31st Annual INCOSE international symposium
virtual event
July 17 - 22, 2021

Security as a Functional Requirement in the Future of Systems Engineering

Keith D. Willett
U.S. Department of Defense
Keith.Willett@incose.org

Abstract. The *Future of Systems Engineering (FuSE) Security* project is an INCOSE-led multi-organizational collaboration exploring a roadmap of foundational topics for integrating security into systems engineering thinking and practices. *Security as a Functional Requirement* elaborates on one topic to help better elaborate security into systems thinking and integrate security into systems engineering processes. Details include integration of security into existing input-output diagrams, convey standard security concepts with which to engage stakeholders, and establish security as an infinite game that requires security be an active consideration throughout the system lifecycle including operations.

Introduction

FuSE[1] is an INCOSE-led multi-organizational collaboration with many initiatives including artificial intelligence, autonomous physical systems, complexity science, agility, and security. Each initiative identifies and elaborates on a series of foundational topics with intent to advance state-of-the-art systems engineering (SE) thinking and practice. *Security as a Functional Requirement* is one foundational topic under the *FuSE System Security* initiative; Table 1 provides a synopsis of this topic.

Table 1: Synopsis

Problem	As a non-functional requirement, systems security does not get systems engineering prime attention.
Need	Systems engineering responsibility for the security of systems.
Barriers	System engineering practice codifies security as a non-functional requirement.
Intent	Establish security as a functional requirement; inherently raise importance.
Value	Integrate security throughout the systems engineering lifecycle processes.
Metrics	Presence of effective functional security requirements.
Notions	Common Criteria. Open Security Architecture. OMG Unified Architecture Framework. Industrial Internet of Things Security Framework.

[1] https://www.incose.org/about-systems-engineering/fuse

Security's Lament

Security's lament, "They call me 'non-functional' and try to make me feel good by labeling me a *quality attribute*. I'm just a forgotten -ility… tolerated, not wanted… an expense, not investment… a constraint, not an enabler… but I'm better than that, and I'm going to prove it! I am quality, and more… I'm *functional*!"

A functional requirement is a qualitative description of an *activity to perform* or *purpose to achieve*. An 'activity to perform' is some action (*behavior*) and a 'purpose to achieve' is some desired result (*value-delivery*). Security has functions (*it does stuff*) and has many purposes to achieve (*it accomplishes stuff*) all to ensure the system continues to provide value-delivery while facing adversity.

A *system* is a subset of social (people), technical (computers and other tools), process (methods, workflows, operations), and environment (system-of-systems, ecosystem). We can have natural systems or engineered systems. Systems can be tangible or intangible, or abstract notions, e.g., mathematical system. We can have composite systems like socio-technical or cyber-physical systems.

The primary goal of any system is *value-delivery*. Some systems are *expendable* where if something goes wrong, we let them go and replace them. Some systems are *adaptable* where if something goes wrong, we want them to continue to produce desired results; some systems *sustain value-delivery while facing adversity*. Adversity imposes potential loss. Determining the degree acceptable loss is an expression of stakeholder *risk tolerance*; risk tolerance drives the *need for security* (Willett 2016).

Without security, the system's extended viability and relevance are left to chance in a nominal world and open to malicious attack in an adverse world. The INCOSE FuSE topic for *Security as a Functional Requirement* explores a standard language and standard approach to elicit stakeholder needs that lead to security and the development of system security functional requirements that are an inherent part of the system lifecycle including design, development, and operations.

Key Artifacts and Security Dynamics

Stakeholders want something from their system of interest; some desired result, some *value-delivery*. Systems provide value-delivery at a cost; resource consumption, time, money, energy, and raw material. The cost of value-delivery while encountering adversity is more than cost under nominal conditions. One point to discuss with stakeholders is the maximum cost they are willing to spend for risk management, safety, agility, dependability, sustainability, reliability, and security.

Part of that discussion is about risk. Risk includes *probability of occurrence* (likelihood) and *degree of impact*, *effect*, and *consequence*. Impact is direct contact, effect is the first-order result, and consequence is the n-order result. For example, in a game of billiards, the cue stick strikes the cue ball (impact), which knocks the eight-ball into the side pocket (effect) thus winning the game (consequence). A catastrophic example, a flock of birds strike both airplane engines while in flight (impact), which causes engine failure (effect) thus resulting in a forced water landing and a lot of people postponing travel plans (consequences).

The risk discussion should deemphasize the potential causes and emphasize the type and degree of potential loss. A system may experience loss in the form of destruction, degradation, disruption, denial, disclosure, deception, disorder, etc. The discussion results in mutual understanding between stakeholder and systems engineer that includes stakeholder tolerance for loss including *direct loss* to the system and especially *indirect loss* resulting from some loss to the system. Having the payroll system go down is less a concern about the system itself (direct) than to employees not showing up

to work because they did not receive a paycheck (indirect). The systems engineer captures the results of this discussion in a risk tolerance document.

Risk Tolerance. *Risk Tolerance* is a formal document expressing the capacity to endure exposure to loss. This artifact starts with risk and risk management including the negative side of risk (loss) and the positive side of risk (opportunity). Details include direct risk to the system and indirect risk resulting from the system. Risk Tolerance is a key input to the risk posture.

Risk Posture. The *Risk Posture* is an intentionally assumed position to address all risk in terms of accept, share, transfer, or mitigate. Influences on the Risk Posture include the ecosystem, value-chain, supply-chain, and potential shocks in the form of threats and vulnerabilities; plus, stakeholder risk tolerance.

