CENTER FOR SPACE POLICY AND STRATEGY

OCTOBER 2020

Principled Design vs. Designing for Principles: Rethinking Capability Development for the Space Enterprise

ERIN T. RYAN
THE AEROSPACE CORPORATION

AEROSPACE

ERIN RYAN

Erin Ryan is a senior project engineer in The Aerospace Corporation's Defense Systems Group. For over two decades, he has been involved with various aspects of DOD space capability development, including architecting, engineering and design, acquisitions, and operations. He holds a Ph.D. in systems engineering from the Air Force Institute of Technology, where he first became passionate about the need for design flexibility.

ABOUT THE CENTER FOR SPACE POLICY AND STRATEGY

The Center for Space Policy and Strategy is dedicated to shaping the future by providing nonpartisan research and strategic analysis to decisionmakers. The center is part of The Aerospace Corporation, a nonprofit organization that advises the government on complex space enterprise and systems engineering problems.

Contact us at www.aerospace.org/policy or policy@aero.org

Summary

United States national security is highly dependent on DOD space capabilities. It is broadly acknowledged that these capabilities are susceptible to adversary attack due to an intrinsically fragile architecture that has long assumed a relatively benign space environment. This fragility has not been a problem historically, but with the expanding number and types of threats to U.S. space systems, it now represents a significant Achilles' heel to America's national security.

To assure that critical space capabilities will be available when needed, the U.S. must develop a more resilient space architecture. But given the increasing complexity of the DOD space enterprise, as well as the dynamic and uncertain threat environment, traditional development and design methodologies may not be up to the task. In order to produce a truly resilient space architecture, a more nimble methodology is required.

Fortunately, the basis of such a nimble methodology already exists, one that emphasizes particular design principles, like flexibility, adaptability, and survivability. It is known as Design for Changeability (DfC) and is inherently better suited to accommodate complexity and more effectively respond to uncertain and changing environments like what we see in the space domain today. This paper proposes maturing and extending the core DfC concepts into a capability development framework referred to as "Designing for Principles" and applying this framework to the development of DOD space capabilities in order to support rapidly achieving and sustaining a resilient space enterprise architecture.

Introduction

> ***It's a poor sort of memory that only works backwards.***
>
> —Lewis Carroll
> *Through the Looking Glass*

This paper describes a novel concept for capability development called Designing for Principles, which is presented as an alternative to the traditional systems engineering (SE) approach. Although SE has proven to be a remarkably effective capability development methodology, it does have some inherent weaknesses. This is certainly the case given how SE is typically implemented in the Department of Defense (DOD)[1] and is especially the case when it must confront highly complex, uncertain circumstances. For a volatile and dynamic

environment like today's space domain, the arguments to adopt an alternative capability development approach are particularly compelling.

The DfP concept builds upon a design idea first introduced in the 2005 timeframe. This idea was known by a few names, the two most prominent being Design for Changeability[2] and Design for Adaptability.[3] The first term (i.e., Design for Changeability, or DfC) is somewhat more common in the literature, so that's what is used here as a collective proxy for all related terminology.

The first part of this paper provides an overview of DfC. The premise of DfC is that, at a certain level of uncertainty, it makes more sense to exchange the rigorous Principled Design approach of SE for one that values certain design principles, such as flexibility and adaptability. In other words, in highly complex and unpredictable environments, it is generally a superior long-term strategy to develop capabilities that are inherently better at effectively responding to future changes than to keep trying to develop capabilities based on today's best guess.

This paper then extends and generalizes the original DfC idea into a more robust capability development framework designated as "Designing for Principles (DfP)." In contrast to SE, DfP emphasizes certain design principles over traditional measures of technical mission performance. Critically, DfP—like its DfC antecedent—recognizes that genuine resiliency does not come from picking one optimal future design now; rather, it comes from the ability to continually adapt designs to a range of future needs and threats that cannot possibly be known and/or characterized today.

To keep the scope of this paper manageable, the discussion will *not* include how to actually implement DfP. Instead, that topic will be addressed in a follow-on paper, which will provide specific, actionable recommendations for implementing DfP across the DOD space enterprise.

The intended audience for both of these papers is primarily enterprise architects and systems-of-systems engineers supporting the DOD space enterprise. However, the implications of what is being proposed here are broad and deep, affecting everything from how space capabilities are acquired to how the U.S. Space Force is structured. As such, if DfP is pursued, it will have significant ramifications to development contractors, government program office personnel, service headquarters staff, and senior decisionmakers across the space community.

…genuine resiliency does not come from picking one optimal future design now; rather, it comes from the ability to continually adapt designs to a range of future needs and threats that cannot possibly be known and/or characterized today.

Design for Changeability (DfC)

Fundamentally, the focus of DfC is on designing a system in accordance with certain principles (i.e., Designing *for* Principles!) rather than just designing a system to meet a specific set of requirements. Strictly speaking, DfC does not entirely abandon the notion of requirements, but rather seeks to balance traditional functional/performance requirements (e.g., speed, weight, resolution, latency) with non-functional requirements or "-ilities" (e.g., flexibility, adaptability, modifiability, changeability,

survivability) as part of the design process.[*] For this reason, DfC has far less dependence on the stability (or even awareness) of functional/performance[†] requirements.

The principal thrust of DfC is that a system must be able to have a sort of "capability fluidity" (this author's term) in order to continue delivering value over its lifecycle. As articulated by Kasarda, et al.—

> [The] concept is based on the hypothesis that product life ends because a product is unable to adapt to change. A product may be retired for myriad reasons including that it is broken, out of style, or has become inefficient due to technology obsolescence. In these cases, the product was not able to adapt to change—it was unable to self-heal, it could not modify or reconfigure to meet changing fashion needs, or it could not be upgraded, for physical or economic reasons, to utilize new technology.[4]

Furthermore, Fricke and Schulz established some "basic" and "extending" design principles for DfC, including:

- **Ideality/Simplicity.** Intended to reduce system complexity; "ideality" is defined as the ratio of a system's sum of useful functions against a system's sum of harmful or undesired effects.
- **Independence.** Intended to minimize the impact of changing design parameters; each system function or functional requirement should be satisfied by an independent design parameter.
- **Modularity/Encapsulation.** Intended to cluster system functions into various modules while minimizing the coupling among the modules and maximizing the cohesion within the modules.
- **Integrability.** Characterized by compatibility and interoperability applying generic, open, or common/consistent interfaces.
- **Scalability.** Characterized by scale-independence, which avoids significant performance degradation with small or large deployments and facilitates design reuse.
- **Decentralization.** Characterized by a decentralized distribution of control, information, resources, attributes, and properties within the system architecture.
- **Redundancy.** Enables capacity, functionality, and performance options as well as fault-tolerance.[5]

This work informed an Office of the Secretary of Defense (OSD)/Systems Engineering Research Center (SERC) research topic titled "Valuing Flexibility." The problem statement for that research read, in part—

> … future DoD systems need to be highly adaptive to rapid changes in adversary threats, emerging technology, and mission priorities, both during development and during operations. Traditionally, however, complex DoD systems have been designed to deliver optimal performance within a narrow set of initial requirements and operating conditions at the time of design. *This usually results in the delivery of point-*

[*] A full discussion of *functional* vs. *nonfunctional* requirements is beyond the scope of this paper. See Glossary of Terms at the end of this paper for a short explanation of the differences.

[†] From this point forward, I will simply use the term "functional" instead of the more cumbersome "functional/performance." Though many sources distinguish between functional and performance requirements (some even contending that performance requirements can be NFRs), the difference is not particularly important here. The salient distinction in this paper is between those types of requirements that relate to technical performance and those that pertain to design/architectural principles.

> *solution systems that fail to meet emergent requirements throughout their lifecycles, that cannot easily adapt to new threats, that too rapidly become technologically obsolete, or that cannot provide quick responses to changes in mission and operating conditions* [emphasis added].[6]

This statement is from 2012 and was intended to apply defense-wide. It could easily have been written today specifically for the DOD space enterprise.

DfC Is Not a Panacea

While flexibility is certainly a worthy goal, it is not an unqualified one. There are reasons *not* to pursue DfC.

> The problem is that flexibility necessarily incurs additional investment costs.… The notion of designing to the "bleeding edge" of performance requirements is antithetical to the aims of flexibility, as it consumes engineering tradespace. An inherently flexible design cannot, axiomatically, achieve the same level of technical performance along every dimension as the performance-optimized design.[7]

Consider a system for which we are extremely confident that we have nailed down all of the requirements and, that over that system's lifecycle, those requirements won't change and there won't be any new requirements. If all of these stipulations are true, then investing in flexibility would be pointless (this description would generally include simple commodities like clothing, can openers, toasters, pencils, etc.). On the other hand, given a system in which we aren't sure of the requirements at all, and we expect there will be new requirements and/or significant changes to the requirements at some point in the system lifecycle, investing in flexibility is the obvious choice.

