SSC21-XIII-09

How Satellites are Moving Beyond the Class System: Class Agnostic Development and Operations Approaches for Constraints-Driven Missions

Barbara Braun
The Aerospace Corporation
2155 Louisiana Blvd NE, Suite 5100. Albuquerque, NM 87110; (505) 314-6670
barbara.m.braun@aero.org

Lee Jasper
Space Dynamics Laboratory
3550 Aberdeen Ave. SE. Kirtland AFB, NM 87117; (435) 713-3400
lee.jasper@sdl.usu.edu

ABSTRACT

Should we abolish the Class System? The Class A/B/C/D mission assurance and risk posture designations familiar to most satellite developers were established in 1986. They are used by both the Department of Defense (DoD) and National Aeronautics and Space Administration (NASA) to define risk and risk mitigation requirements for flight missions. However, many of today's satellites are different – smaller, digitally engineered, designed for production, and increasingly destined for proliferated architectures. The rate of development is increasing while the uniqueness of the systems being built is decreasing.

The need to move faster and the ability to utilize, for the first time in space, real product-line components challenges the premise and assumptions behind the Class A through D designations. The traditional "Class System" is not as applicable to most small satellite developments, which instead focus on ways to prioritize key, high impact, agile processes in an effort to cut costs and timelines. Operating within this environment requires satellite developers to apply practices that are agnostic to class definition (e.g., the practices that are most fundamental to ensuring the mission meets the needs).

This paper outlines the Class Agnostic approach and constraints-based mission implementation practices. It will describe several real-life examples from Air Force Research Laboratory, Space and Missile System Center, and Space Rapid Capabilities Office missions that are applying a "class agnostic" approach to their missions. It will include lessons learned from missions which failed critical *Do No Harm* requirements and lost a flight to missions that have fully utilized the class agnostic approach. It will also discuss how the several missions used class-agnostic techniques to balance requirements of scope, risk, cost, and schedule to maximize the chances of mission success within hard constraints. The approaches used in these missions are applicable not only to small satellites, but also to any mission intending to move beyond the "Class System" to a more agile and flexible mindset for risk mitigation and mission assurance.

INTRODUCTION

Most satellite developers have encountered the "Class System," the approach to satellite mission assurance that divides missions into classes based on criticality or risk tolerance. The most commonly-used class system – which makes use of four mission assurance categories referred to as Class A, B, C, and D – was originally developed in 1986 by the Department of Defense for *one-of-a-kind space equipment*, and documented in a since-canceled handbook, MIL-HDBK-343[1]. In comparison, the first Small Satellite Conference would not be held until one year later, and in the 35 intervening years much has changed about space and space missions. The old A, B, C, and D mission assurance levels are increasingly difficult to apply to today's smaller constraints-driven missions, many of which are not one-of-a-kind.

The traditional "Class System" suffers additional shortcomings. All too often, these classifications are used as 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 lacks flexibility to the changing priorities encountered during program execution. Today, many missions instead focus on ways to prioritize high impact risk-reduction processes in an effort to cut costs and timelines and to get the most "bang for the buck" out of mission assurance efforts.

The Class A-D terminology can still be useful up front to help determine the applicability of redundancy and reliability requirements, testing approaches, contractual deliverables, and standards documents. And, for true requirements-driven missions the "Class System" may still be a good way to articulate risk posture and guide execution. In this paper, however, we outline a approach to mission assurance that breaks down the barriers of the "Class System," opting instead for a value-driven approach that is agile and changes with the program. In class-agnostic approach, "Time, Cost, & Quality are [all] commodities," and we "treat them as such. Quality is a commodity even if we don't think it is."[6]

KEY CONCEPTS

The class-agnostic mission assurance approach[14] is driven by several key concepts and relies on several assumptions about mission types. It may not be suitable for all missions. For example, large-team, requirements-driven launch missions which cannot fail may be more suited to traditional Class A mission assurance approaches. Other missions, ones that embrace the concepts and characteristics listed below, may benefit from a more class-agnostic approach.

Requirements-driven vs. Constraints-driven Missions

The distinction between requirements-driven and constraints-driven missions was first articulated as part of a paper at the 2018 Conference on Small Satellites[2], and later refined in an Institute of Electrical and Electronics Engineers (IEEE) paper[3].

A Requirements-Driven Mission is a mission where mission objectives / requirements drive the schedule and budget and where achieving those objectives is typically prioritized over schedule and budget. Requirements-driven missions are focused on mission system performance with less emphasis on how that drives budget and schedule.

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, and achieving mission requirements is the focus. Requirements-driven missions have at least some flexibility to delay a launch or add funding to ensure all mission requirements are met.

In contrast, constraints-driven missions are highly focused on achieving, within externally constrained budgets and schedule, "good enough" performance. A Constraints-Driven Mission is a mission where the objectives and/or scope are traded with, or bounded by, schedule and budget and all three may evolve as the system is defined, designed, tested, and operated[3].

Constraints-driven missions typically have little to no flexibility for delaying a launch or adding funding to resolve issues that arise. 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; if stakeholders wish to prioritize adherence to a standard

Figure 1: Constraints and Requirements-Driven Mission Assurance Spectrum

interface, they must allow developers the freedom to trade functionality or performance requirements if needed to fit that standard.