Desired Security Posture. The *Desired Security Posture* is the intentionally assumed position to enforce the Risk Posture. The Desired Security Posture defines the desired safeguards (products and services) to sustain the Risk Posture. For example, safeguards for risk sharing include insurance to spread out the risk. Safeguards for risk transfer include the use of external services; e.g., managed services. Mitigations are in the form of system features, functions; and external products and services on which the system relies (enablers).

Continual Monitoring Plan. The Continual Monitoring Plan includes that which to monitor and the trigger events for taking snapshots. That which to monitor spans people, process, technology, and environment; e.g., knowledge (education), skills (training, certification), tactics, techniques, procedures, inventory (what we should see or expect to see), and perimeter details. Trigger events are conditions that drive taking a snapshot; e.g., time/schedule, anomalous event, new compliance driver.

Actual Security Posture. The Actual Security Posture is the current state of system security. Continual monitoring provides snapshots of the Actual Security Posture. A gap analysis shows the differences between actual and desired resulting in inputs to a gap closure plan.

Gap Closure Plan. A Gap Closure Plan lays out how to move from actual to desired. Depending on resource constraints, implementation of this plan may take minutes or years; e.g., implementing all known security controls exceeds any single year budget thus requiring a multi-year gap closure plan.

The *Risk Posture* is dynamic; stakeholder risk tolerance, ecosystem, and the value-chain constantly change. This implies a dynamic *Desired Security Posture*; i.e., that which security attempts to achieve is constantly changing. This implies the need for *agile security* throughout the system lifecycle to sustain an appropriate set of safeguards. Figure 1 shows the systems dynamics *escalation archetype* where two balancing loops interact in such a way as to create a reinforcing loop. Action by A (new safeguard) results in a more secure A. Action by B (adapt adversary tactic) results in breaching A's new safeguard, which prompts a new cycle in continual escalation.

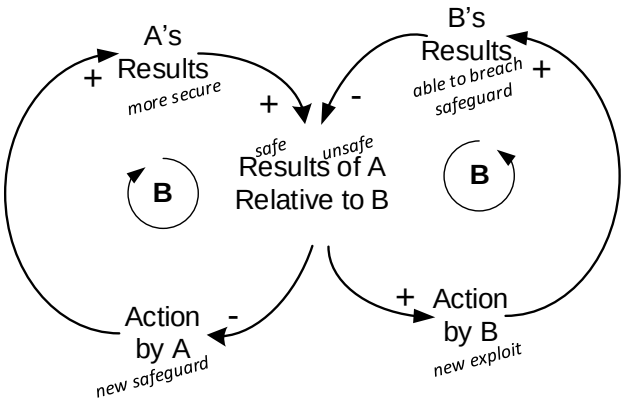

Figure 1: Systems Dynamics Escalation Archetype

Figure 2 shows the systems dynamics *oscillation archetype* where *feedback delays* cause over and under compensation. Consider taking a shower in a bath with old plumbing. You turn on the hot water and it comes out cold. The delay in hot water flow prompts turning up the hot water volume. Water coming out too hot prompts turning down the volume. The delay in temperature drop prompts turning the hot water down even further. Water coming out too cold prompts turning the volume up again. And so, we proceed with too hot and too cold in decreasing amplitudes until settling on an acceptable middle.

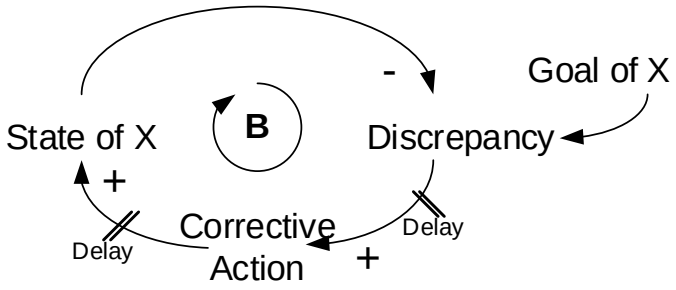

Figure 2: Systems Dynamics Oscillation Archetype

We can adapt the oscillation archetype for security (Figure 3). Security as an oscillation includes all the key artifacts and their interrelations; a change in one has cascading effects on the others.

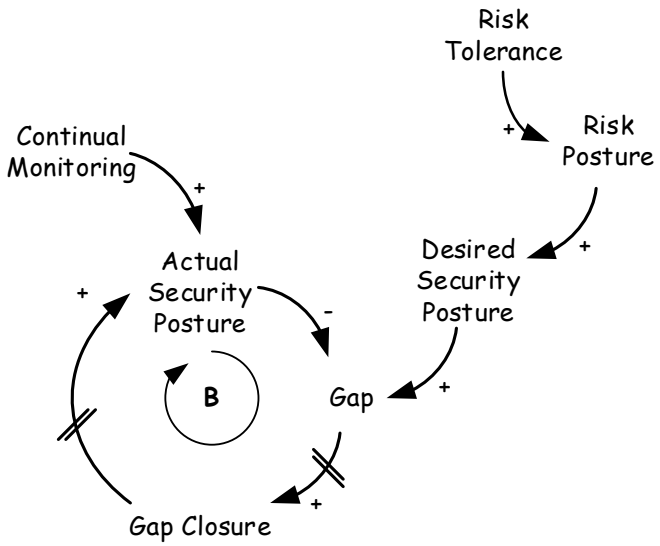

Figure 3: Security as Systems Dynamics Oscillation

Figure 4 shows a graph of security as a system dynamics oscillation archetype. The desired security posture provides a non-static target with upper and lower bounds of acceptable deviation as captured in the risk tolerance artifact. Too much security is a waste of resources and too little security is an unacceptable level of risk. When combining these security dynamics with the escalation archetype we see security as an infinite game with over and under compensating around non-static targets.