The upshot is that there is a sort of "tipping point" that must be determined for each application or domain in which *Principled Design* as the right overarching methodology gives way to *Designing for Principles*. The argument in this paper is that developing a resilient architecture for the space enterprise is an excellent example where this tipping point has been reached.

Of course, this begs the question, "How can we actually employ DfC to achieve our goals?" How does the concept *really* work? After all, the SE Principled Design process may have its weaknesses, but at least we're familiar with it, and we know it can produce some amazing capabilities.

These are fair questions. The truth is DfC is certainly not nearly as mature as traditional SE methodologies (which have at least a 50-year head start). DfC is really the *kernel* of an idea—the basis for thinking about capability development in a new way that offers some stark contrasts to traditional SE approaches employed by the DOD. Although some important principles have been captured above, the task remains of translating these into a structured, repeatable methodology to guide capability development.[‡]

This paper attempts to partially remedy that, extending and generalizing the core concept of DfC into a framework to inform development of capabilities across the DOD space enterprise. This framework is referred to as "Designing for Principles" or simply "DfP." The essence of the DfP framework is captured via three pillars—

[‡] There have been some notable attempts to implement more rigor. One of the more interesting is "Designing Systems for Adaptability by Means of Architecture Options," by A. Engel and T. Browning (2006). Also consider the truly excellent, "Fundamentals of Decision Making for Engineering Design and Systems Engineering," by G. Hazelrigg (2014).

1. Don't Be Obsessed with Requirements
2. Keep the Big Picture in Mind
3. Embrace and Understand Uncertainty

Each of these pillars will be explored in the sections that follow, to include explaining the meaning of the pillar, discussing rationale for its inclusion, and highlighting salient differences with the traditional Principled Design approach. Several examples are provided to illustrate the purpose and power of these pillars, but the intent is to keep the discussion fairly general. As noted at the outset, specific, actionable recommendations on how to implement DfP across the space enterprise are deferred to the subsequent paper.

When Does DfP Apply?

Before addressing the pillars, let's first clearly establish the scope of DfP as a potential methodology. Thus far, this discussion has mostly related to systems. The question naturally arises, "How does the DfP concept apply to *architectural* development and design?"

Simply stated, every argument for the DfP approach to capability development is more compelling for higher levels of design abstraction, whether that be an enterprise architecture or a systems-of-systems (SoS) application. During early architecture development, the time to system fielding is greater, requirements are less well understood, and the design space is much broader. This is the point in time with the largest number of variables with the greatest amount of uncertainty. Recalling the notion of a "tipping point," if DfP makes sense during the system design phase, then it makes even more sense during the architecting phase.

In his book, *System of Systems Engineering*, Jamshidi observes that traditional SE assumes far more stability than is warranted in today's world, and these assumptions are particularly problematic at the architecture level:

> …twenty-first century engineering is witnessing an unprecedented change in the way we conceive, develop, field, and sustain systems. Many of the premises underlying the traditional systems engineering (SE) strategies are no longer valid. Traditional SE has been focusing on developing stand-alone systems with stable architecture and static technology base in which improvements were slow and very costly. These strategies incorrectly assume that all of the systems of systems requirements are known in the beginning of the development process and can be frozen in time or assumed to be stable. The traditional SE strategies also wrongly assume that the concepts of operation and various technologies used for constructing today's SoS are static and are subject to minor future changes.[8]

Another reason why DfP is well suited to higher-level architecting is because there are certain non-functional requirements that can best be assessed and delivered at the architectural level. Survivability (a concept closely related to resilience, to be addressed later) is a good example of this. If we wish to maximize the persistence of capabilities provided by the U.S. space enterprise, there are certainly some design strategies that pertain to individual systems or nodes (e.g., maneuverability, shielding, physical barriers, etc.). However, the truly powerful design strategies come into play at the SoS or enterprise architecture level (e.g., distribution, disaggregation, proliferation, entanglement, etc.).

Finally, the use of DfP at one level of design does not require its use at all levels. To wit, DfP is scalable. It may be used at the architectural level (or the Systems-of-Systems level), but traditional SE methods may still be used at the system level.

Don't Be Obsessed with Requirements

Why, sometimes I've believed as many as six impossible things before breakfast.

—Lewis Carroll
Through the Looking Glass

In traditional SE (especially as practiced within the DOD), requirements are the bedrock upon which everything else rests. DfP acknowledges that requirements are important, but it also harbors a healthy dose of skepticism for them, given established problems with overly rigid specifications and the inherent failings of requirement decomposition. Further, DfP also recognizes that that bedrock is malleable because requirements are, in reality, almost never stable.

Logically, the more requirements we have, the more opportunity there is for "requirements creep" and/or requirement failure. Further, each time there is a requirement failure—or just the perceived risk of requirement failure—there are significant resources expended in coping with that. This slows us down and costs us money. Also, the greater the number of requirements, the greater the likelihood that we make a mistake in decomposition. This means that we may focus our efforts on the wrong goals and/or not include the right ones. Finally, each time we decompose a requirement, we run the risk of optimizing sub-elements of the system at the expense of the overall system. For all of these reasons, DfP encourages the use of fewer requirements or the use of *objectives* in lieu of rigid requirements.

DfP prioritizes meeting long-term (unknown) requirements over meeting short-term (known) requirements.

In practice, this also means that DfP tends to emphasize flexibility over functional requirements. Take a moment to digest this deceptively simple statement, as it could be considered blasphemy. To many SE practitioners, nothing is more important than ensuring we provide a capability that meets a validated set of functional requirements. However, it may be less contentious when couched this way: DfP prioritizes meeting long-term (unknown) requirements over meeting short-term (known) requirements.

In the end, it's really a question of robustness to change vice slavery to specification. DfP is essentially posing the question, "Does it make sense to expend enormous time and resources carefully specifying and decomposing large numbers of stringent requirements under the assumption that those requirements are not only fully valid now, but will remain so over the lifecycle of a defense system that typically spans decades?" The DfP answer is a resounding "no," at least under conditions of high complexity and extreme uncertainty.

Objectively Speaking

So what would it actually mean to capture fewer requirements or to use objectives in lieu of requirements? The most effective point in the requirements process to reduce the number of requirements is at the apex of the decomposition effort—i.e., Key Performance Parameters (KPPs). This is because the decomposition process spawns requirements on an exponential basis. Consequently, having fewer KPPs, even one or two fewer, could easily result in hundreds of fewer

system specification requirements. There is even an argument to be made that the right number of top-level requirements[§] is *zero*, at least in certain situations. After all, how many requirements did the Internet have (see upcoming section for a short case study on the Internet)?

If prioritizing flexibility over functional requirements is blasphemy, then not having requirements at all might be viewed as high treason. How can we possibly develop a sophisticated defense capability without requirements? After all, if we don't have requirements, what will we decompose, allocate, analyze, argue about, test, verify, change, and ultimately fail to meet? Okay, that rhetorical question was, perhaps, unduly cynical. Here's a fairer question (or compound question): Without requirements, how will we know that what is being developed will have value to the end user, and how will we hold the developer accountable for delivering something of value?

The answer is by using *objectives*. And this isn't as crazy as it sounds. The basic idea is prevalent in other disciplines, including education (e.g., objective-based learning or outcome-based education), finance (e.g., objective-based investing), and business operations (e.g., objective-based management). By specifying objectives in lieu of requirements as part of our capability development strategy, we can convey what it is we really wish to achieve while simultaneously greatly expanding the design tradespace we have to work with, both now and across the lifecycle. This can be a particularly good match for developments entrenched in uncertainty, as it is an intrinsically flexible approach to foster flexibility.

Consider this example. Suppose someone knows they're overweight, and their *objective* is to lose body fat. To achieve this, they might impose a *requirement* on themselves to reduce their weight by 20 pounds over the next six months. Assume that three months later—after increased exercise and decreased ingestion of Skittles—they've lost seven pounds. Is this a good outcome?

From a traditional requirements management mindset, things are not going well. It does not appear this person is on track to achieve their weight loss requirement, and so they might be inclined to give up and try something new. But, in reality, this person has likely made progress toward their objective; i.e., they've almost certainly reduced their body fat.

Now let's suppose that—at the three-month mark—this hypothetical someone suddenly decides to begin training to be an American Ninja Warrior, and they wish to develop a lot more muscle mass. After three additional months of intense training and bland protein shakes, their weight does not reduce further, but they find they have a significantly lower body fat percentage than ever before. After six months, this person weighs only seven pounds less than they did originally. Is *this* a good outcome?

If we only go by the *requirement* of losing 20 pounds, then they have failed. If, instead, we go by the *objective* of losing body fat, then they have succeeded. And because they prioritized their ultimate objective (i.e., body fat reduction) above the derived requirement (i.e., weight loss), they had the flexibility to shift from losing weight to gaining muscle once something unexpected happened (like the decision to try out for American Ninja Warrior).