Figure 1 shows the concept of constraints-driven vs. requirements-driven mission architectures as another dimension to the typical Class A/B/C/D mission assurance levels. Missions that focus on the Class A/B/C/D categorization sometimes ignore this other driving factor.

The Agile Mindset and Approach

The class-agnostic framework is rooted in concepts borrowed from the Agile software development movement. 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."[4] 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**Error! Reference source not found.**.

Another way of stating the manifesto, similar to that described by Scott Ambler of the Agile Alliance, is as follows[5]:

- Mission assurance tools and processes are important, but it is more important to have competent people working together effectively.
- 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.
- Requirements lists and Contract Data Requirements Lists (CDRL) 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.

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: Agile Mission Assurance "Manifesto"

- 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.

Mission Attributes

The Agile concepts that underpin the class-agnostic approach 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 approach 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 to work 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, class-agnostic mission assurance relies on the following 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 support contractors, as well as the prime contractor and its subcontractors – makes up the mission assurance function. In most applications of the class-agnostic heuristic, mission assurance is the collective

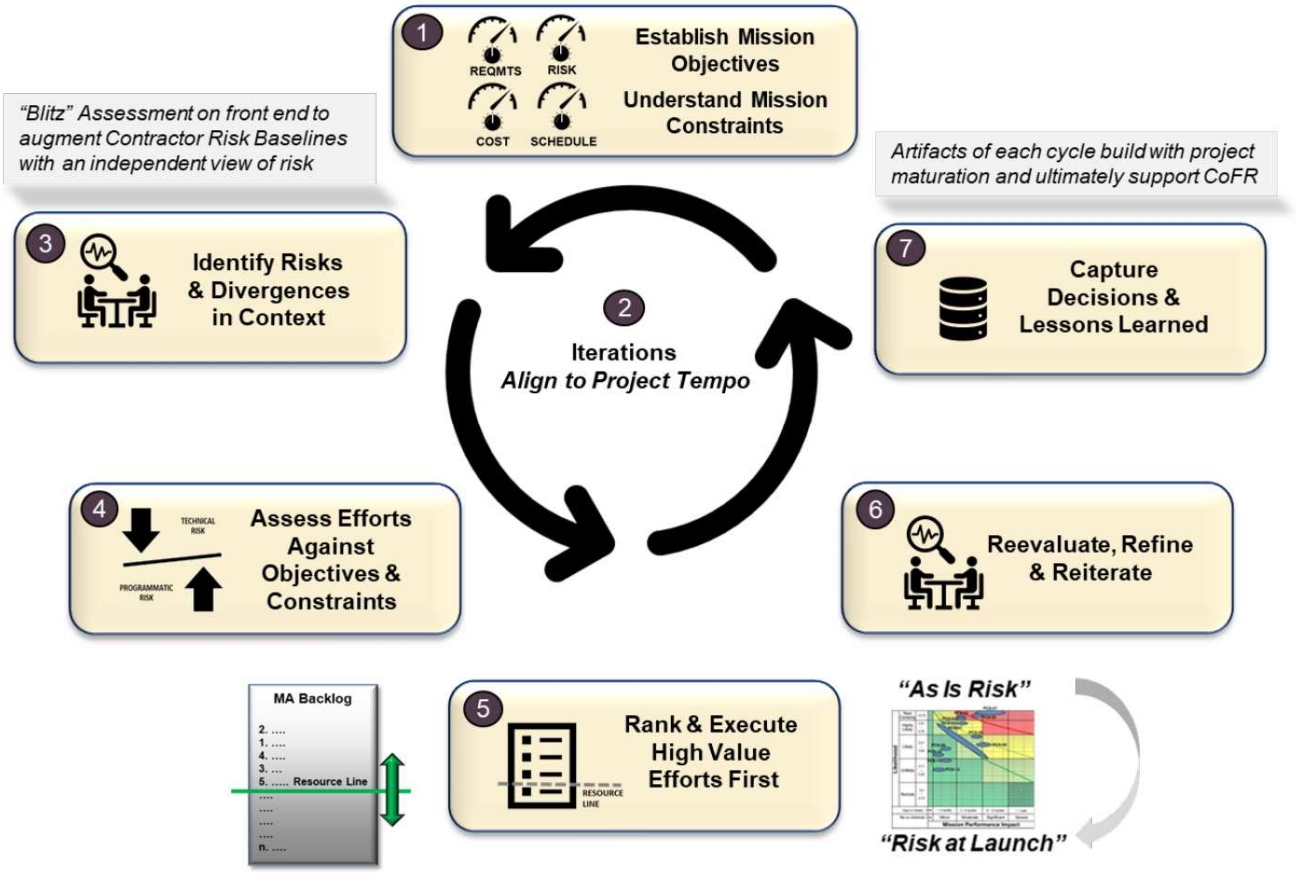

Figure 3: Class-Agnostic Mission Assurance Approach

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[2,6] and then add on what is needed and desired by the stakeholders to reach the final balance between capability and risk. Starting at a "Class A" approach and tailoring back is not only extremely cumbersome for resource-constrained missions. This approach also provides no fallback position should program realities change.

CLASS AGNOSTIC MISSION ASSURANCE

Figure 3 illustrates the overall approach to Class-Agnostic Mission Assurance. A more detailed discussion of the steps to this approach follows.

