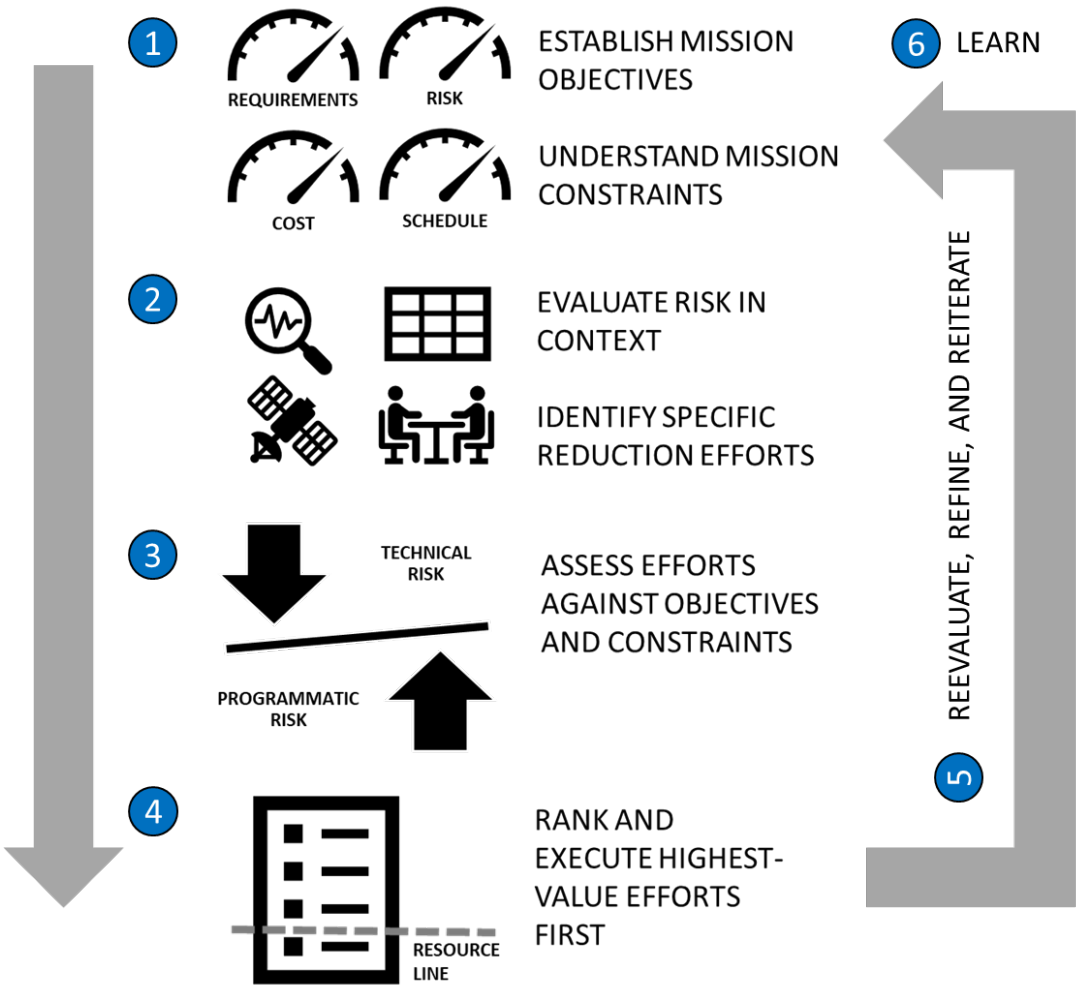


Figure B-4. "Square" diagram of class agnostic mission assurance (Jasper et al., 2020).

B.2 AFRL Program-Driven Approach

A version of the "square" diagram (See Figure B-5) appeared in a paper for the IEEE Aerospace Conference. (Jasper et al., 2020)

In this paper, the authors lay out the class agnostic concept against a typical small satellite program execution. This approach makes heavy use of peer reviews and the regular discussion of trades between requirements, risk, cost, and schedule, with higher-level programmatic reviews conducted at longer intervals and where "knobs" must be adjusted.

This paper shows how the overall construct can be adapted to a program's needs, with steps adjusted to fit the program execution cycle and the practices of the organization.