This scenario is rather trivial, but it does illustrate the potential of employing an objective-based approach. It allows us to remain focused on the

[§] The "top-level" qualifier is critical to this argument. It's one thing to have a latency objective of 60 minutes and then achieve 80 minutes; it's quite another to specify an interface connection be at least 5 cm and then provide a part that is only 4 cm.

outcome that matters while simultaneously encouraging incremental progress toward that outcome even if lower-level requirements aren't being met.

Finally, it's important to note that an objective-based approach to acquisition should not be confused with the much-maligned *capability-based planning* prevalent during the early 2000s. Prioritizing objectives over requirements is not an excuse for gold-plating. It is not free license to build the coolest, most awesomest weapon ever just because we can. *Objective-based* simply means constraining the solution space to what really matters and eliminating—or at least greatly reducing—requirements where possible in order to increase flexibility.

Non-Functional Requirements (NFRs)

Having too many requirements can be cumbersome and counterproductive, slowing us down and constraining our solution space. But the more insidious problem may be having the incorrect *kind* of requirements. At present, the DOD is far more focused on functional (i.e., technical performance) requirements than non-functional requirements (i.e., the "-ilities"). For space system acquisitions in an increasingly threatened operating environment, this is almost certainly the wrong focus.

> ***We built exquisite glass houses in a world without stones.***
>
> — Heather Wilson, Secretary of the Air Force (2018)[9]

As long as capability developers remain consumed with functional requirements, the systems they field are likely to remain locked into the vicious cycle of increasingly exquisite point solutions and longer development timelines. This results in greater vulnerability of critical capabilities and an increasingly brittle space architecture. The Air Force recognized this problem, which is exactly why the seventh and eighth satellites of the Space-Based InfraRed System (SBIRS) program were cancelled in 2018, replaced with a successor system intended to be more survivable.[10] The thinking behind this decision was essentially this*: no matter how capable a system is during peacetime, if it's likely to not be available when it's needed most (like during a major conflict), then there's little point in having it.*

We need more of this kind of thinking. We need to pivot the focus away from functional requirements toward non-functional requirements (NFRs). Truly shifting the paradigm will mean that leadership is willing to routinely sacrifice some technical performance capability in order to enhance the survivability of that capability. That may mean we employ smaller, less capable, primary satellite payloads in order to make room for warning sensors or increased fuel reserves. It may mean accepting the implicit performance offsets associated with using a standard interface for modularized satellite buses and launch vehicles. It may mean incurring latency in order to diversify capabilities among commercial or Allied partners. It may mean that a particular satellite ground station is less able to support a single mission function, but better able to support a broader range of mission functions.

To cope and thrive within this complex environment, DfP calls for a greater commitment to NFRs, prioritized at least as highly as technical performance requirements. This is not only the best way to counter known threats in a complex environment—it's also the best way to respond to *unknown* threats.

Prediction is not the only way to confront threats; developing resilience, learning how to reconfigure to confront the unknown, is a much more effective way to respond to a complex environment.

—General Stanley McChrystal
Team of Teams[11]

Acquiring the Internet

The Internet is an excellent case study of the power of DfP, especially vis-à-vis traditional conceptions of requirements. Viewed from the perspective of a major system development, the Internet has to be regarded as an extremely impressive accomplishment. Of relevance to this discussion, it was achieved *without any formal requirements* and with intentional *emphasis on non-functional requirements*.

Most people likely are aware the modern Internet had its origins as a Defense Advanced Research Projects Agency (DARPA) research effort. DARPA had a simple, focused objective: establish communication protocols for networked computers. They approached this in a way that exemplifies DfP. They focused on a single objective instead of a litany of requirements, and the objective was related to an -ility (i.e., *interoperability*, in the form of an interface standard) instead of a performance parameter. The result was the foundational Transmission Control Protocol/Internet Protocol (TCP/IP) suite.

Eventually, a number of government agencies invested in backbone networks that conformed to the TCP/IP standards. They, too, saw promise in the concept, but none of them created a formal program with a plethora of formal requirements. With the infrastructure and the interfaces established, incentives were in place for the nascent Internet to expand rapidly. Government entities (state and federal), educational institutions, and various private sector companies took advantage of this networking infrastructure, expanding the reach of the Internet by filling in local and regional connections across the globe.

And because of certain design principles pursued at the outset of this effort (e.g., interoperability and extensibility), it was relatively easy to accommodate another key change, i.e., the incorporation of additional protocols, such as the Open Systems Interconnection protocols. This allowed for the inclusion of functionality that could not possibly have been foreseen originally (e.g., user-friendly browsers, electronic mail, hyperlinking, web-based applications, etc.). This flexibility was critical to the stunning success of the Internet, though perhaps also the bane of many parents of young children.

And although the Internet is subject to local outages and targeted denial of service attacks, it is practically impossible to take down the entire thing. It is almost certainly the most resilient, major system that humans have ever created. This is a remarkable achievement that some are too prone to attribute to luck. Though luck played a part, it was also the result of some smart, deliberate design decisions—

> The Internet, and consequently its backbone networks, do not rely on central control or coordinating facilities, nor do they implement any global network policies. The resilience of the Internet results from its *principal architectural features* [emphasis added], most notably the idea of placing as few network state and control functions as possible in the network elements, and instead relying on the endpoints of communication to handle most of the processing to ensure data integrity, reliability, and authentication.[12]

Contrast this Designing for Principles approach with the Principled Design approach. Imagine if the DOD had established a program early on to develop the

capabilities we see in today's Internet. These are the types of KPPs we might expect:

- The system shall support at least three billion unique users on a daily basis
- The system shall support at least one billion unique content sites
- The system shall support exchange of at least 200 million rich text messages every minute
- The system shall allow users to access sites and content via any device that conforms to a few simple communication protocols
- Users shall be able to use a protocol-conforming device to send 1MB of data via the system to any protocol-conforming device anywhere on earth in less than one second
- The system shall allow for near-instantaneous search of all hosted content
- The system shall support secure exchange of sensitive data
- The system shall achieve 99.9 percent functional availability

If these *had been* the KPPs, we could reasonably expect that this program would either:

- Not have delivered on even a small fraction of the performance requirements
- Have been cancelled
- Be the most expensive program in the history of civilization

And truthfully, it probably would have managed to be all three.

At best, the Principled Design approach would have been cost-prohibitive, taken decades to implement, and/or delivered only a subset of the performance attributes. On the other hand, *the DfP approach gave us greater performance and resilience than we could have ever imagined possible*. Granted, no one could have envisioned the full potential of the Internet at the outset. But that's precisely the point. The Internet was poised to take advantage of opportunities that could not possibly have been foreseen because the developers deliberately chose to emphasize objectives over requirements and prioritize NFRs over technical performance.

The DfP approach may be sacrilege, but it is potentially transformative sacrilege.

Keep the Big Picture in Mind

"Would you tell me, please, which way I ought to go from here?"

"That depends a good deal on where you want to get to," said the Cat.

"I don't much care where—" said Alice.

"Then it doesn't matter which way you go," said the Cat.

— Lewis Carroll
Alice's Adventures in Wonderland

One of the more interesting aspects of NFRs is that many of them can best—and sometimes only—be achieved via a broader, more strategic approach. Case in point is *resilience*. This is a concept that is referenced extensively in space architecture discussions; it is also a decidedly *architectural* attribute. In DOD space policy, resilience is defined as, "the ability of *an architecture* [emphasis added] to support the functions necessary for mission success…"[13] In joint-capability parlance, resilience is the "ability of the *collection of systems* [emphasis added again] to support the functions necessary…" and applies to "the overall force (broader than a

single system architecture) to complete the mission despite the loss of individual platforms."[14]

These definitions of resilience make it clear we're not talking about a single system, but rather a systems-of-systems architecture or an enterprise architecture. This means that resilience is not something an individual program is responsible for. This is not to say that system developers have no role in delivering a resilient architecture. There are certainly many smart things that can be done at the system level to contribute to the overall resilience of the enterprise space architecture. Examples include:

- Apply radiation shielding to satellite vehicles
- Erect physical barriers around ground facilities
- Hide, or otherwise disguise, the location and purpose of the ground facility
- Increase antenna transmission power
- Implement communication link frequency hopping
- Maintain network antivirus protection

But the really powerful resilience strategies can only be implemented—and meaningfully assessed—at the enterprise architecture level. For instance:

- Standardization of interfaces (satellite, launch vehicles, ground systems, etc.)
- Distribution of capabilities across multiple systems
- Proliferation of systems across multiple orbital regimes or geographic locations
- Entanglement of capabilities with commercial or Allied systems
- Provisioning of deployable, integrated defense force packages
- Establishment of rapid deployment (e.g., rapid launch) and sustainment (e.g., on-orbit servicing) capabilities

Fundamentally, an enterprise architecting approach that emphasizes DfP shifts the focus from the resilience of a particular *system* to the resilience of an overall *capability*. This allows capabilities to be apportioned among systems in a way that achieves strategic levels of survivability (perhaps the most important NFR of all for defense systems) simply not attainable on a system-by-system basis. This approach recognizes that doing what's best for the entire space enterprise often is incompatible with doing what is best for individual systems. This notion is the essence of *Keep the Big Picture in Mind*, the second pillar of DfP.

