Tailoring Ground Software Standards Towards Automation and Augmentation

Jesus Rodriguez
Alan Kwan
Software Process, Modeling and Measurement Department
The Aerospace Corporation

February 26, 2025

Software Development Standards for Mission Critical Software

SMC-S-012 and ISO-IEC-IEEE 12207-2017

- SMC-S-012 – Space and Missile Systems Software Development Standard
  - *The SMC/SSC standard for mission critical software*
  - *Prescriptive processes aimed at qualification of software to meet mission critical requirements*

- ISO-IEC-IEEE 12207-2017 Systems and software engineering – Software life cycle processes
  - *A generic software engineering standard (business or mission software)*
    - Specifies the “what”, not the “how”
    - Software life cycle model agnostic

- How do we leverage both standards and the latest software development best practices?
  - *Agile Software Development (e.g., Augmentation)*
  - *DevSecOps (e.g., Automation)*
  - *AI/ML*

IEEE 12207 Standard – Overview

ISO-IEC-IEEE 12207-2017 Systems and software engineering – Software life cycle processes

- A generic software engineering standard
  - *Business or mission software*
- Processes for all software life cycle stages
  - *Concept, development, sustainment, and retirement*
- Comprehensive set of software life cycle process groups
  - *Agreement Processes*
  - *Organizational Project-Enabling Processes*
  - *Technical Management Processes*
  - *Technical Processes*

- We will focus on the Technical Processes with a focus on Augmentation, Automation, and AI\ML

IEEE 12207 Standard – Technical Processes

- Business or Mission Analysis process
- **Stakeholder Needs and Requirements Definition process**
- **System/Software requirements definition process**
- **Architecture Definition process**
- **Design Definition process**
- **System Analysis process**
- **Implementation process**
- **Integration process**
- **Verification process**
- **Transition process**
- **Validation process**
- **Operation process**
- **Maintenance process**
- **Disposal process**

Business or Mission Analysis process

IEEE 12207 Standard – Technical Processes

- Business or Mission Analysis process
  - *Defines the business or mission problem or opportunity, characterizes the solution space, and determines potential solutions.*

- *SMC-S-012 Business or Mission Analysis process*
  - *SMC-S-012 does not address business or mission analysis process*

- *Business or Mission Analysis process in Agile and DevSecOps*
  - *Continuous stakeholder engagement to gather feedback, clarify requirements, and adapt the product vision.*
  - *Business needs are translated into user stories to describe a feature from the user's perspective.*
  - *Product backlog is regularly refined by the product owner, development team and stakeholders to prioritize features based on business value and feasibility.*
  - *Business analysis is an ongoing process where requirements are continuously refined and validated.*

Stakeholder Needs and Requirements Definition Process

IEEE 12207 Standard – Technical Processes

- Stakeholder Needs and Requirements Definition process
  - *Defines the requirements for the software system that provide the capabilities needed by users in a defined environment.*
  - *Operational concept view of requirements.*
  - *Includes identification of critical performance measures, critical requirements, and bidirectional traceability*
  - *Described in technical requirements documents (TRDs)*

- *SMC-S-012 Stakeholder Needs and Requirements Definition process*
  - *Very similar to IEEE 12207*

- *Stakeholder Needs and Requirements Definition process in Agile and DevSecOps*
  - *Agile teams use user stories, which describe a user's need from their perspective, to capture stakeholder requirement in a concise way.*

System/Software Requirements Definition Process (1 of 2)

IEEE 12207 Standard – Technical Processes

- System/Software Requirements Definition process
  - *Transforms the stakeholder or user requirements into technical, software requirements*
    - *Attributes: necessary, implementation-free, unambiguous, complete, singular, feasible, traceable, verifiable, and bounded*
  - *Described as use cases, features, user stories, or scenarios*
  - *Includes interface, data, and database requirements*
  - *Includes verification methods*
  - *Includes constraints, but suggests not to imply any specific implementation*
  - *Includes identification of critical performance measures, critical requirements, and bidirectional traceability*
- SMC-S-012 Requirements Definition process
  - *Very similar to IEEE 12207*
  - *Government constraints often require specific implementations*

System/Software Requirements Definition Process (2 of 2)

IEEE 12207 Standard – Technical Processes

- Requirements Definition process in Agile and DevSecOps
  - *Agile encourages an iterative process in defining requirements*
    - Challenge with end user involvement on defense systems
  - *Requirements can change with any software development life cycle model (including waterfall)*
  - *Requirements for defense systems tend to be well defined up front (not as much need for change)*
  - *Requirements should be prioritized to enable incremental and iterative development, use of CI/CD pipelines, early integration and risk reduction*