Step 1: Establish the "Knobs" of the Mission

The program team and stakeholders must understand mission goals and constraints and the relationship between them when applying the class-agnostic heuristic. The program team and stakeholders should not just develop an idea of what minimum mission success might look like – which may include ground-based objectives like maturing a technology or process – but also determine whether the scope of the mission matches this vision, and what level of risk the program is willing to accept, at least initially.

The first step is to explore the mission's objectives. The class-agnostic concept of objectives is not so much a requirement list as a story or a set of statements about what the mission is supposed to do. 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. 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?).

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. One of the major lessons learned from studies of CubeSats is to ensure that missions are scoped appropriately for the vision.[8] 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.

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. 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.

Developers and stakeholders should jointly understand the initial risk posture of the mission. 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.[9] However, few missions truly fit within a single 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. 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. Table 1 summarizes this taxonomy, and more details can be found in the referenced paper.[2]

Table 1: Risk Taxonomy for Space Missions

Demonstrated Level of Capability	Goals and Implications (Orbit)	Implication (Ground) + Risk at Launch
Do No Harm	DOA is ok (education and/or fully constrained and not requirement driven)	Hypothesis(es) untested; unclear capability
Survival	Not DOA (power + low-rate comm). May have no higher-level functionality	Hypothesis(es) untested; vehicle should be capable of supporting some operations and tests
Minimum Functionality	Minimum mission success. Mission recoverable in event of fault. No mishap / failure review board if achieved.	Hypothesis(es) partially tested; elements incomplete. Expect it to work but full evidence not created
Full Functionality (payload performance driven by constraints)	Full mission success. Functionality limited by constraints.	Hypothesis(es) tested on ground; should work. Need on-orbit data for further evidence
Class A/B/C/D (payload performance driven by requirements)	Full mission success, full functionality expected at set risk levels	Hypothesis(es) tested on ground; evidence of capability. Need on-orbit data for further evidence

This risk taxonomy provides several level-sets of capability that is demonstrated through verification activities and also provides implications for what that really means for the overall system's known capability. Here the idea of "Hypothesis" testing is added to help convey the premise. For example, if a system were to demonstrate a specific technology, the *hypothesis* being tested is both that that technology performs a certain way in space, and that the system is properly built to allow that technology to operate. Given this taxonomy,

stakeholders can make decisions about how much knowledge they'll have on the ground of the overall mission's ability to test that hypothesis. In the end, this is the real "risk acceptance" being made during the program.

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. Developers commonly encounter missions that are constraints-driven at kickoff, but requirements-driven at Launch Readiness Review.

The items described here (loosely categorized as requirements, risk, cost, schedule, or other constraints) represent "knobs" for mission execution (See Figure 3, Step 1). 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. 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.

When the team or stakeholders make a fixed designation of mission risk tolerance before they have any real knowledge of the technical issues, those designations are usually premature. Issues that arise during development, and the stakeholders' willingness to tolerate those risks, are generally opaque. Mission assurance approaches that rely on fixed assumptions are not flexible to the uncertainty in the mission.

Step 2: Align Iterations to Project Tempo

It is important for stakeholders 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. Peer reviews and major milestones can be tied to this cycle, or drive the cycle. Major milestones can become less about progress checks and more about decision-making with regard to the mission "knobs."

These iterations optimize mission assurance activity based on the risks to mission success objectives. It also burns down risk to an acceptable that is agreeable and well understood among stakeholders, and creates a more realistic expectation of mission success.

The first cycle will likely take longer than those following since more time is required for establishing a baseline of risks, divergences, and responses.

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. The orbit, lifetime, and intent of a mission provides the context within which to evaluate risk.

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.[10,**Error! Reference source not found.**,**Error! Reference source not found.**] Such processes are often designed for larger Class A missions, but provide a useful mental "checklist" for class-agnostic missions to ensure all aspects of the mission are at least considered. Areas to consider include:

- Generally-agreed-upon critical areas (e.g., power, communication, *Do No Harm* (DNH), 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)
- Lessons learned or "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.

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

- Set up specific analysis and test campaign for new developments or space readiness of terrestrial hardware

- 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
- 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

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.

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 divergences from the current scope or constraints.

Step 4: Assess Efforts against Objectives and Constraints

Once the team has the initial or most recent iteration of the risk list, it can estimate the level of each risk and the amount of risk handling possible, as well as the effort required to conduct each handling action beyond simple acceptance of the risk. The amount of risk reduction expected for each handling 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 and key for communication with stakeholders. This estimation allows the mission team to explain the tradeoffs of risk reduction, and why some risks might be mitigated while others are simply accepted (or watched).

There are several frameworks for estimating risk. The standard 5x5 risk matrix is familiar to most system developers. It provides a useful tool for identifying risk levels by likelihood and consequence, but it has pitfalls. 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 non-production or limited batch systems, 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. Another pitfall is the tendency for risks to span categories across the 5x5 risk matrix. 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. Except for very clear-cut, quantitative risks, teams should consider the numerical values in the risk matrix as guidelines.

Some programs, especially smaller, constrained programs, might consider a simplified 2x2 risk matrix instead, as shown in Figure 4. While this might seem oversimplified, it helps programs quickly categorize major risks and focus their efforts on handling them. Teams may also consider using other Agile methods like "planning poker" to estimate risk. 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[4].