Capabilities, Not Systems

Because the more you optimize elements of a complex system … for some specific goal, the more you diminish that system's resilience. A drive for efficient optimal state outcome has the effect of making the total system more vulnerable to shocks and disturbances.

— B. Walker and D. Salt
Resilience Thinking[15]

Standard decomposition practices often result in the optimization of the pieces rather than the whole, and the more complex a system is, the greater the risk of optimizing the wrong thing. Relative to capability development strategies, we need a way to not lose focus on the mission capabilities needed by the warfighting community. Arguably, we do not have

such a framework at present. Figure 1 depicts (in a greatly simplified manner) the current relationship between capabilities and systems across the space enterprise.

The process starts with the identification of a general mission capability need. For the space enterprise, this includes things like Missile Warning, Environmental Monitoring, and Satellite Communications, et al. From there, we establish performance measures that are assigned to a particular system development activity. Those small, light brown blobs represent a notional spider chart of top-level requirements (e.g., KPPs) where we capture Missile Warning performance measures for SBIRS and another set for DSP (Defense Support Program) and so on ("GOES" is Geostationary Operational Environmental Satellite; "AEHF" is Advanced Extremely High Frequency; "WGS" is Wideband Global SATCOM). As each of these systems is developed and fielded, we can think of the resulting architecture as an equation: the sum of the individual system capabilities is equivalent to the capabilities of the entire space enterprise. In this way, we have achieved a space architecture that is the fact-of-life, non-deliberate consequence of aggregating the constituent systems.

Note that this process effectively precludes enterprise architecting, as *it makes it virtually impossible to optimize for anything other than individual systems.*

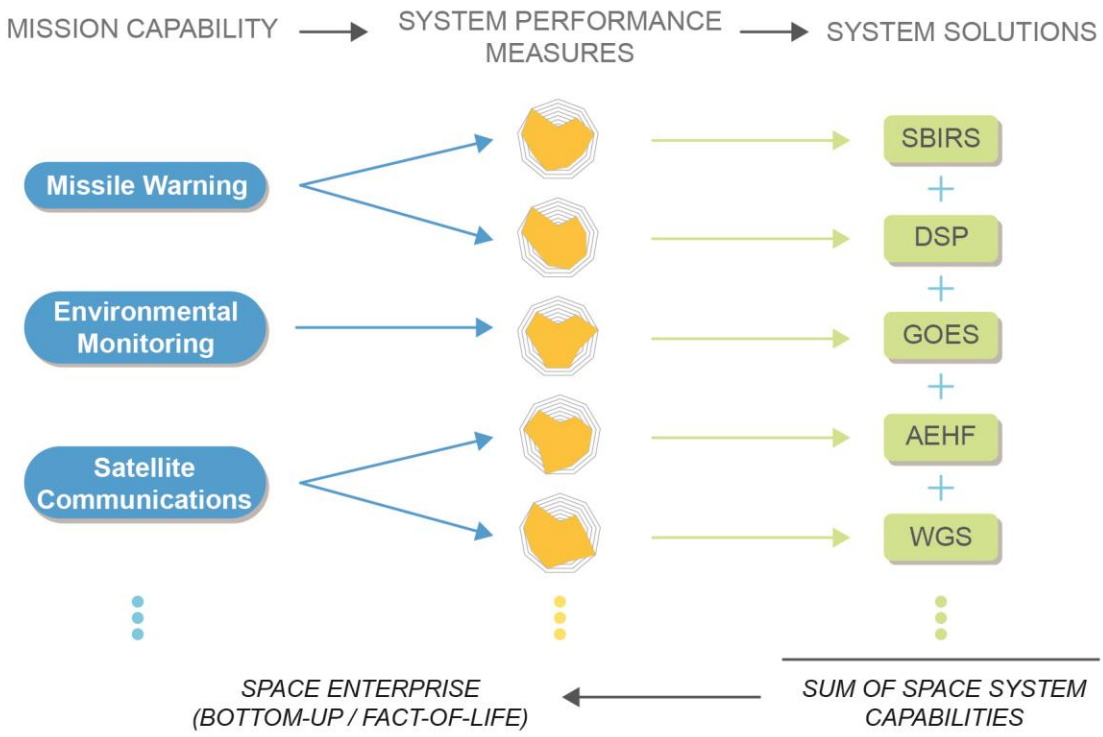

Figure 1: Current system-centric approach.

Consider, instead, the alternative process in Figure 2; this reflects the DfP-driven approach. We start with the same mission capabilities,[**] but now, instead of developing performance measures that are intended to guide system development, we stay at the capability level. We document what performance we need relative to that entire mission capability (i.e., "System Performance Measures" are replaced with "Capability Performance Measures"). Notably, there is no consideration, yet, of what specific systems will deliver those capabilities. The sum of all the needed capabilities across all the mission areas represents the complete objective for space enterprise capabilities. We next use that objective as the basis to develop an *enterprise* architecture, one that takes into account all needed capabilities simultaneously and balances them against one another, as well as NFRs, to ensure optimization of the top-level objectives.

This is a subtle, but crucial, difference. This second, capability-centric approach allows the architecture to be realized through the deliberate acquisition of a set of systems (and services) that support the capability performance measures without having to build a series of exquisite, monolithic systems that may be excellent point solutions, but not-so-excellent enterprise solutions. The result is likely to

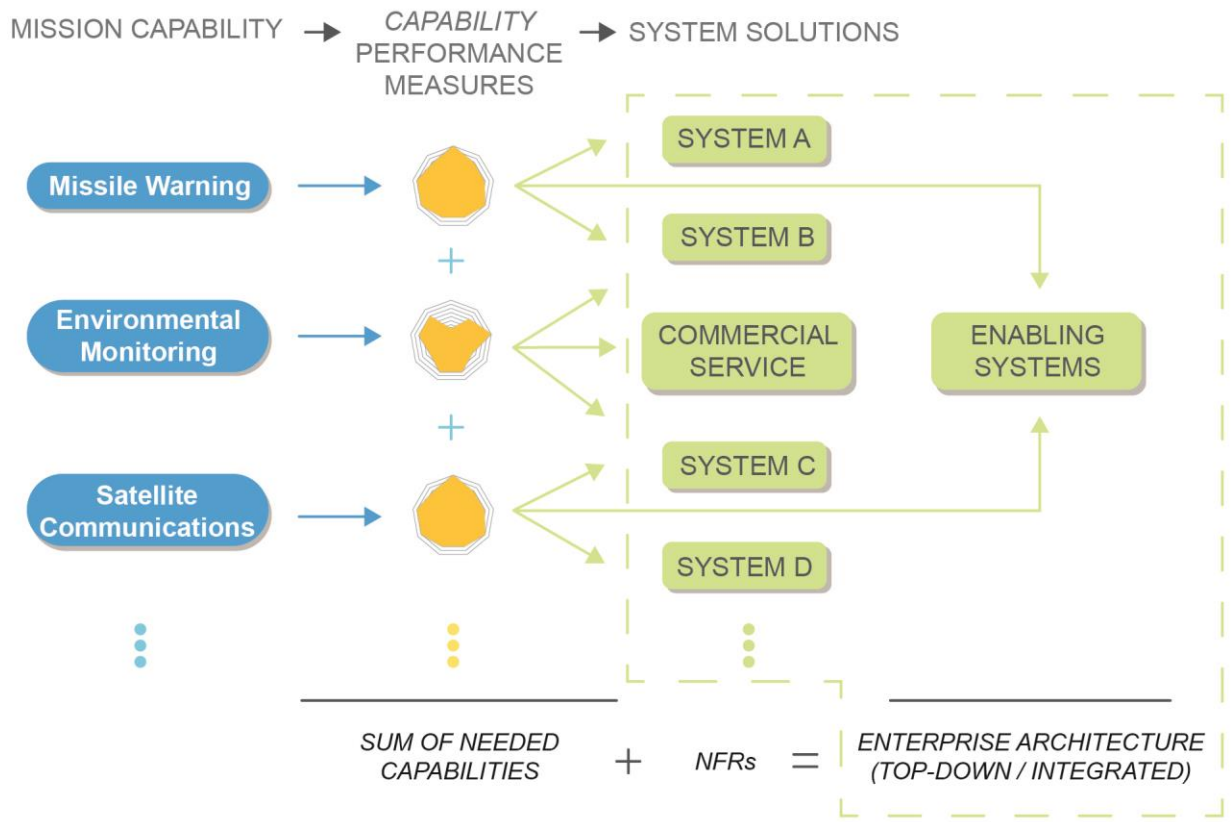

Figure 2: Proposed capability-centric approach.