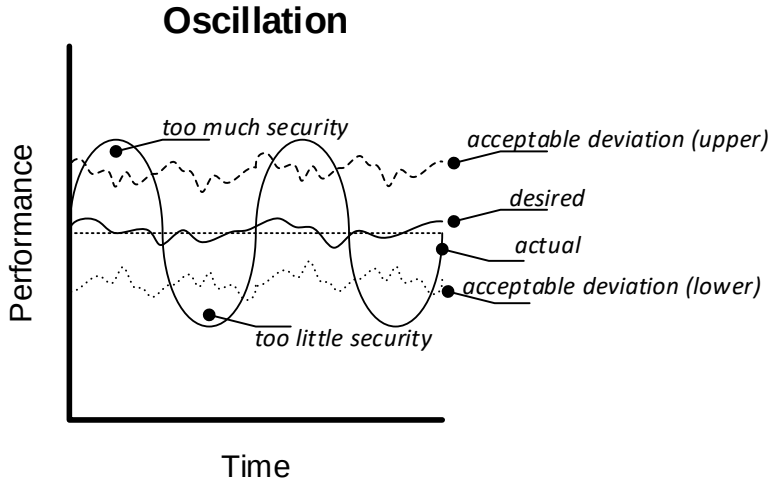

Figure 4: Measuring Security as an Ongoing Oscillation

As we proceed to codify security as a functional requirement, we will explore the integration of security into the systems engineering lifecycle processes with intent to realize the iterative process in Figure 3 and Figure 4 as an inherent part of engineered systems.

Codifying Security as a Functional Requirement – First Steps

Security is an infinite game. Every new technology increases the attack surface. Every advance in safeguards meets an advance in adversary exploits. That which constitutes being secure (*desired security posture*) is non-static driven by continual changes in stakeholder needs, risk tolerance, ecosystem conditions, value-chain, supply-chain, and vulnerabilities. Sustaining a level of acceptable security is ongoing throughout every phase of the system lifecycle including operations from both outside the system (stakeholders, architects, engineers, operators, users/beneficiaries) and within the system; i.e., *agile security* as a system function.

Systems engineering traditionally focuses on *design to build*. If we accept security as an infinite game, systems engineering focus expands to include *design to operate*; i.e., systems engineers design the system, design the operations (workflow), and design the role, fit, function, and impact of the system within operations including to some degree continual dynamic adaptation of that system to remain viable and relevant while facing adversity.

The *INCOSE Systems Engineering Handbook v4* (INCOSE SEHB) provides four *process groups* each with multiple processes to support systems engineering (SE): *technical* (14 processes), *technical management* (8 processes), *agreement* (2 processes), and *organizational project-enabling* (6 processes). Each process has an input-process-output (IPO) diagram (Figure 5). The integration of security into the systems engineering discipline and the expression of security as a functional requirement begins with the introduction of security related concepts in the systems engineering lifecycle processes.

If we explicitly introduce security in the early processes, we then see the remainder of the processes naturally absorb security; i.e., security inherently becomes part of the system lifecycle.

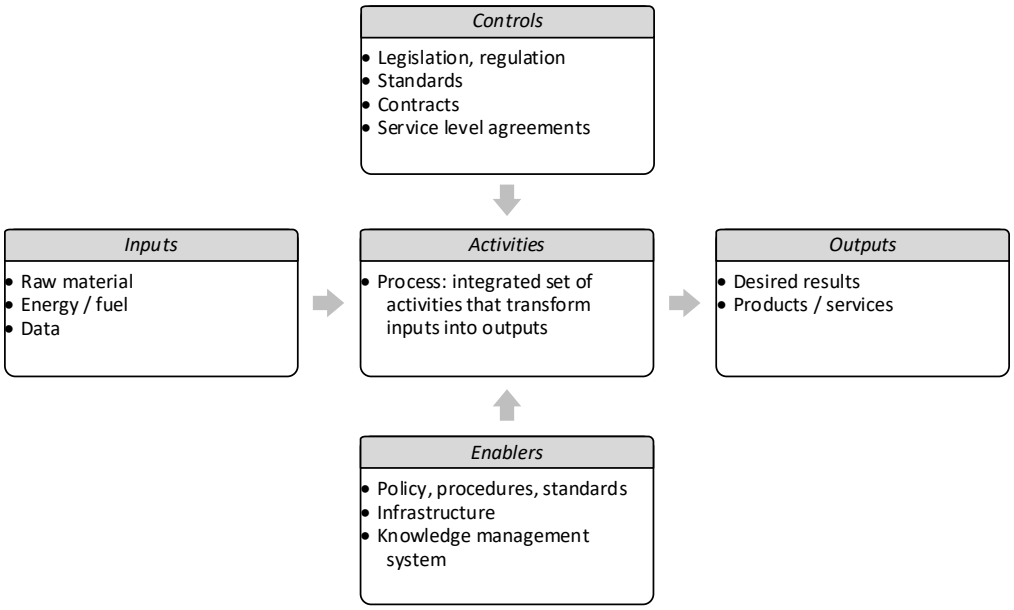

Figure 5: Example IPO Diagram (Derived from INCOSE SEHB)

The objective in the first steps toward elevating security as a functional requirement include the integration of the key artifacts into the systems engineering lifecycle: *Risk Tolerance*, *Risk Posture*, *Continual Monitoring Plan*, *Desired Security Posture*, *Actual Security Posture*, *Gap Analysis*, and *Gap Closure Plan*. *Risk Tolerance*, *Risk Posture*, and *Desired Security Posture* are part of system design and development. All these artifacts are part of operations (workflow). System design and development provides initial versions; operations are iterative with dynamic interactions (Figure 3) for continual adaptation of all artifacts that subsequently drive changes in the system, system security, and in operational workflow.