Architecture Definition Process

IEEE 12207 Standard – Technical Processes

- Architecture Definition process
  - *Defines the software architecture (models and views), assesses and manages the architecture*
  - *Goal for an architecture that is as design-agnostic as possible to allow for maximum design flexibility*
  - *Defines the elements or modules of the software and relationships between them to meet the requirements*

- SMC-S-012 Architecture Definition process
  - *Not as detailed as IEEE 12207 in the main architecture definition section, but more detailed by requiring the use of Software Architecture Description (SAD) template*
  - *Requires using modular open systems architecture (MOSA)*
  - *Requires consideration for unit integration, HW/SW integration, and system integration*
- Architecture Definition process in Agile and DevSecOps
  - *Encourages iterative or incremental development of the software architecture, however, doesn't restrict a complete architecture up front*
    - However, changes to the architecture late in the software life cycle should be avoided
  - *Define enough of the architecture to enable implementation of the highest priority requirements through a CI/CD pipeline*
  - *Use of open, industry standards interfaces (for cyber and use of CI/CD toolsets)*

Design Definition Process

IEEE 12207 Standard – Technical Processes

- Design Definition process
  - *Detailed design of the software elements, interfaces, and databases*
  - *Performed iteratively and incrementally with requirements definition and architecture definition*
- SMC-S-012 Design Definition process
  - *Coupled with architecture definition*
  - *Not as detailed as IEEE 12207, but more detailed by requiring the use of Software Design Description (SDD) template*
- Design Definition process in Agile and DevSecOps
  - *Performed concurrently with implementation*
    - E.g., Design what is needed for the current sprint
  - *Use of automated tests and CI/CD to catch interface incompatibilities for design changes*
  - *Use of DevSecOp tools can drive software design pattern choices*
    - E.g., Modeling & Simulation tools with GUI – separate GUI layer from algorithm logic to unit test each separately and identify where errors occur immediately

System Analysis Process

IEEE 12207 Standard – Technical Processes

- System Analysis process
  - *Provides data and information for technical understanding to aid decision-making across the life cycle*
  - *Used for analytical needs concerning operational concepts, determination of requirement values, resolution of requirements conflicts, assessment of alternative architectures or system elements, and evaluation of engineering strategies (integration, verification, validation, and maintenance).*
- SMC-S-012 System Definition process
  - *Coupled with architecture definition*
  - *Not as detailed as IEEE 12207, but more detailed by requiring the use of Software Design Description (SDD) template*

- System Analysis process in Agile and DevSecOps
  - *Performed concurrently with implementation*
    - E.g., Design what is needed for the current sprint
  - *Add tactics (e.g., logging) to check component health (heartbeat), functional correctness, and performance*

Implementation Process

IEEE 12207 Standard – Technical Processes

- Implementation process
  - *Develop the software system in accordance with its architecture, design, and interface definitions to meet its requirements*
  - *IEEE 12207 strives for being less prescriptive, but includes strategies like test driven development, code coverage, other unit testing strategies (e.g. use of simulators)*
  - *Suggests being done concurrently with integration and along with Verification (which identifies anomalies (errors, defects, faults) and Validation processes*
  - *Encourages the use of automated tests (doesn't suggest how)*

- SMC-S-012 Implementation process
  - *Combines implementation process with unit testing*
  - *Emphasis is on unit testing, including nominal and off-nominal testing*

- Implementation process in Agile and DevSecOps and use of AI/ML
  - *Encourages the use of continuous integration and automated unit testing*
  - *Use of AI/ML can accelerate writing source code and unit tests (e.g., use of Copilot)*

Integration Process

IEEE 12207 Standard – Technical Processes

- Integration process
  - *Includes the integration of systems or system elements and interfaces*
  - *Coordinates with Architecture Definition and Design Definition processes to check that interface definitions are adequate and that they consider the integration needs.*
  - *Software system integration iteratively combines implemented software system elements to form complete or partial system configurations to build a product or service.*
  - *Software integration is performed daily or continuously during development and maintenance stages, using automated tools.*
- *SMC-S-012 Integration process*
  - *Focuses on integration with software units, and the software items with the hardware items on which they execute*
  - *Heavy emphasis on regression*
  - *Use of simulators and emulators*
- Integration process in Agile and DevSecOps
  - *Utilizing Continuous Integration tools to automate integrating changes into the software*
  - *Existing automated tests will check against changes and immediately flag failed tests, and prevent those changes from being committed*

Verification Process

IEEE 12207 Standard – Technical Processes

- Verification process
  - *Identifies the anomalies (errors, defects, or faults) in any information item (e.g., system/software requirements or architecture description), implemented system elements, or life cycle processes using appropriate methods, techniques, standards or rules.*
  - *For software systems, it includes software verification, software qualification testing and system qualification testing.*