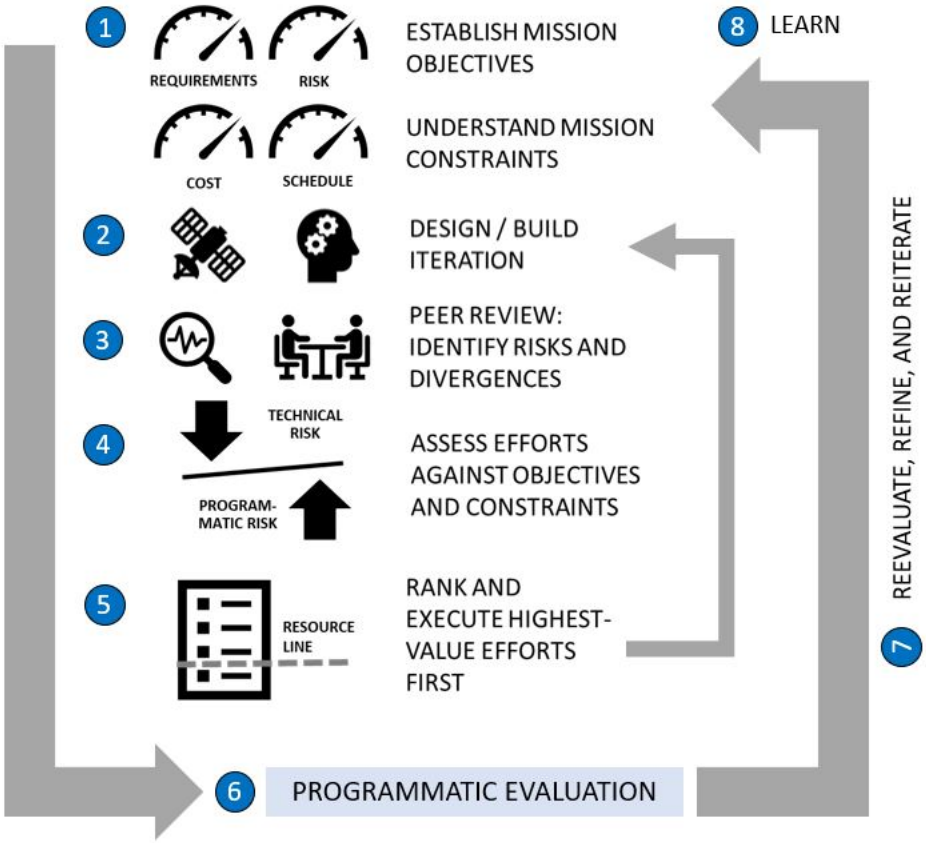


Figure B-5. "Square" diagram of class agnostic mission assurance (Jasper et al., 2020).

B.3 References

[1] Jasper, L., Braun, B., and Hunt, L. (2020, March 8-13). *New Constraint-Driven Mission Construct for Small Satellites and Constrained Missions* [Paper presentation]. IEEE Aerospace Conference, Big Sky, MT, USA.

AEROSPACE REPORT NO.
TOR-2021-00133

A Class Agnostic Mission Assurance Approach

Approved Electronically by:

Mark P. Jelonek, GENERAL MANAGER
STRATEGIC SPACE OPERATIONS
DEFENSE SYSTEMS GROUP

Jay G. Santee, VICE PRESIDENT
DEFENSE SYSTEMS GROUP
OFFICE OF EVP

Cognizant Program Manager Approval:

Kara A. O'Donnell, PRINCIPAL DIRECTOR
ADVANCED DEVELOPMENT & PLANNING DIVISION
STRATEGIC SPACE OPERATIONS
DEFENSE SYSTEMS GROUP

Aerospace Corporate Officer Approval:

Martin Whelan, SENIOR VP DEFENSE SYSTEMS GROUP
OFFICE OF EVP

Content Concurrence Provided Electronically by:

Douglas A. Harris, SENIOR PROJECT LEADER
STRATEGIC INITIATIVES AND SMALL LAUNCH
SPACE INNOVATION DIRECTORATE
DEFENSE SYSTEMS GROUP

© The Aerospace Corporation, 2021.

All trademarks, service marks, and trade names are the property of their respective owners.

SY0720

AEROSPACE REPORT NO.
TOR-2021-00133

A Class Agnostic Mission Assurance Approach

Technical Peer Review Performed by:

Kara A. O'Donnell, PRINCIPAL DIRECTOR
ADVANCED DEVELOPMENT & PLANNING DIVISION
STRATEGIC SPACE OPERATIONS
DEFENSE SYSTEMS GROUP

Eleni M. Sims, SENIOR PROJECT ENGINEER
AGILE SPACE ACQUISITION & IMPLEMENTATION
SPACE INNOVATION DIRECTORATE
DEFENSE SYSTEMS GROUP

© The Aerospace Corporation, 2021.

All trademarks, service marks, and trade names are the property of their respective owners.

SY0720