The INCOSE SEHB Figure 4.2 provides an IPO diagram for 'business and mission analysis'. Requirements definition begins with the expression of mission needs[i]. We can add *strategic risk* to inputs and derive risk in part from dependencies within the ecosystem and supply-chain. We integrate strategic risk into activities; e.g., 'define the problem or opportunity space' includes the *negative side of risk* (loss) and the *positive side of risk* (opportunity). Integrate risk into existing outputs and add an explicit output for *stakeholder risk tolerance*. Figure 6 provides additions to the *business and mission analysis* IPO diagram.

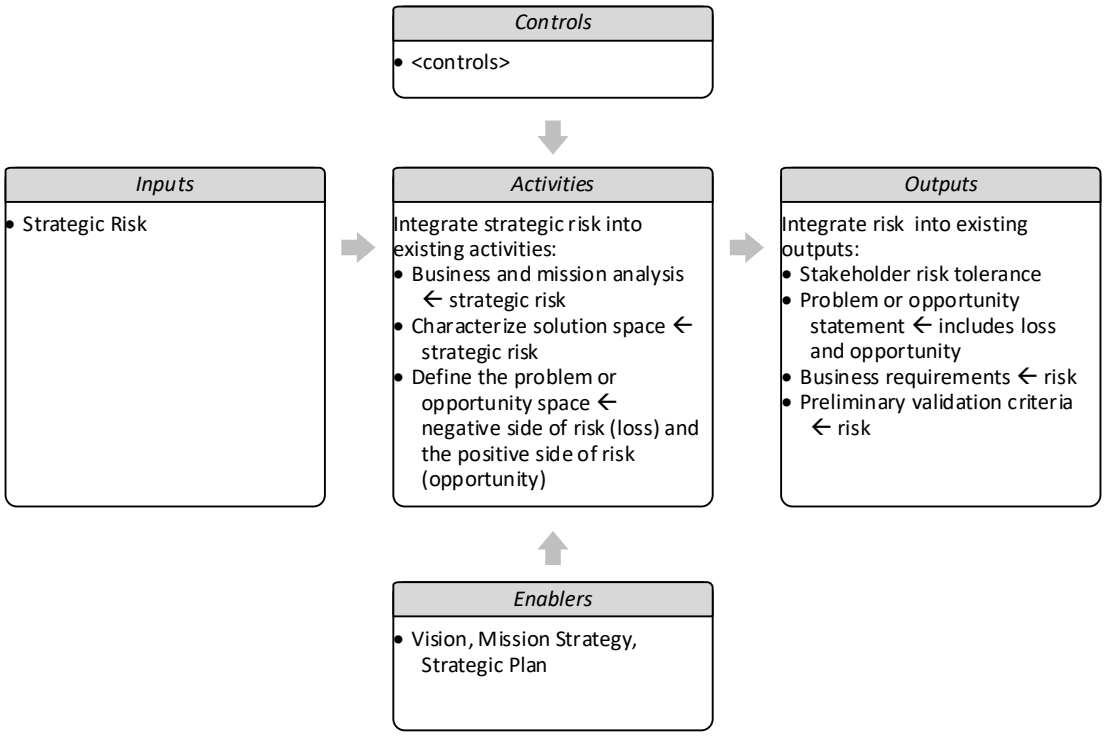

Figure 6: Additions to Business and Mission Analysis IPO Diagram

INCOSE SEHB Figure 4.4 provides an IPO diagram for 'stakeholder needs and requirements definition process'. To integrate security, we can add: ***stakeholder risk tolerance*** to inputs, *translate risk tolerance into a* ***risk posture*** in processes, and *risk posture* to outputs (Figure 7). And thus begins the integration of key artifacts such that security becomes not only an inherent part of the systems engineering discipline but is an explicit functional requirement and subsequently part of ongoing system operations; continual dynamic adaptation of the system and system security.