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. Teams will also need to estimate how much effort is required to conduct each proposed risk handling action. 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. Risk reduction must take place in the context of the effort required, especially for resource-constrained missions. Small teams cannot do everything and must decide where to focus their effort to have the best chance for success in both mitigations actions and in overall mission performance.

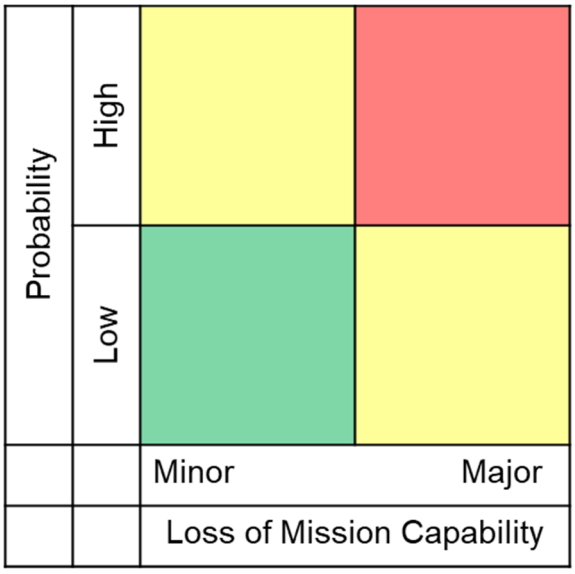

Figure 4: Simplified Risk Matrix

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.

Overall, 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."

Step 5: Rank and Execute High Value Efforts First

Once risks and actions have been defined, and the effort to accomplish each has been 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 stakeholders 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.

One way to visualize this would be in a table of risk reduction efforts. The table lists the estimated amount of risk reduction, the confidence in that estimate, the estimated cost or effort required, and the confidence in that estimate. Figure 5 is a notional example. Missions could use similar charts to show how their reduction efforts reduce their risk to match their overall risk posture. The charts can also show 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 or technical risk matrix as shown in Figure 6. Missions can

Risk	Action	Risk Reduction (story points)	Confidence in Risk Reduction	Cost (story points)	Confidence in Cost	Ratio	Implement?
Software issues delay test schedule	Add software review	5	High	3	Medium	1.7	Yes
	Add extra time for testing	8	High	13	High	0.6	No
Finite element model not correlated, leads to structural failure	Add modal survey test	20	Medium	20	High	1.0	No
	Add extra accelerometers to sine vibe test	3	Medium	2	High	1.5	Yes

Figure 5: Table of Risk Reduction Efforts

then determine which actions to prioritize based on where they fall along green / yellow / red boundaries the team has drawn itself.

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 upon and provide the best mission assurance value(s).

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. 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!

Step 6: Reevaluate, Refine & 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

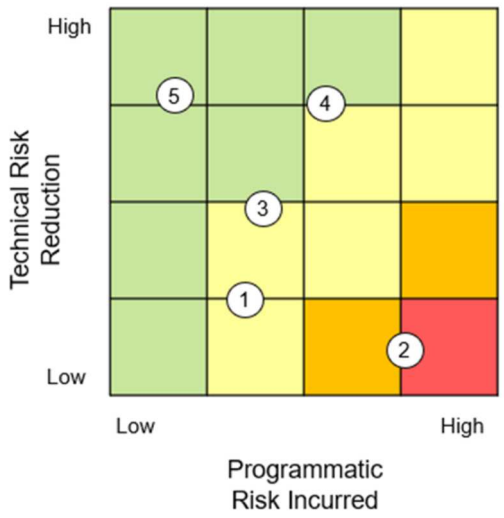

Figure 6: Technical vs. Programmatic Risk

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. It is worth noting that the team may have margin built into their plan and this can be used to absorb smaller changes before larger "knob" adjustments are required; the level of margin and control over that margin helps dictate how much capability the team has to execute on its own.

If enough issues have arisen that one of the "knobs" needs to be adjusted, the 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.

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 decisions made. 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 generally 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.

REAL MISSION EXAMPLES OF CLASS AGNOSTIC, CONSTRAINT-BASED SYSTEMS

While the class-agnostic concept has been used informally over the last number of years, there hasn't been a formalized assessment for how these practices actually apply. This section illustrates elements of the class-agnostic mission assurance approach across multiple real flight missions. The discussion abstracts many details of the missions themselves but focuses on one or several exemplar scenarios in the missions' developments that were enhanced by the class agnostic approach and references instances where one of the Steps was utilized. Table 2**Error! Reference source not found.** provides an overview of the missions discussed, their attributes, how they fit within the class-agnostic approach by reference, and the takeaways concerning implementation of the approach..

Mission A

Mission A[13] took, effectively, two nearly separate attempts to progress to flight. The mission was a concurrent measurement science mission that fizzled partly due to instrument and (early) CubeSat technology challenges but also due to a poor establishment of stakeholder interaction. With the delay of the partner mission, an assessment of the efficacy of the overall mission occurred; the mission was found to still add scientific value.

The revamped mission quickly set up the environment for the design team to re-work, design, build, and test the entire flight system. The rework was bounded by known, achievable, mission objectives that the team could easily understand how changes they made did, or did not, affect mission capability (step 1, 3, 4). While this required notable engineering resources (staff and funding, step 1), the team's overall authority to make decisions to progress quickly (schedule constraint), while maintaining a reasonably tight communication loop with stakeholders (few formal reviews, mostly peer and stakeholder discussions throughout; step 2, 6, 7) enabled build-to-launch to be about 18 months. The risk taxonomy flexed and was reduced downwards in the program due to both overall scoping reduction and timeline requirements to meet launch (see Table 2).