[**] An even more innovative approach would be to reevaluate the suitability of these mission capability categories; a different capability ontology could yield revolutionary solutions. However, that goes beyond the scope of the current discussion.

include some systems that are still focused on supporting a single mission capability area (e.g., "System A"), but there will also be systems that support multiple capability areas (e.g., "System B") in order to achieve greater efficiencies and realize enterprise architectural goals like proliferation, distribution, and disaggregation.

Importantly, the proposed capability-centric approach also elevates derived, enabling capabilities (depicted as "Enabling Systems"). This allows for explicit support of cross-cutting capabilities like Launch, Data Transport, and Protection, all of which are not of direct concern to the warfighter, but are crucial to the success of other mission capabilities. These enablers may be best achieved via common, enterprise solutions instead of stove-piped, system-specific solutions. Finally, by considering system capabilities in the aggregate, we can make smarter allocations of capabilities across the enterprise, involving contributions across not just the DOD, but also civilian, commercial, and Allied partners.

To summarize the difference between the two approaches, think of the U.S. space enterprise as a *group of wolves*. Using this analogy, Figure 1 represents the current situation (i.e., the Principled Design approach) where we have a number of lone wolves each hunting in their own territory using the techniques they're good at. They may survive for some time scrounging for berries and small game while avoiding predators, but their odds of long-term survival are not great as they have virtually no capacity to overcome adversity.

Figure 2, on the other hand, represents the power of the wolf pack (i.e., DfP). They have unity of purpose and unity of effort. Here, all the wolves are working together in a coordinated fashion, complementing one another's strengths and compensating for one another's weaknesses to achieve gestalt. Consistent with *Keeping the Big Picture in Mind*, the survival of an individual wolf is largely inconsequential; it is the survival of the wolf species that matters. In this way, the pack of wolves are far more resilient to a wide range of threats, both known and unknown.

Embrace and Understand Uncertainty

If it was so, it might be; and if it were so, it would be; but as it isn't, it ain't. That's logic.

—Lewis Carroll
Through the Looking Glass

Embracing Uncertainty, the third pillar of DfP, is all about recognizing uncertainty, accepting it, and treating it in a mathematically rigorous way. This may sound like a simple change in thinking, but it's far from it. Truly accepting uncertainty would represent a momentous change in capability development because it has implications to virtually every aspect of the traditional SE processes. Relative to traditional conceptions of risk, DfP (1) has a diametric attitude toward considering it, (2) expands the scope of what should be considered, and (3) posits an alternative core purpose to managing it.

The entire DfP mindset regarding uncertainty is drastically different. Whereas conventional SE necessarily abhors uncertainty, DfP thrives on it. This is largely attributable to the established positive correlation between uncertainty and the value of flexibility.[16,17,18] So whereas Principled Design approaches risk as a reluctant participant, dismayed by the emergence of any new risk, and continually seeking to "mitigate" it wherever possible, DfP *embraces* uncertainty, knowing that every source of uncertainty serves as greater justification for investing in flexibility, to include from an overall affordability perspective.[19,20]

Scope is the second key difference. The DOD risk management process typically only concerns itself with the downsides of uncertainty, ignoring all the

possible upsides, i.e., *opportunities*. DfP, however, inherently involves both positive and negative implications of uncertainty, whether we wish to thwart a new adversary threat (risk) or take advantage of a new technology advancement (opportunity). Thus, DfP is more than risk management—it's actually *uncertainty* management. This distinction is important not just because both risks and opportunities must be accounted for in any robust valuation of tradespace, but also because it is often not clear whether a source of uncertainty is simply "good" or "bad." Binary categorization of all uncertainty into one of these two buckets is a gross oversimplification that ignores how assessments can change over time.

Lastly, and most importantly, the entire purpose of trying to manage uncertainty has a different focus under DfP. Put simply, the traditional risk management process is tactical in nature, while DfP uncertainty management is inherently more strategic. Programs that employ risk management are understandably consumed by things that can jeopardize their established cost, schedule, and/or technical baseline. These are important considerations to be sure, but what actually matters is that the joint warfighter is provided with the right capability at the right time. This realization leads to a reconception of risk, one that is based on *operational capabilities* instead of *programmatics*.

This realization leads to a reconception of risk, one that is based on operational capabilities instead of programmatics.

Broadening our conception of risk management (again, really *uncertainty* management) to a strategic level also supports the second DfP pillar: *Keep the Big Picture in Mind* (the pillars should not be regarded as independent). Focusing on operational capabilities instead of system acquisitions reprises the central message of Figure 2, which urges us not to view the world through the lens of an individual program.

To illustrate this point, imagine that we have chartered four programs to deliver a high-risk capability as part of a considered acquisition strategy to respond to a critical operational need. Assume that the chance of any individual program delivering this capability is estimated to be 50 percent. This is certainly an extremely high risk from the perspective of any one of these programs, but as long as all aspects of each program are fully independent from one another, the chance that at least one program will succeed is nearly 94 percent.[††] In this example, the tactical-level, program risk is unacceptably high, but the strategic-level, operational capability risk is easily tolerable.

Moreover, failing to meet cost, schedule, or technical requirements does not necessarily mean that we have failed from an operational perspective. Suppose, instead, only one program was commissioned to respond to this operational need, and that it fell short on the technical requirement by 20 percent. Recall one of the messages from the first DfP pillar (*Don't Be Obsessed with Requirements*) that we should focus on objectives in lieu of requirements. From an operational perspective, an 80 percent solution available right now is likely to be preferable to a 100 percent solution five years from now.

Regardless of the attitude toward uncertainty, the scope of it, or the purpose of managing it, the real

[††] Based on a standard binomial probability distribution

key is how we regard it and characterize it in our capability development strategies. And this may be the biggest difference of all between Principled Design and Designing for Principles. Because traditional SE is not well suited for accommodating change, the natural tendency is to ignore or downplay sources of uncertainty. In a highly uncertain environment, this head-in-the-sand approach is the worst possible tactic. DfP, in contrast, acknowledges that uncertainty is a fact of life that can be understood—and sometimes exploited—as long as we are willing to confront it in a robust way. The rest of this section explains how this can be achieved.

The Failing of Functional Availability

DfP treatment of uncertainty can be understood by considering the widely used concept of satellite constellation *functional availability* (FA). This metric is often used as the principal basis for assessing the current and forecasted health of a satellite architecture and thus informing multi-billion-dollar replenishment decisions.

In this author's younger days, he had a colleague who would often proclaim that "consistency is better than accuracy." Anyone who has ever been involved in the development of satellite constellation FA numbers can likely attest to the truth of this aphorism. There are a lot of assumptions buried in the FA calculations that mask high levels of uncertainty. But the output product is generally achieved through a consistent methodology and has the benefit of being readily understood by senior leaders. As a result, the FA numbers are often viewed by decisionmakers as being more accurate and reliable than they really are.

> ***The way we have acquired space systems for the past several decades was on the basis of something called "functional availability," … That is an approach that might make sense in a benign environment but … that's not the environment we find ourselves in anymore.***
>
> — Winston Beauchamp, Deputy USecAF[21]

In recent years, the entire concept of satellite FA has (rightfully) come under scrutiny. The traditional FA metric only considers the probability of internal component failure as a result of reliability calculations and, to a small extent, naturally occurring threats. As Beauchamp notes, we need to expand the concept of FA significantly. To make FA a truly useful metric, we would need to consider all threats to required mission capabilities, especially *artificial* threats (i.e., bad guys doing bad things to our stuff).

But this is easier said than done. In order to quantitatively analyze the risk of artificial threats, there are a number of questions that become relevant, including:

1. Who could hurt us?
2. How could they hurt us?
3. *Would* they hurt us?
4. How would we respond?

These are sensible questions, but providing meaningful answers is a daunting endeavor, primarily because none of the answers can be known with certainty. The answers we would likely have

the most confidence in—those to questions one and two—would likely be based on intelligence estimates, which are still, well, *estimates*. But the uncertainty in those answers would be relatively low compared to our answers to questions three and four. Knowing what an adversary will do is extremely difficult to predict even in the best of circumstances, let alone during the fog of war. And knowing what our nation would do in response is not much clearer (care to guess how the U.S. would respond to every known threat from every known actor?). These large uncertainties translate to large probability distributions. And, of course, the degree of uncertainty will increase, exponentially, over time (care to guess how future political administrations will react?).