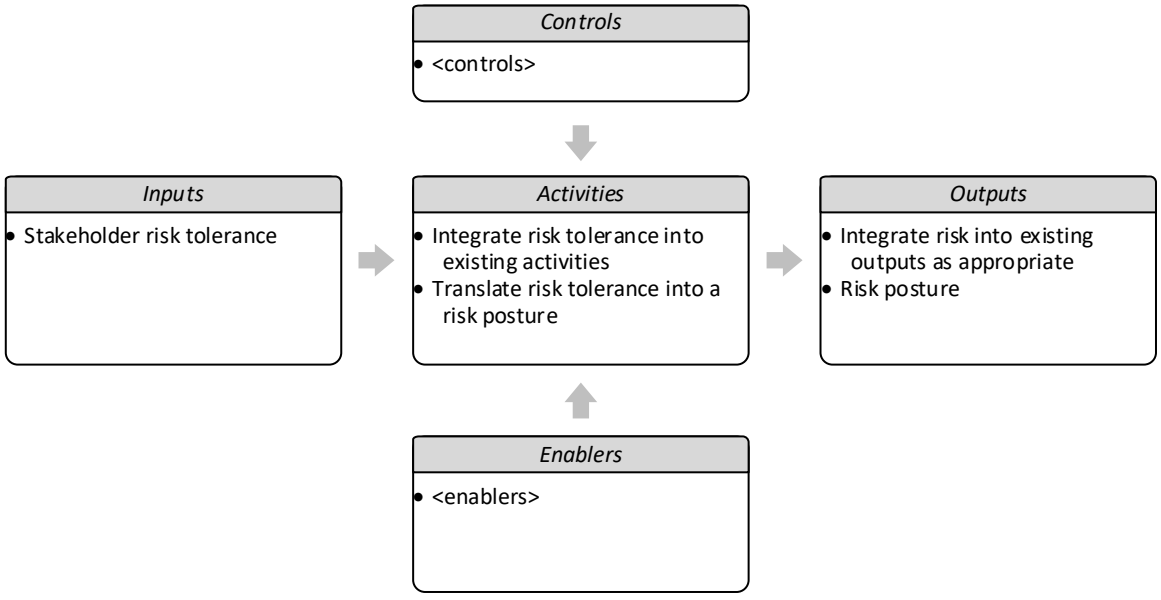

Figure 7: Additions to the Stakeholder Needs and Requirements Definition Process IPO Diagram

The INCOSE SEHB Figure 4.5 provides an IPO diagram for 'system requirements definition process'. We may add *risk posture* to inputs. We integrate use of the risk posture in all the existing activities. We include aspects of ***desired security posture*** in 'system requirements definition strategy', 'system function definition', 'system requirements', 'system functional interface identification', 'verification criteria', 'system requirements traceability', 'requirements verification and traceability matrix (RVTM), and 'system requirements definition record' as part of outputs (Figure 8). At this point, security embeds in artifacts that cascade through the remainder of the systems engineering process.

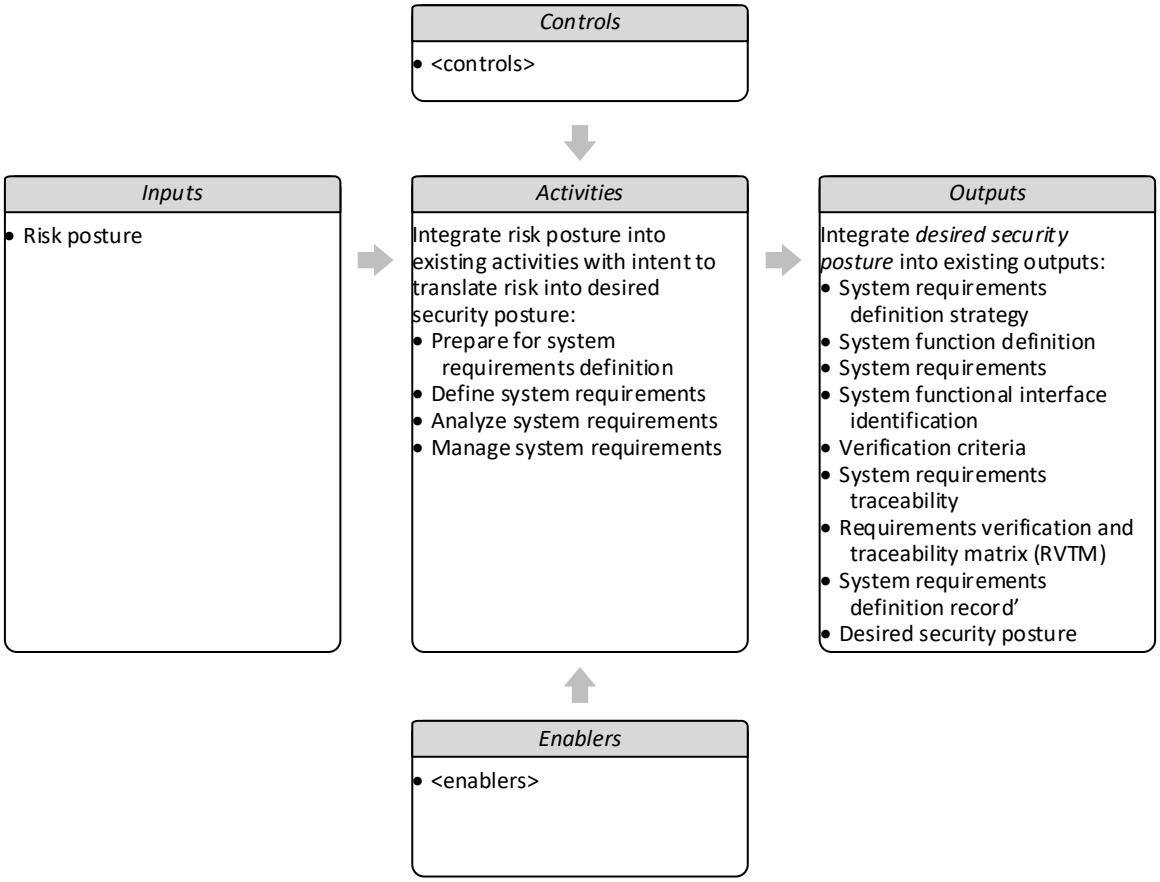

Figure 8: Additions to the System Requirements Definition Process IPO Diagram

The INCOSE SEHB Figure 4.6 provides an IPO diagram for the 'architecture definition process'. At this point and throughout the remainder of the processes, the input artifacts inherently include the *desired security posture*. Additionally, the desired security posture artifact provides a starting point for continual monitoring.

Operations includes a ***continual monitoring plan*** for assessment of the ***actual security posture*** versus *desired security posture*. Formalizing continual monitoring is important to baseline that which we should monitor, what *should* be there (input from inventory management) and comparing what is actually there (what monitoring detects). If something expected is not showing up, prompt action to find out why; is it shutdown, inoperative, off network, or missing. If something unexpected shows up, prompt action to find out why; missed in inventory, rogue device, or personal device inadvertently connected to the system.