Mission B

Mission B was a technology/capability demonstration that purposely had large flexibility in its scope to help push for "art of the possible". Unfortunately, this flexibility came with poorly established stakeholder interactions and poorly defined mission objectives to properly allow the team to balance risk impacts on capability (step 1, 2, 4). This had the effect of the mission being defined by *delivered* capability.

As time began to run out, cost, schedule (launch), and technical capability (staff) all became limited, therefore, driving three constraints at the same time (step 3). Multiple de-scopes were allowed in performance and functionality; focus on "what matters" was hard to define (step 4, 5). Over time, the taxonomy in Table 1 was heavily used to re-evaluate the program and defined both final design attributes as well as verification/risk level (note that the taxonomy should not be used to define design attributes, only risk/verification levels). By the end, a *Do No Harm* violation was discovered and the lack of remaining budget or schedule to find root cause led to de-manifestation of the mission (steps 6, 7). Key challenges on this program resulted from the lack of experienced management, consistent loss of personnel, and poorly defined objectives and constraints making it unclear which "knobs" existed for the team (step 7, 1).

Mission C

Mission C was a technology/capability demonstration. Technology developments were prioritized early on over mission development. This led to some "cart before the horse" issues for vendor specifications and definition of the needs of the overall mission (step 1, 3). Further, iteration loops and assessments as well as stakeholder interactions were not well established at the beginning (step 2). Multiple technology risk reduction decisions were made (acceptable for the program). Programmatically, the team identified a lack of correct staffing or skillset availability (step 3) which was only addressed late in the program with a notable staffing increase (step 6, 1). Towards the end of development, delivery delays of both vehicle and payload, along with launch schedule deadlines, required the team to utilize the risk taxonomy as a framework to define a test campaign that was sufficient for stakeholders (minimum

success) while meeting the hard schedule constraint of launch (step 4, 5).

Descoping was the most common outcome when issues occurred on this program; however, a launch slip allowed the team to expand their test campaign to include more edge case testing, buy down more risk with the ground segment and flight hardware, and mitigate several late-breaking risks. This yielded greater functionality and even helped reduce known capability gaps that had been "accepted" prior to the slip (step 6, 7, taxonomy).taxonomy) It shows the agile adjustment of the class agnostic approach.

In the end, many of the issues experienced by Mission C could have been reduced by having a more experienced team, or better mentoring and coaching (say through peer review). Multiple unnecessary rework and scope reductions occurred due to these issues and the mission would not have been recoverable without a late staffing increase.

Mission D

Mission D is the first mission to start with the class-agnostic/constraint-based mission approach and was another technology/capability demonstration. As a cost-constrained mission, it became apparent that a high functioning team was required to stay within limitations without too many design iterations. Multiple early reviews and discussions were held with stakeholders to establish “knobs,” acquisition strategy, etc. Eventually key risk ranking and mitigation options were discussed and prioritized. This emphasizes the need for a well scoped and well-established program/execution plan early on (steps 1 – 3). The largest design review was only a preliminary design review (PDR) and stakeholder meetings are generally only held two times a year for updates and decisions (step 2, 6, 7). Risk assessment has been one of the key tools used by the team to convey areas for investment (time or funds) and to focus conversations of “this will or will not affect success, so does it matter?” (step 3, 4, 5).

Over time, and thanks to the team’s efficiency, the program has expanded its risk taxonomy level beyond minimum functionality, allowing for better system characterization. Cost and scope are now the primary drivers, instead of just cost, since the stakeholders have gained high confidence in the team and the established communication structure. The class-agnostic approach built trust and yielded a highly effective communication approach that has enabled the team to own much of the decision-making and expand capability when the opportunity arose while ensuring stakeholders had the correct buy-in and buy-off to major risks and trades.

Mission E

Mission E is a capability demonstration and had numerous, known, technical hurdles to overcome before a space test could be performed. This program started with risk mitigation activities to ensure a space demonstration was possible. With this effort, the overall scope and objectives became very clear and have been very static throughout the development process (steps 3, 4, 5, 7, 1). Again, well-defined scope (e.g.,. defining “what matters”) has greatly aided execution of this mission and enabled a small team at the program / system engineering level. Generally, stakeholder interactions were minimal and/or low effort but frequent, reducing overall communication burden (step 2). Further, significant portions of the mission have been completed by subject matter experts (in-house or contractor) who were brought into the project early creating much more effective risk identification, evaluation, and iteration within the well understood scope of the mission (steps 3-6).

Multiple technical risks emerged including poor intra-team communication and shifts to under-developed products, causing related programmatic risks. The class-agnostic approach was not well utilized in addressing these risks and multiple trades/discussions were fairly drawn out or somewhat ambiguously concluded. Further, the class-agnostic approach still does not have a particularly effective counter to realized risks (issues/problems) which have been an issue for the Mission E. However, the approach does encourage earlier identification of risks and prioritization, as well as a framework for handling how to address realized risks (changes to knobs, shifting in the taxonomy). For Mission E, this has resulted in schedule elongation (a constraint) to trade for higher quality deliverables.