And all of this ignores Black Swans[‡‡] we cannot (or will not) predict such as disruptive technologies, a coronal mass ejection, or realization of the Kessler syndrome. Even with all of these challenges, we could construct some probability curves and try to convince ourselves they are authoritative. We could then conduct the statistical analysis that yields a discrete probability distribution of the expanded FA concept. But who, in good conscience, would try to sell this result as something remotely accurate or reliable? The sheer number of overlapping probability curves and associated large uncertainty ranges would render the output effectively meaningless. It would be *sound and fury, signifying nothing*.

There are numerous, well-documented cognitive biases at play here, to include overconfidence effect, illusion of control, and planning fallacy, just to name a few.[22] But the real problem is more epistemological. As the collective community charged with developing space warfighting capabilities in a rapidly changing world, we need to acknowledge that there are some things we just do not know and that we cannot meaningfully characterize. This notion is deeply disturbing to some; nevertheless, it is true, and it is a fundamental reason why Principled Design is flawed.

As the collective community charged with developing space warfighting capabilities in a rapidly changing world, we need to acknowledge that there are some things we just do not know and that we cannot meaningfully characterize.

Complexity and Deep Uncertainty

In casual parlance, *complexity* is a synonym for *complicated*. However, within the fields of systems theory and operations research, the two concepts are distinct.

A "complicated" system will certainly consist of many parts and/or processes, and may have many intricate interactions. But the elements are finite, and their interactions are relatively few in number and can be characterized such that system outputs can be reliably predicted based on inputs. Examples of complicated systems include software programs, combustion engines, and 3D printers.

Systems regarded as "complex," on the other hand, not only tend to have more parts and/or processes but, more crucially, the number of interactions is generally much greater, and the nature of those interactions is often nonlinear and unpredictable. Knowing the inputs does not allow us to determine the output. Examples of complex systems include local weather, national economies, and insect swarms.

Relevant to the discussion at hand, *complicated* systems can generally be decomposed whereas

[‡‡] Highly improbable, but highly consequential events that are often disregarded in planning and analysis

complex systems cannot. The gestalt of complex systems means the holistic behavior cannot be reliably discerned by summing the parts. In other words, we could fully characterize all of the constituent elements of a 3D printer and how they fit together so that someone else could recreate one that theoretically performed just like the original. The same cannot be said for a localized thunderstorm. The complexity of all the constituent air molecules, thermal radiation, and flow dynamics (and the interactions between) is simply too complex. Even if we knew all the characteristics of all the constituent molecules in the region (pretend it's a *closed* system), we cannot possibly know the full characteristics and behavior of the thunderstorm.

Complexity produces a fundamentally different situation from the complicated challenges of the past; complicated problems required great effort, but ultimately yielded to prediction. Complexity means that, in spite of our increased abilities to track and measure, the world has become, in many ways, vastly less predictable. This unpredictability is fundamentally incompatible with reductionist managerial models based around planning and prediction. The new environment demands a new approach.

—General Stanley McChrystal
Team of Teams[23]

An essential element of complexity is an absence of knowledge and a high degree of uncertainty. This high degree of uncertainty is sometimes referred to as "deep uncertainty." This term is being used increasingly often in the decisionmaking domain to describe situations in which we have virtually no idea what the probabilities are associated with various inputs and factors (in other words we have a lot of uncertainty about the uncertainty) or how these relate to outputs and outcomes.[24] Deep uncertainty recognizes that these types of situations should not be treated in the same quantitative manner as your run-of-the-mill uncertainty problems where some degree of characterization is feasible.

This notion of deep uncertainty has given rise to an alternate decision framework known as "robust decision making" (RDM). RDM is better suited to deal with *uncertainty about uncertainty* as it rejects single probability distributions in favor of sets of probability distributions, emphasizes robustness over optimization, and provides a more explicit tie to foundational assumptions in order to improve decisionmaking. RDM is a natural companion to DfP as both concepts inherently embrace uncertainty.

Returning to the FA example, a fundamental question to answer is, "How do we go about determining the best constellation design?" Or, put another way, "In the face of myriad natural and artificial threats, what constellation architecture has the best chance of providing the capabilities we need when we need them?" The only thing certain here is that this is *not* a trivial question.

In attempting to answer this question, one option would be to employ the time-honored Principled Design approach along with the conventional treatment of uncertainty that assumes we are only dealing with complicated systems. Most readers are probably familiar with the basic approach.

First, we would conduct an Analysis of Alternatives that seeks to optimize constellation performance based on a set of fixed evaluation criteria (e.g., cost, schedule, technical performance), informed by "known" factors (e.g., threats, technology, market conditions), each with "known" certainty. We would then apply our evaluation criteria to the

proposed architectures across a set of "known" future scenarios informed by the factors with "known" probability distributions. We would then arrive at a single, preferred design that performs the "best." This would arguably be the "right" decision; however, it would be based on our knowledge today and a misguided confidence in the accuracy and stability of that knowledge. (And yes, that was a lot of scare quotes for one paragraph.)

Option two would be to assume system *complexity* and employ the principles of DfP and RDM. In this approach, we would consider design solutions in the real world, a world where there is not just variance in relevant factors, evaluation criteria, and scenarios, but the nature of that variance defies meaningful quantification. This methodology would likely *not* result in a single point-solution design that is optimized for current conditions, but instead provides a "family" of design options (an "architectural vector," if you will) that would be expected to perform the best over the range of possible futures and can most readily adapt and continue to deliver value as circumstances change.

In the end, we must recognize that most of our world is much more complex (in the strict sense of the word) and much more unpredictable than most of us are comfortable with or would care to admit. And this is certainly true in the contemporary space environment. The best way to cope with this is by being as flexible as possible and keeping our options open so we're better equipped for whatever lies beyond the horizon.

Architectural Tradespace

To better illustrate the difference between these two approaches, and the power of DfP in an uncertain environment, consider Figure 3. Each colored box (numbered 1 through 5) is intended to represent a separate candidate architecture. The total area of each box correlates to the expected value of that candidate architecture. The light red rectangle marked "T" depicts the threat environment we currently anticipate. The degree of overlap a given candidate architecture has with the red threat box represents the extent to which that architecture can continue delivering value in the face of that threat. In sum, the larger the box, the better, and the more overlap with "T," the better.

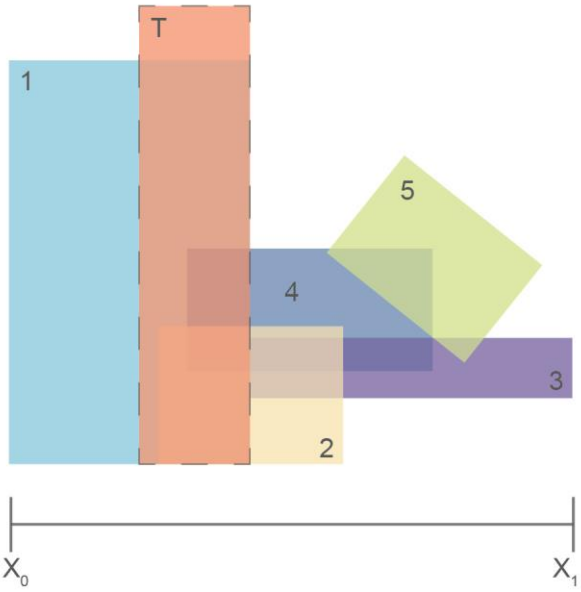

Figure 3: Candidate architectures (simplified).

With just this information to go on, it would seem that candidate architecture #1 is clearly superior because the size of box #1 is not only significantly larger than any of the other boxes, but it almost entirely encompasses the anticipated threat. Although the combined area of boxes #2 through #5 is equal to the area of box #1 (trust me, I've done the math), each individual box is significantly smaller, and only boxes #2 and #4 overlap with the threat at all (and then only to a limited extent).

So our work is done here, right? Let's go with candidate architecture #1 and be on our way.

Not so fast. There are some other pieces of information that you should take into account before you make your decision. The first thing to know is that the nature of the threat is highly uncertain; in

other words, the red "T" box can vary significantly. For simplicity, let's assume that it will remain the same size and shape, but may slide horizontally anywhere between X_0 to X_1 with equal probability.

The second thing to know is that candidate architectures #2 through #5 are all closely related, such that it is relatively easy to transform certain facets of one architecture into one (or more) of the others late in the design process, or even after implementation (this is what the overlapping areas represents). However, we must commit to a general path now: either Option "A," corresponding to candidate architecture #1 or Option "B," corresponding to the "family" of candidate architectures #2 through #5.

These additional pieces of information are depicted in Figure 4. Assuming the costs are the same, which option is better? It's not so clear anymore. On the one hand, Option A still performs the best as long as we assume that the threat environment remains constant or moves left toward X_0. On the other hand, movement of the threat toward X_1 quickly makes Option B better with respect to its improved threat response. In addition, there is clearly inherent value in being able to rapidly revector between options #2 through #5. Logically, if a thinking adversary observes us advancing toward a particular candidate architecture, they are more likely to counter with threats that exploit weaknesses in that architecture. Option B gives us more tradespace to respond while simultaneously complicating the adversary's design calculus.