As we learn to integrate systems engineering artifacts as part of operations in the form of codifying DevOps, emergence, agility, or continual dynamic adaptation, security is inherent and exists in relevant forms that include functional requirements.

Setting Up Operational Agility – Continual System Change

In agile-systems, functional requirements carry through to operations with expectation of continual system change. The drivers of change in operations are similar to those in design/development. Changes in the ecosystem, stakeholder needs, and risk tolerance drive change to risk posture which prompts change to desired security posture (Figure 3). Change is on both the strategic and tactical level with mutual influences by both. Figure 9 provides two interlocking cycles for strategy and tactics in an infinite game within which security is one of many domains.

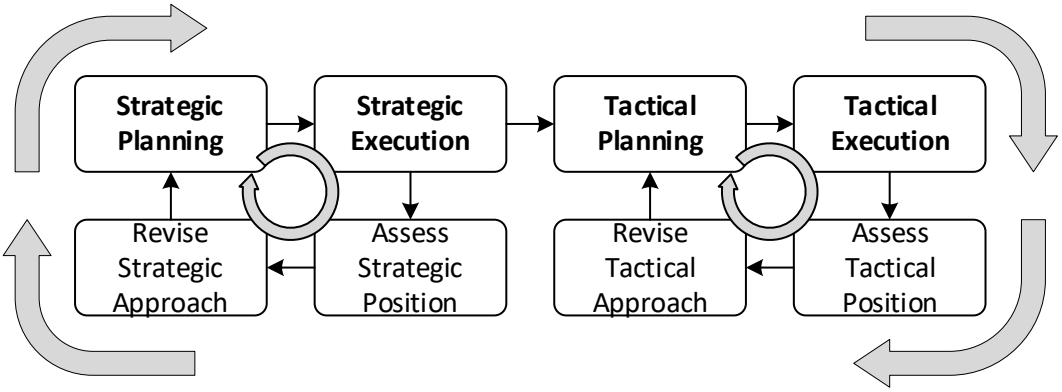

Figure 9: Two Interlocking Cycles to Help Represent Operational Agility

Integrating security into systems engineering is to produce and sustain inherently more secure systems. We manage security as an infinite game. Following the adage of *you can't manage what you can't measure*, let's explore a structure for measurements that supports operational agility with focus on security (Willett 2020):

- *Goals* capture what we're after (desired value). Goals are non-static; they will change. Risk tolerance and risk posture are the goals for security.
- *Strategies* support goals. Three general strategies: *function-driven* (what the system does to provide value-delivery), *loss-driven* (negative side of risk), and opportunity-driven (positive side of risk). We can focus on security as one domain under loss driven.
- *Objectives* are measurable steps within strategies. Objectives may or may not be static. Some will fluctuate as the strategies adjust to address changes in goals (the infinite game). Desired security posture capture security objectives.
- *Methods* are the tactics, techniques, and procedures to achieve objectives. Methods invoke solutions. Continual monitoring is a method for sustaining an acceptable degree of security.
- *Solutions* are in the form of safeguards (features, functions, tools, services)
- *Measures* in the form of X discern Y with respect to some aspect of solution or method and provide insight into the degree to which we are achieving the objective.
    - X ∈ (binary, degree, statistics, probability, time, distance, quantity, accuracy, etc.)
    - Y ∈ (structure, state, behavior, function, functional exchange, contents, resources, environment, value-delivery, etc.)

Table 2: Measurement Structure

Goal	**Strategy**	**Objective**	**Method**	**Solution**	**Measure**
<goal>	<supporting strategy>	<measurable step within strategy>	<TTPs to achieve objectives>	<tools; products, services>	<status, state>

Stakeholder Needs-Value (Goals)

Context Matters. Context is that which facilitates the expression of meaning and value. Context includes social/cultural (who), technical (what), spatial (where), temporal (when), and behavior (how), and desire (why). Context frames need, goals, risk, and risk tolerance.

Let's explore stakeholder needs that lead to security with intent to identify a standard set of needs (beyond the scope of this paper) leading to *security* We cannot provide a rote checklist applicable to all circumstances, so we select from a non-static standard set of needs that which is most appropriate to the context. The following goal statements are in the form of maximize, minimize, and optimize that inherently force tradeoffs among tensions in costs, benefits, performance, and future options:

- Optimize value-delivery: produce desired results
- Optimize risk (risk neutral)
- Minimize risk (risk averse)
- Maximize risk (risk seeking)
- Minimize threats
- Minimize threat efficacy
- Minimize vulnerabilities
- Minimize vulnerability efficacy
- Minimize negative impact
- Minimize negative effect
- Minimize negative consequences
- Maximize safeguard efficacy
- Maximize positive impact
- Maximize positive effect
- Maximize positive consequences
- Maximize learning (knowledge)
- Maximize profit
- Minimize cost

Needs-Value. Let's focus on ten security principles (Willett 2008) to derive need-value relationships; i.e., we need X to provide the value of Y:

- We need **confidentiality** to safeguard our secrets.
- We need **integrity** to keep the system whole.
- We need **availability** to ensure the system is ready for use.
- We need **possession** to retain physical control; minimize effects of loss or theft.
- We need **authenticity** to ensure the system is compatible with reality; anti-deception.
- We need **utility** to ensure the system is fit for purpose.
- We need **non-repudiation** to ensure attribution; ensure anti-anonymity.
- We need **privacy** to ensure ability to remain unobserved and ability to be forgotten; ensure anonymity.
- We need **authorized use** to ensure resource control; minimize theft of service or misallocation of resources.
- We need **accountability** to ensure explain-ability; learn and minimize repeating mistakes; minimize malicious anonymity.