Mission F

Mission F was a half-ESPA class satellite with five experimental payloads and a relatively low budget (<$50M). It was part of a commercial rideshare with a fixed launched date. The mission had an initial risk posture of Class D and a mission assurance approach that involved a small team of four generalists embedded in the program.

The program encountered issues during integration, primarily related to software, that held up environmental testing. Cost overruns led to severely limited funds. The team employed class-agnostic concepts, allocating resources to software efforts necessary for survival and minimal functionality (steps 4 and 5). The cost and schedule “knobs” were fixed; the risk and requirements knobs were adjusted (step 6). On orbit, the mission

scaled back objectives to prioritize the collection of science data. The team worked around data issues.

Critical to the implementation of the class-agnostic approach was the recognition that the cost and schedule slips were also the greatest technical risk (step 1); if the satellite missed its launch, it would be unlikely that the five payloads would ever be re-manifested. The satellite had to meet *Do No Harm* and safety requirements, but despite the Class D designation the real driver of mission assurance efforts (beyond Do No Harm) was the highest bang for the buck.

Mission G

Mission G is a large Class C/D mission with Class A elements. It includes two operational demonstrations and several smaller experimental payloads. While the overall risk acceptance level of the mission was listed at Class C/D, the mission – a long-lifetime satellite designed for geosynchronous orbit – required much greater mission assurance than typically implied by a Class C/D designation. The approach followed on this mission was more a traditional / tailored approach, but as issues arose, class-agnostic concepts were used to determine how best to adjust the mission's "knobs." Since requirements and risk posture were more fixed than usual in a Class C/D mission, the schedule was lengthened to accommodate the resolution of issues (steps 1 and 6). Initially funding was also increased, but toward the end of integration a greater risk posture was accepted in order to cap overruns at an acceptable level. Cost pressures required smart decision-making in order to prioritize the greatest "bang for the buck," and class-agnostic concepts were used to drive decision-making on risk.

This mission demonstrates several limitations of the traditional "class system" approach – missions rarely fit neatly into a single category, and Class C or D category designations are sometimes chosen more to reflect the optimistic hope of cost savings than the real risk tolerance of stakeholders. The class-agnostic system is more transparent regarding the decision-making required to achieve the mission: in this case, cost and schedule were flexible enough to maintain requirements up to a point, and after that point, the cost "knob" was fixed, and risk posture increased.

Mission H

Mission H is a more traditional operational mission, a larger high-value asset with a low risk tolerance. However, the mission is being executed by a small team on a compressed schedule, and very few resources are available for independent mission assurance (step 1). The mission is in its early stages, but the class-agnostic approach is being used to drive mission assurance efforts.

One class-agnostic concept used in this mission is the up-front mission assurance "blitz," (step1) where a (comparatively) large set of subject-matter experts are reviewing the mission's preliminary requirements flowdown and design in order to determine where best to apply risk-reduction efforts. Peer review and focused analysis on the areas of greatest risk have helped drive early decision-making. Major milestone reviews are being conducted in a more Agile, iterative process, where traditional system requirements review (SRR), preliminary design review (PDR), and critical design review (CDR)SRR, PDR, and CDR meetings are as short capstone events to a longer informal subject-matter expert review period (step 2). The mission is constrained by schedule and staffing, but seeks to maintain a low risk posture; the scope, requirements, and budget "knobs" may end up being the primary method by which stakeholders handle issues. Already, however, the schedule is slipping, mainly due to funding availability; in the end, the schedule may also need to be flexible.

The application of class-agnostic mission assurance to this mission is unusual due to its large size, budget, and mission-critical nature. It remains to be seen if the approach will enable the mission to make hard choices between cost, schedule, performance, and risk, or if the class-agnostic approach will become harder to execute as the mission moves on and the natural tendency for risk tolerance to decrease kicks in.

Mission J

Mission J was a demonstration mission carrying five experimental payloads and designed to be deployed through the International Space Station (ISS) airlock. As such, it was a lower-budget, small-team mission limited primarily by form factor and staffing, but also by cost. Schedule was slightly more flexible given the ability to re-manifest to a later commercial resupply mission if needed. The initial, and final, mission designation was minimal functionality; however, the actual success criteria for the mission were not formally documented (step 1 and the risk taxonomy).

The mission prioritized functional testing and successfully uncovered and fixed several issues before launch. The mission was successful in collecting data on orbit; however, a failure in a key component led to mission loss after about four months of data collection. "Do No Harm" was preserved – the mission will re-enter approximately two years after deployment, and will not pose a safety hazard to the ISS. While the foreshortened mission was sufficient to meet minimum functionality, it led to issues with the mishap prevention community, who conducted a full failure investigation – perhaps overkill for a low-budget, experimental, single-string mission. This mission illustrates the benefit of defining

clear success criteria up front and interacting with stakeholders (including non-obvious stakeholders such as those responsible for identifying and reporting mishaps) to ensure that the risk posture is fully understood (step 1, and the definition of minimum success criteria). The taxonomy in Table 1, and the class-agnostic approach, may provide a way for missions to better document the true objectives of a mission, and the true risk posture being accepted, to ensure that expectations are clear across all stakeholders.

Mission K

Mission K was executed several years before the first formal documentation of class-agnostic mission assurance. Nevertheless, it helped define the approach and provides several instructive examples. The mission was larger than ESPA-class, with a relatively large budget, but a highly-constrained schedule. Early in the mission, the mission priorities were established: schedule (launching on time with minimum functionality) was considered the highest priority, with cost as the second-highest priority, and performance as the third priority (step 1).