It should be clear that Option "A" is a proxy for the Principled Design approach, whereas Option "B" is the DfP approach. Traditional SE wants to nail down the requirements and start working toward *the* design solution as soon as possible. DfP, on the other hand, seeks to maintain options for multiple design solutions as long as possible and be poised to move between them should circumstances change. There is a solid argument that Option B is the

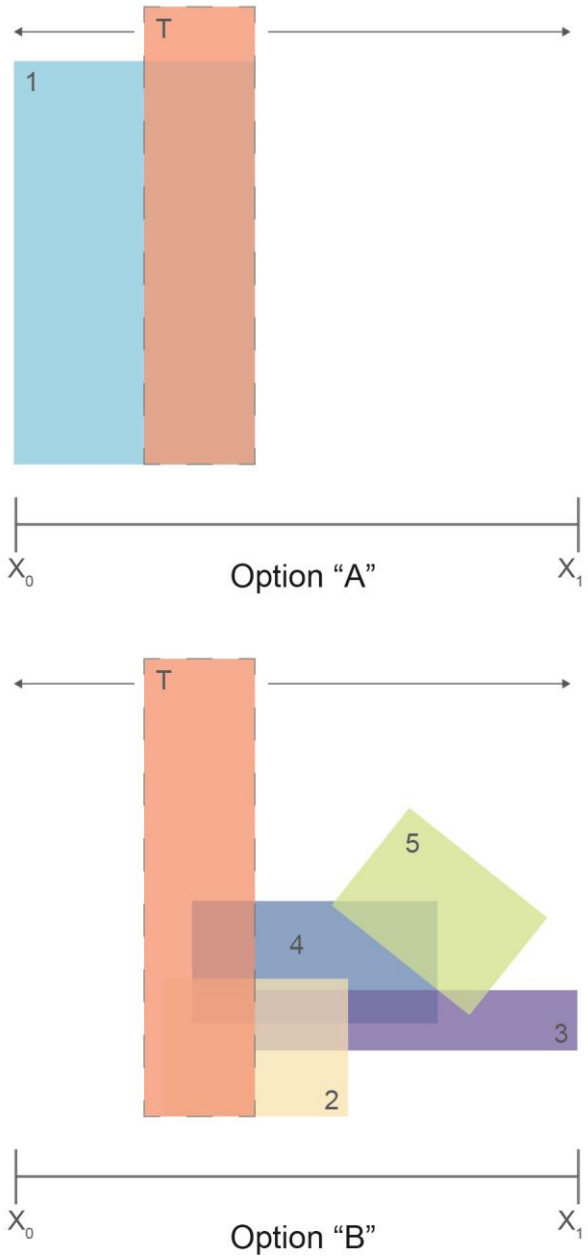

Figure 4: Candidate architectures (Not so simplified).

smarter way to go regardless, but that argument becomes increasingly compelling the more uncertainty there is regarding each parameter we considered, whether it be the number of candidate architectures, the value of those architectures, the nature of the threat environment, or the extent to

which the threat environment may change. Moreover, Option B recognizes the fact that the design process is never truly done, but is a never-ending series of design re-vectors.

This is the essence of capability fluidity. DfP fundamentally assumes that *genuine resiliency does not come from picking one optimal future architecture now; it comes from the ability to continually adapt the architecture across a range of future needs and threats, many of which we cannot reasonably predict today.*

Principled Design vs. Designing for Principles

To better appreciate the salient differences between Principled Design and Designing for Principles, it may be useful to see a side-by-side comparison of the two. If so, Table 1 is just for you.

In terms of specific design principles, the emphasis of DfP, not surprisingly, is on strategies like component modularity, physical component distribution, functional disaggregation, and implementation/adoption of open protocols and standards. Table 2 shows what that might look like in the context of space systems.

Summary of Designing for Principles

I know who I was when I got up this morning, but I think I must have been changed several times since then.

— Lewis Carroll
Alice's Adventures in Wonderland

Design for Changeability (DfC) was an idea introduced over a decade ago. The core premise is simple, but potentially transformational: the useful life of virtually every product or system is constrained by its inability to adapt to change. The logical extension of this insight is that if we are to sustain the utility of a system over a prolonged time period, we must *design for changeability*; i.e., design the system such that it can better accommodate change. According to DfC literature, this includes a host of specific non-functional design strategies such as flexibility, adaptability, and agility, as well as the adoption of design principles like simplicity, independence, modularity, scalability, and redundancy.

This paper introduces the concept of *Designing for Principles* (DfP), which amplifies and extends the basic tenets of DfC with the intent of making it implementable as a capability development methodology across the DOD space enterprise. Unlike the traditional DOD implementation of SE, which is generally requirement-based, system-centric, and driven primarily by technical mission performance, DfP is *objective*-based, *capability*-centric, and driven primarily by *design principles*.

Although the proposed DfP approach is still far from a formal methodology, it does provide a more structured and actionable framework for use in developing and fielding DOD capabilities. This is achieved, in part, through the establishment of three, interrelated pillars, which were described in detail and related to development strategies across the space enterprise. These pillars are summarized as follows:

- *Don't Be Obsessed with Requirements*. Reduce the total number of requirements, use objectives in lieu of requirements, and prioritize non-functional requirements over functional requirements.

- *Keep the Big Picture in Mind*. Emphasize the broader perspective of capabilities over systems and extend this thinking to all facets of development, including performance, resilience, and risk.

- *Embrace and Understand Uncertainty*. Uncertainty is not to be feared; recognize that

Table 1: Principled Design vs. Designing for Principles		
Design Function or Attribute	**"Principled Design" (Traditional SE)**	**"Designing for Principles" (Propose Framework)**
Technical Performance	Optimized for known functional/performance requirements; best option if current circumstances do not change and assumptions remain valid	Reduced performance relative to current circumstances/assumptions, but generally superior performance for broader range of possibilities; emphasizes non-functional requirements (i.e., "-ilities")
Requirements	Large number, specifying as much as possible, including thresholds of acceptable performance; overarching question is "*Do we have everything we need*?"	Fewer in number, or not used; specify only what matters and prioritize objectives over traditional requirements; overarching question is "*Do we need everything we have*?"
Approach to Uncertainty	Eschews uncertainty and pursues limited risk management that is largely tactical and acquisition-based	Embraces uncertainty and engages in full *uncertainty* management that is more strategic and operationally based
Interfaces	Highly integrated with tight coupling between elements; functionally monolithic; intra-system interfaces are the focus	Modular/interoperable with loose coupling between elements; functionally disaggregated; inter-system interfaces are the focus
Architectural Focus	Typically, on major systems; assumes that if all systems meet their allocated requirements, the enterprise will perform as the sum of its parts	May be at any level, but typically on SoS or the enterprise; assumes emergent behavior and recognizes that certain "-ilities" can generally best (or only) be realized at the enterprise level
Resilience to Known Threats	Moderate; survivability strategies tend to be system-centric, focusing on point-solution defendability	Lower from the perspective of an individual system, but higher from the perspective of the overall capability
Resilience to Unknown Threats	Extremely low (only by pure chance) because approach is inherently threat-centric	Moderate; flexible implementations foster ability to survive broader range of threats, even if not anticipated; approach is *more threat-agnostic*
Summary	Reactive, downplays uncertainty and assumes stability in resources, requirements, and threats; emphasis is on near-term, exquisite technical performance and efficiency	Proactive, embraces uncertainty and expects the unexpected; emphasis is on networked adaptability, capability fluidity, resilience, balanced with enduring "good enough" technical performance

Table 2: Principled Design vs. Designing for Principles (Space System Version)		
Design Element	**Traditional SE "Principled Design"**	**DfP-based Approach "Designing for Principles"**
Architecture	Fundamental architectural characteristics (e.g., constellation size, orbital characteristics, ground station locations) are established as early as possible to constrain design problem; difficult to modify architecture during development and operations.	Fundamental architectural characteristics are not fixed or are determined as late as possible; once established, architecture can be more readily modified/adapted—even after fielding—should circumstances warrant
Space Architecture	Emphasis on small number of large, monolithic satellites working independently; number of satellites fielded is no more than absolute minimum required to meet requirements.	Emphasis on large number of smaller satellites working cooperatively; more satellites are fielded than necessary in order to enhance flexibility and support contingency/reserve capacity.
Ground Architecture	Ground sites are large, but few in number; sites are also fixed in location with ability to support a limited number and/or type of satellites; each site operates independently. Number of ground sites is no more than absolute minimum required to meet requirements.	Ground sites are smaller, but greater in number; sites are either physically mobile, geographically distributed, or functionality distributed in order to support a large number and/or type of satellites; ground sites are interoperable and purposely exceed minimum requirements in order to enhance flexibility and support contingency/reserve capacity.
Satellite	The satellite bus, primary payload, and associated subsystems are tightly integrated and optimized to achieve maximum technical performance. Mass of satellite is minimized through custom interfaces, de-prioritization of SWaP, and minimal propellant.	The satellite bus, primary payload, and associated subsystems are loosely coupled to achieve modularity at acceptable performance. For larger satellites, mass is minimized through reduced technical performance, though offset by modular/standard interfaces, prioritization of SWaP, and additional propellant.
Ground Site	Each site is highly capable and is optimized for a specific mission; operations personnel are also focused on that mission. Facilities focus on minimizing physical footprint and using space as efficiently as possible.	Each site is less capable, but is able to support multiple missions; operations personnel are trained to support multiple missions as well. Facilities are less concerned about physical footprint or using space efficiently; more emphasis on dual use and ready expansion.
Common Architecture	Communication pathways are highly reliable and robust (to known threats), but fewer in number and type and generally dedicated to a single mission; bandwidth is based on current capacity needs.	Individual communication pathways may be less reliable, but are greater in number and type (e.g., crosslinks, dual-pathing) and mission applicability; bandwidth capacity is greater than currently needed.

nothing is as certain as we think it is and that to pretend otherwise is myopic and counterproductive.