Each need expresses *what we want* explicitly avoiding *how to achieve* what we want. Each value expression includes perspectives of value *to* the system, value *from* the system, and may refer to any subset of system characteristics (structure, behavior, content, resources, environment, and value-delivery).

Strategies

Three general strategies are: *function-driven* (behavior to provide value-delivery), *loss-driven* (avoid, withstand, and recover), and *opportunity-driven* (seek gain, continual optimization).

Strategic Planning. Strategic planning looks forward to the goal (to-be), reasons backward to the starting point (as-is) to articulate a set of objectives as measurable steps to achieve the goal; segue into strategic execution for transition.

Strategic Execution. Strategic execution includes four game types: *zero-sum* (win/lose), *positive-sum* (win/win), *negative-sum* (lose/lose), and *infinite*. A system may engage adversaries in a zero-sum game and engage allies/partners in positive-sum games that help win the zero-sum game. A negative-sum game may be necessary to cut losses. Success in an infinite game is to continue playing; there is no win/lose but ongoing meandering in and out of advantage. Infinite game goals include *retain* advantage (resistance) and *regain* advantage (resilience).

Objectives

Tactical planning creates an operational blueprint for the strategic plan with focus on short-term objectives. An approach to tactical planning includes:

1. Identify strategic priorities; priority goals from strategic plan (strategic target).
2. Define *objectives* as measurable steps to achieve or sustain goals.
3. Identify *necessary reality* to achieve objectives (tactical target, to-be). The necessary reality may differ from desired reality; i.e., what we need may not be what we want.
4. Identify *current reality* with respect to achieving objectives (determine actual, as-is).
5. Identify gaps between *current* and *necessary* reality (gap analysis).
6. Develop a resource allocation plan (add | delete | move) to achieve necessary reality (gap closure plan); including resolving the zero-sum game of resource allocation.
7. Research *best practices*; filter into *viable practices* and down select into *adequate practices* for current organizational context. Adequate practice provides higher RoI than best practice.
8. Allocate resources accordingly to position tactical execution to achieve objectives.
9. Iterate periodically (time) or given some other trigger event; e.g., change in goal, external force (adversary activity), internal force (available resources; budget constraints).

Note: there are discrete iterations as well as holistic iteration; e.g., steps 4-7 may occur more often in a field of fast changing technology (cybersecurity).

Tactical planning includes *that which to safeguard*; system characteristics:

- **Structure**: organization of parts; state
    - Boundaries: perimeter, exterior
    - Core: interior
- **Features**: tools to perform functions
- **Behavior**: function, functional exchange (inputs/outputs)
- **Contents**: real (people, cargo), virtual (data)
- **Resources**: inputs; raw material, energy/fuel
- **Environment**: containing whole (system of systems), current order (ecosystem)
- **Value-delivery**: produce desired results; desired impact, effect, consequence

Methods

Tactical Execution. Tactical execution is operations and workflows that include risk management, identifying and implementing system security functional requirements, system performance and performance requirements, and continual monitoring. Methods invoke solutions that provide desired results. **Risk management**. There are many industry examples of risk management frameworks to draw upon: NIST SP 800-137[ii] *Risk Management Framework for Information Systems and Organizations*; ISO[iii] 3100 *Risk Management*; COSO[iv] *Guidance on Enterprise Risk Management*; and the RMA[v] *Enterprise Risk Management Framework*.

System security functional requirements. Since we speak to functional requirements, let's focus on behavior including functions and functional exchanges where the latter includes internal and external communications, interfaces, and communication pathways. Functions take the form of action verbs. A standard set of functional requirements from which we choose those appropriate to context include (incomplete):

- The system shall [maintain | access] a set of user roles.
- The system shall [maintain | access] a set of allowable actions per role.
- The system shall [maintain | access] a set of unique identifications per person or non-person entity attempting to access the system.
- The system shall interoperate with the existing identity management infrastructure.
    - Assumption: the identity management system maintains roles and responsibilities (allowable actions per role).
- The system shall authenticate a claim of identity.
- The system shall authorize a claim of privilege.
- The system shall interoperate with the existing privilege management infrastructure.
- The system shall encrypt X; X $\in$ (data at rest, data in transit, data in use)
- The system shall interoperate with the existing encryption key management infrastructure.
- The system shall support explicit blocking of X (blacklist).
- The system shall support explicit allowing of X (whitelist).
- The system shall safeguard data.
    - The system shall disclose data to only authorized users.
    - The system shall allow data modification only by authorized users.
    - The system shall clear memory upon process termination to remove data in use.
- The system shall provide an Internet homing beacon; *phone home* on a periodic basis.
- The system shall log X [logons | failed logons | Y folder access | Y file access].
- The system shall notify in real-time of X where X is any state, function, functional exchange, etc. outside of specified acceptable parameters / thresholds.
- The system shall backup relevant content and configuration data on periodic basis.
- The system shall interface with a security orchestration engine.
    - The system shall be able to act independently <actions to be defined>.
    - The system shall be responsive to authorized direction from the orchestration engine.