The mission used an airborne payload minimally modified for space applications, and the bus was based on a previously-flown satellite. During execution, the stated priorities (schedule first, then cost, then performance) were not always followed; repeated inquiries about adding performance or incorporating additional capabilities led to cost growth. The team learned to keep the primary goal in mind, and strictly limit investigations into possible enhancements until that mission had been achieved. The team also learned to formally document all decisions – as leadership changed throughout the program, previous decisions (especially hard ones) were continuously revisited, leading to wasted time (step 7).

Despite this, the mission ultimately accepted many risks instead of mitigating them, keeping the schedule "knob" relatively fixed and accepting risk against performance when issues were uncovered during testing. Having a small, high-performing team helped; once roles were established and bureaucracy was minimized, the team wasted little time with unnecessary documentation. The mission was successful on orbit, but required significant courage (for lack of a better term) to launch: the risk acceptance level at Flight Readiness Review was high. Fortunately, issues encountered on-orbit were not mission-threatening, likely due to the focus on critical activities and a strong "test like you fly" mentality.

Table 2: Summary of Missions

	Mission Attributes	Vehicle / System Config	Driving Constraint	Spectrum of MA (Figure 1)	Taxonomy Baselined (Table 1)	Taxonomy Implemented (Table 1)	Key Takeaways
Mission A	Late application of the approach; one of key missions to help formalize this approach. Schedule constrained system. High risk approach for new technology	CubeSat 6U	Schedule	Constraints Driven / High Risk Tolerance	Full Functionality	Minimum Functionality	- Development team had to make quick course corrections (sometimes sweeping) - Team prioritized functionality and testing over large trade studies / analysis - Enabled a revamped mission to build and launch in ~18 months (after several years of stagnation)
Mission B	Late application of the approach; schedule and staff constrained system. High risk approach for new technology	Half ESPA	Schedule and Staff	Constraints Driven / High Risk Tolerance	Full Functionality	Do No Harm	- Taxonomy and agile approach enabled mission team and stakeholders to discuss issues, impacts to scope, and agree on immutable requirements
Mission C	Late application of the approach; overall mission developed on CubeSat-like approach with limited staff and high risk approach for new technology	CubeSat 12U	Staff->Schedule	Constraints Driven / High Risk Tolerance	Full Functionality	Minimum Functionality / Full Functionality	- Agile implementation can suffer from continually reducing scope, creating unnecessary rework - Experience and/or good mentorship are required
Mission D	First mission to go through the full approach purposefully; CubeSat-like approach with limited funding but evolutionary hardware/software from previous missions. Medium risk tolerance for new technology and mission application	CubeSat 6U - 12U	Cost	Constraints Driven / Low(er) Risk Tolerance	Minimum Functionality	Minimum Functionality / Full Functionality	- Class-agnostic approach provided a communication tool with stakeholders, and within the development team, to define risks and trades - Active "knob adjustments" allowed team to start by "promising less and delivering more"
Mission E	Late application of the approach; overall mission developed on CubeSat-like approach. Medium risk tolerance for new technology and mission application	CubeSat 12U	Staff	Constraints Driven / Low(er) Risk Tolerance	Full Functionality	TBD - Expect Full Functionality	- Demonstrated key practices of properly scoping the mission and addressing high value efforts first - Focused time and funds on early risk mitigation activities leading to successful implementation. - Constraints can be variable and is one of the adjustable "knobs"

Mission F	Helped refine the approach; initial mission developed under Class D mission assurance practices, but used class-agnostic approach as funding and schedule became tight. High risk tolerance for new technology and mission application	Half-ESPA	Schedule and Cost	Constraints Driven / High Risk Tolerance	Full Functionality	Minimum Functionality / Full Functionality	- Helped refine the approach - Focused efforts on critical-for-launch issues and accepted scope changes and risk - Experienced on-orbit anomalies in high-risk areas but completed mission
Mission G	Class C/D mission with Class A elements; two operational demonstrations and multiple experimental payloads. Class-agnostic concepts used to focus effort in critical areas toward end of development	Large	Requirements	Requirements Driven / High(er) Risk Tolerance	Full Functionality	Full Functionality	- Missions are rarely monolithic and single-class - Important to "pick your battles" when budget is fixed
Mission H	Mission still in early development but using the approach purposefully. Operational mission with critical implications, but heavily constrained by resources. Low risk tolerance, desire for full functionality	Larger than ESPA	Staff	Requirements Driven / Low Risk Tolerance	Full Functionality	TBD	- An up-front full mission assurance "survey" can help identify critical areas early - Mindful trades can reduce risk - Harder decisions are coming
Mission J	Demonstration mission carrying multiple payloads. CubeSat-based approach in larger form factor designed for deployment from ISS. Completed on-orbit mission but failed before end of design life	ISS Airlock Deployable	Form Factor	Constraints Driven / High Risk Tolerance	Minimum Functionality	Minimum Functionality	- Focus on high-value testing uncovered issues early - Would have benefited from better definition of minimum success criteria
Mission K	Operational but heavily constraints-driven (schedule). Larger budget than most, higher risk tolerance. Successful on orbit	Larger than ESPA	Schedule	Constraints Driven / Low(er) Risk Tolerance	Full Functionality	Full Functionality	- Good prioritization of requirements (schedule, then cost, then requirements) but prioritization not always followed - Good example of deliberative risk decision-making - On-orbit issues were in lower-criticality areas - Communication with stakeholders and documentation of decision-making is critical

CONCLUSIONS AND FUTURE WORK

The class-agnostic mission assurance approach, like any Agile process, is aa dynamic, living process. It is intentionally not prescriptive and should adjust to the needs of the program, the needs of the organization, and the needs of stakeholders. So much of the class-agnostic approach really is about finding and assessing **what matters** for mission success, creating and maintaining a communication structure to convey this to stakeholders, and making timely decisions.