These DfP pillars collectively embody this central assertion: Genuine resiliency does not come from picking one optimal future architecture now; it comes from the ability to continually adapt the architecture across a range of future needs and threats, many of which are unpredictable and unknowable. Consequently, the more complexity and uncertainty that we are faced with, the more compelling the DfP approach becomes. Given the enormous complexity of the DOD space enterprise—coupled with volatile adversary behaviors, dynamic threats, and unpredictable technologies—the national space enterprise is an ideal candidate for DfP.

Glossary of Terms

This paper uses jargon that is likely to be familiar to systems engineers as well as those in the space community. To assist understanding for those readers not as well versed in this jargon, an explanation of some of these terms is provided below.

DOD Space Enterprise. Describes the broad scope of space capabilities—and supporting activities and resulting architecture—that are the responsibility of the DOD. Previously, it was difficult to correlate these responsibilities to a single DOD organization, but with the recent establishment of the United States Space Force, this scope is now largely the purview of that Service.

Emergence. Salient characteristic of complex systems. It occurs when a system exhibits characteristics not seen in its constituent elements, usually due to difficult-to-quantify/difficult-to-characterize interactions. The homicidal behavior of HAL 9000 in the book and movie *2001: A Space Odyssey* is a good example of emergent behavior.

Engineering Tradespace. In engineering parlance, this is the range of all possible design options. In general, the larger the tradespace, the more implementation options are available.

Functional Requirements. These describe—typically qualitatively—the activities (or functions) that need to be performed in operations. Functional requirements pertain to what the system must do and are typically achieved as part of specific, deliberate design decisions. A functional requirement for a Jedi's lightsaber would be that it can cut through virtually any substance.

Functional Availability. Defined as the probability that a satellite constellation will meet the system KPPs given the current state of the constellation and the planned replenishment schedule.

Kessler Syndrome. This is a scenario in which collisions between objects in low Earth orbit create more debris, which increases the likelihood of even more collisions, i.e., a cascading effect. The 2013 movie *Gravity* may have had its technical (and plot) shortcomings, but it effectively conveys the basic nightmare of the Kessler syndrome.

Key Performance Parameter (KPP). Represents the topmost requirement in the DOD requirements process, which are considered the most critical/ essential for successful mission accomplishment.

Systems Engineering (SE). Definitions of SE abound. Here's one from the DOD (per the Defense Acquisition Guidebook): "methodical and disciplined approach for the specification, design, development, realization, technical management, operations and retirement of a system." In this paper, the term "Principled Design" is used synonymously with SE.

Non-Functional Requirements (NFRs). These define general system attributes, sometimes referred to as "system-wide requirements," but more often as

"quality attributes." Examples include flexibility, agility, versatility, and resilience (note prevalence of common suffixes; this is why NFRs are also referred to as the "-ilities"). NFRs are typically achieved as part of an architectural/design implementation. Potential NFRs for a Jedi's lightsaber could include durability and portability.

Requirements Creep. Pejorative term used to describe the situation where baselined requirements tend to unexpectedly change over time, generally to be more stringent/demanding. Requirements creep is generally regarded as the bane of a structured requirements management process.

SWaP: Acronym that stands for "size, weight, and power." It is a shorthand description for key form factor elements that are allocated for potential future use. In the space community, the term is most frequently used in relation to satellites where size, weight, and power are especially scarce.

Acknowledgments

I would like to thank the following colleagues for their thoughtful and valuable contributions to this paper: Ryan Noguchi, Angie Bukley, Allison Taylor, and Albert Hoheb. And a special thanks to my daughter, Mckenna Ryan, for her eagle-eyed proofreading skills.

References

[1] Defense Acquisition University, *Defense Acquisition Guidebook*, DAU (https://www.dau.edu/tools/dag).

[2] Armin Schulz and Ernst Fricke, "Design for Changeability: Principles to Enable Changes in Systems Throughout Their Entire Lifecycle," *Systems Engineering,* 8(4), 2005, pp. 342-359.

[3] Mary Kasarda, Janis Terpenny, Daniel Inman, Karl Precoda, John Jelesko, Asli Sahin and Jaeil Park, "Design for Adaptability (DFAD)--A New Concept for Achieving Sustainable Design," *Robotics and Computer-Integrated Manufacturing,* 23(6), 2007, pp. 727-734.

[4] Ibid.

[5] Armin Schulz and Ernst Fricke, "Design for Changeability: Principles to Enable Changes in Systems Throughout Their Entire Lifecycle," *Systems Engineering,* 8(4), 2005, pp. 342-359.

[6] Systems Engineering Research Center, *Valuing Flexibility – Phase II Technical Report,* 2012.

[7] Erin Ryan, *Cost-Based Decision Model for Valuing System Design Options*, PhD Dissertation, Air Force Institute of Technology, 2012.

[8] Mohammad Jamshidi, *System of Systems Engineering,* John Wiley & Sons, 2008.

[9] Colin Clark, "Air Force Acquisition Test: Missile Warning Sats in 5 Years to Orbit?!" *Breaking Defense*, 2018 (https://breakingdefense.com/2018/04/air-force-acquisition-test-missile-warning-sats-in-5-years-to-orbit/).

[10] Sandra Erwin, "The end of SBIRS: Air Force says it's time to move on," *SpaceNews*, 2018, (https://spacenews.com/the-end-of-sbirs-air-force-says-its-time-to-move-on/).

[11] Stanley McChrystal, *Team of Teams: New Rules of Engagement for a Complex World,* Portfolio/Penguin. 2015.

[12] Wikipedia, "*Internet Backbone*," 2020 (https://en.wikipedia.org/wiki/Internet_backbone).

[13] Department of Defense, *DoD Directive 3100.10: Space Policy*, 2016.

[14] Chairman of the Joint Chiefs of Staff, *Joint Capabilities Integration and Development,* 2015.

[15] Brian Walker and David Salt, *Resilience Thinking: Sustaining Ecosystems and People in a Changing World,* Island Press, 2006.

[16] John Gershenson, Gajendra Prasad, and Ying Zhang, "Product Modularity: Measures and Design Methods," *Journal of Engineering Design,* 15(1), 2003, pp. 33-51.

[17] Roshanak Nilchiani and Daniel Hastings, "Measuring the Value of Flexibility in Space Systems: A Six-Element Framework," *Systems Engineering,* 10(1), 2006, pp. 26-44.

[18] Stefan Thomke, "The Role of Flexibility in the Development of New Products: An Empirical Study," *Research Policy,* 26(1), 1997, pp. 105-119.

[19] Owen Brown and Paul Eremenko, "Application of Value-Centric Design to Space Architectures: The Case of Fractionated Spacecraft," American Institute of Aeronautics and Astronautics, 2008.

[20] Erin Ryan, *Cost-Based Decision Model for Valuing System Design Options*, PhD Dissertation, Air Force Institute of Technology, 2012.

[21] SpaceNews Staff, *For Air Force Space Planners, Diversity is its Own Deterrent*. *SpaceNews*, 2015 (https://spacenews.com/for-air-force-space-planners-diversity-is-its-own-deterrent/).

[22] Jonathan Baron, *Thinking and Deciding (4th ed.),*Cambridge University Press, 2007.

[23] Stanley McChrystal, *Team of Teams: New Rules of Engagement for a Complex World,* Portfolio/Penguin, 2015.

[24] Julie Shortridge, Terje Aven and Seth Guikema, "Risk Assessment Under Deep Uncertainty: A Methodological Comparison," *Reliability Engineering & System Safety*, 159 (March 2017), pp. 12-23.