Many of these standard functional requirements imply the need for supporting infrastructure (enablers) that include:

- Identity management
    - Establish, operate, and maintain identities and identity credentials
- Privilege management
    - Establish, operate, and maintain privileges and privilege credentials
        - Explicit allow, explicit deny
        - Default allow, default deny
        - Deny all unless explicitly allowed (don't trust until given a reason to trust)
        - Allow all unless explicitly denied (trust until given a reason not to trust)
        - Block all and allow only explicit (white list)
        - Allow all and block only explicit (black list)
- Encryption key management
    - Establish, operate, and maintain keys
    - Key storage and retrieval (recovery)
    - This relates in part to the principle of *utility* where an encryption key may be lost and the data otherwise not usable without key recovery
- Backup management
    - Establish, operate, and maintain system backups; recoverability.
    - This relates to disaster recovery, continuity of operations, and to ransomware where wiping a system and restoring may be cheaper than paying ransom.
- Physical security
    - Establish, operate, and maintain perimeter safeguards including a subset of campus, building, floor, room, office, and workstation.

All systems need explicit requirements regarding interfaces to supporting security infrastructure; also, stating explicit assumptions for their existence makes sense. Systems engineering planning may capture each of these as an external system with appropriate interfaces.

Note: the security requirements are about *system behavior* and do not include development activities leading to the system; e.g., secure coding practices or supply chain management. Also, not in scope is program risk which is distinct from system risk.

Functional requirements specify what the system shall do or may do. Related performance requirements specify some degree of efficiency. For example, **performance requirements** may include:

- The system shall authenticate users within X milliseconds with a deviation of no more than +/- Y milliseconds.
- The system shall verify actions against a black list and return a block decision within X milliseconds with a deviation of no more than +/- Y milliseconds.
- The system shall verify inputs against a white list and return a block decision within X milliseconds with a deviation of no more than +/- Y milliseconds.
- The system shall provide an option to log all X activities.
- The system shall maintain a log of all X activities in a moving window of Y [hours | days].

Solutions

Solutions provide for how to achieve objectives. The path to solutions includes security controls as abstractions of solutions. Solutions include tools, products, and services each with features and functions that satisfy the functional requirements to some degree. Not all security solutions are necessarily within the system. Many are enablers existing outside the system and provide safeguards that the system inherits from its environment (physical security), containing whole (system of systems), or current order (ecosystem).

Clearly defining objectives helps guide solution selection and configuration. Security dynamics (Figure 3) provides the iterative influences on solution viability and relevance. New need/objectives require variances in the solution space where that variance may be to replace existing with new, add a feature/module, or update/patch.

Measures

Types of measures. A comprehensive description of metrics and measures is outside the scope of this paper; however, the structure from which we discern security requirements and provide security solutions lends itself to capturing security measures. General types of measures include: binary (Y/N, on/off), statistics (identify and explain), probability (predict), ratio, percentage (degree), rate, time, duration, distance, quantity (volume, count), and quality (accuracy, useful, useable, actionable, timely).

What we measure. That which we can measure includes any system characteristic: structure, state, feature, behavior, function, functional exchange, contents (real, virtual), resources (raw material, fuel/energy), environment (interaction, waste), value-delivery, etc.

Measures help validate performance requirements both initially and continually in operations (continual monitoring). We expect and allow some deviation from desired and then define thresholds or trigger points to prompt corrective action when we find the system has too much security (wastes resources) or too little security (higher than tolerable risk) (Figure 3).

Conclusion

As a non-functional requirement, systems security does not get adequate attention in system design, development, and operations. This paper presents system security as a core responsibility for systems engineering and provides a foundation for codifying security as a functional requirement by integrating security into systems engineering lifecycle as part of existing processes. The proposition of security as a functional requirement introduces a set of new artifacts the details of which become part of existing systems engineering activities and outputs early in the lifecycle and thus integrate security as a natural part of performing systems engineering.

The foundation herein establishes security as an infinite game of complex dynamics and includes initial thoughts for a standard set of security functional requirements and performance requirements that will evolve and grow from the experience of many. Notional contributions / explorations for expanding upon security as a functional requirement include the Common Criteria and Open Security Architecture. Next steps are to apply and build on the foundations herein to establish the presence of effective functional requirements in systems development and operations that inherently include security.

References

Common Criteria, https://www.commoncriteriaportal.org/

INCOSE. 2015. *Systems Engineering Handbook: A Guide for System Life Cycle Processes and Activities*, version 4.0. Hoboken, NJ, USA: John Wiley and Sons, Inc, ISBN: 978-1-118-99940-0.

Open Security Architecture, https://www.opensecurityarchitecture.org/cms/

Willett, K.D. 2020. *Systems Engineering the Conditions of the Possibility*, INCOSE International Symposium.

Willett, K.D. 2016. Cybersecurity Decision Patterns as Adaptive Knowledge Encoding in Cybersecurity Operations. Stevens Institute of Technology. ISBN: 978-1-369-57208-7.

Willett, K.D. 2008. Information Assurance Architecture. Auerbach Publishing. ISBN: 978-0849380679.

Biography

Dr. Keith D. Willett is a senior strategist and enterprise security architect for the United States Department of Defense**.** He has a PhD in systems engineering from Stevens Institute of Technology. He is co-chair for the INCOSE working groups on Systems Security Engineering and Agile Systems & Agile SE. Dr. Willett is sole author of *Information Assurance Architecture* and coauthor of three books including *Multisystemic Resilience – Adaptation and Transformation in Contexts of Change* (Oxford Press) February 2021.

---

[i] INCOSE Systems Engineering Handbook v4, p.47
[ii] National Institute of Standards and Technology (NIST) Special Publication (SP)
[iii] International Standards Organization (ISO)
[iv] Committee of Sponsoring Organizations of the Treadway Commission (COSO)
[v] Risk Management Association (RMA)