The class-agnostic approach also relies heavily on experience. 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. Young engineers should be paired with experienced engineers to help them develop good instincts.

The program team (program managers, mission assurance, engineering team) should continuously evaluate what works for them and what improvement is needed. Similarly, this discussion and assessment should occur with stakeholders to get buy-in and cognizance of the flow of the approach. In the end, this is a messy approach and will be naturally tailored throughout a program (instead of purely up front). The Agile mindset encourages re-evaluating and discarding things that do not work. *This takes experience, mentoring, and practice to get right*.

In an effort to figure out what matters, the class-agnostic approach is also continuously being evaluated as it is applied to programs. This has yielded changes and understanding of the useful elements (presented here) but also elicits further questions. Future work may include further study and comparison to how Agile, a more software-based process, merges with the software/hardware development of space and ground systems (perhaps considering examples like the Toyota Production System). Also the merging of traditional Class A-D approaches with the class-agnostic system is mostly unexplored but could be correct for higher reliability non-unique systems; in the end when missions are constraints-driven, and mission risk posture is just one more requirement that can be traded, the Class Agnostic heuristic can apply.

ACKNOWLEDGEMENTS

The Air Force Research Laboratory's Small Satellite Portfolio (SSP) continues to lead the exploration into constraint-based mission practices in an effort to better balance speed of development, cost, and scope. Much of the experience gained to create this paper has come from its missions and through the University Nanosatellite Program. AFRL's SSP team has been integral in conceiving and creating these concepts.

The authors would also like to thank the Aerospace Corporation's Space Innovation Directorate and all the individuals who have supported the DoD Space Test Program and the Space Rapid Capabilities Office over the years, who helped create the class-agnostic approach. Special thanks go to Peter Chang and Andrew Read for developing the early concepts, to Doug Harris for co-authoring the Aerospace Technical Report, and to Mark Jelonek, Gayla Walden, and Kara O'Donnell for their review and feedback.

REFERENCES

1. "Design, Construction, and Testing Requirements for one of a kind space equipment," SPVT-2016-005, ORIGINAL ED., DOD-HDBK-343. February 1986.
2. Jasper, L. E. Z. and Hunt, L. and Voss, D. and Jacka, C., "Defining a New Mission Assurance Philosophy for Small Satellites," SmallSat Conference, Logan, UT, Aug 4-9, 2018. Paper No. SSC18-WKII-05
3. Jasper, L. E. Z. and Braun, B. and L. Hunt, "New Constraint-Driven Mission Construct for Small Satellites and Constrained Missions," IEEE Aerospace Conference, Big Sky, MT, Mar 7 – 14, 2020. Paper No. 2.0409.
4. Agile Alliance. (n.d.), "Agile 101 - What is Agile Software Development?" Retrieved October 13, 2020, from https://www.agilealliance.org/agile101/Ambler, n.d.
5. Ambler, Scott. (n.d.), "Examining the Agile Manifesto," Retrieved October 13, 2020, from http://www.ambysoft.com/essays/agileManifesto.html.
6. Read, A. and Chang, P. and Braun, B. and Voelkel, D, "Rideshare Mission Assurance and the Do No Harm Process," Report No. TOR-2016-02946, The Aerospace Corporation, 2016.
7. Cavender, D. "Emerging Low Toxicity "Green" Chemical Propulsion Technologies for SmallSats," Monthly Webinar Series, Small Spacecraft Virtual Institute, Marshall Space Flight Center, 16 September 2020.
8. Venturini, C. "Improving Mission Success of CubeSats," Report No. TOR-2017-01689, The Aerospace Corporation, 2017.

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

10. ISO/TC 20/SC 14 Space systems and operations, Space Systems Risk Management, ISO 17666:2016, International Organization for Standardization, 2016.

11. Office of the Deputy Assistant Secretary of Defense for Systems Engineering, Risk, Issue, and Opportunity Management Guide for Defense Acquisition Programs, Defense Acquisition Programs, Department of Defense, Washington, D.C., 2017.

12. Office of Safety and Mission Assurance, Agency Risk Management Procedural Requirements, NPR 8000.4B, NASA, 2017.

13. Willett Gies, T. and Shirley, B. and Jasper, L. and Enger, C., “Very Low Frequency Propagation Mapper (VPM) On Orbit Operations: Results and Experiments with Modernizing Operations,” SmallSat Conference, Paper No. SSC20-II-04, Logan, UT, Aug 1-6, 2020.

14. Braun, Barbara M. and Berenberg, Lisa A. and Herrin, Sabrina L. and Musani, Riaz S. and Harris, Douglas A., “A Class Agnostic Mission Assurance Approach,” Report No. TOR-2021-00133, The Aerospace Corporation, 2021.