- SMC-S-012 Verification process
  - *Specifies Software Item Qualification Testing (SIQT) to verify requirements in the Software Requirements Specification (SRS) and software-related interface requirements in Interface Requirements Specifications (IRS)*
  - *Defines requirements for Software-Hardware item Integration and Testing, and System Qualification Testing*

- Verification process in Agile and DevSecOps
  - *Automated testing in conjunction with Continuous Integration*
  - *Early testing provides risk reduction*

Transition Process

IEEE 12207 Standard – Technical Processes

- Transition process
  - *Moves the system in an orderly, planned manner into the operational status, such that the system is functional, operable and compatible with other operational systems.*
  - *Installs a verified system with enabling systems, e.g., planning system, support system, operator training system, user training system, as defined in agreements.*
  - *It is used at each level in the system structure and in each stage to complete the criteria established for exiting the stage. It includes preparing applicable storage, handling, and shipping enabling systems.*
  - *For software systems, the purpose of the Transition process is to establish a capability for a system to provide services in a different environment.*

- SMC-S-012 Transition process
  - *Defines documentation (version descriptions, manuals, and plans) required for acquirer to install, understand, use, and maintain the software for operations.*

- Transition process in Agile and DevSecOps
  - *Continuous Integration and Continuous Delivery tools to help automate software artifacts, as well as automated deployment into operational environments*

Validation Process

IEEE 12207 Standard – Technical Processes

- Validation process
  - *Its objective is to acquire confidence in the ability of a system or system element to achieve its intended mission, or use, under specific operational conditions. Validation is ratified by stakeholders.*
  - *It provides the necessary information so that identified anomalies can be resolved by the appropriate technical process where the anomaly was created.*
  - *For software systems, it includes software validation, and software acceptance testing.*

- SMC-S-012 Validation process
  - *Includes Independent Verification and Validation (IV&V)*
  - *More emphases of validating elements used in verification of requirements*

- Validation process in Agile and DevSecOps
  - *User acceptance feedback provides validation of software capability, or provides discovery of changes required*

Operation Process

IEEE 12207 Standard – Technical Processes

- Operation process
  - *Establishes requirements for and assigns personnel to operate the system and monitors the services and operator-system performance.*
  - *Identifies and analyzes operational anomalies in relation to agreements, stakeholder requirements and organizational constraints to sustain services.*

- SMC-S-012 Operation process
  - *Defines requirements for developing plans for operations and for installation, configuration, and checkout*
  - *Single requirement to define plan for uninterrupted operations through transition, and continuity of operational data through each transition*

- Operation process in Agile and DevSecOps
  - *Continuous Delivery provides immediate operational use, driving feedback to developers for improvements*
  - *Rollback capability of data and software used to mitigate any catastrophic, unexpected errors*

Maintenance Process

IEEE 12207 Standard – Technical Processes

- Maintenance process
  - *Monitors the system's capability to deliver services, records incidents for analysis, takes corrective, adaptive, perfective and preventive actions and confirms restored capability.*
  - *For software systems, the Maintenance process makes corrections, changes, and improvements to deployed software systems and elements.*
  - *The need for software system maintenance arises from latent system defects, changes to interfaced systems or infrastructure, evolving security threats, and technical obsolescence of system elements and enabling systems over the system life cycle.*

- SMC-S-012 Maintenance process
  - *Defines manuals required to maintain the system, and developer responsibilities (to acquirer) for the installation, checkout, training, and assistance.*

- Maintenance process in Agile and DevSecOps
  - *Continuous monitoring of applications and infrastructure for security vulnerabilities*
  - *Automated patching of identified issues*
  - *Regular review of security configurations and update of security tools throughout the software lifecycle*

Disposal Process

IEEE 12207 Standard – Technical Processes

- Disposal process
  - *Deactivates, disassembles and removes the system or any of its elements from the specific use.*
  - *Addresses any waste products, consigning them to a final condition and returning the environment to its original or an acceptable condition.*
  - *Destroys, stores, or reclaims system elements and waste products in accordance with legislation, agreements, organizational constraints and stakeholder requirements.*
  - *Includes preventing expired, non-reusable, or inadequate elements from getting back into the supply chain.*
  - *For software systems, it encompasses the termination of services and disposal of software elements, stored data, media and firmware, information items, and associated hardware elements that will not be reused or transitioned to another system.*
- SMC-S-012 Disposal process
  - *Not mentioned*
- Disposal process in Agile and DevSecOps
  - *Not particularly mentioned in Agile / DevSecOps*

Discussion Topics

SMC-S-012 vs ISO-IEC-IEEE 12207-2017

How do we leverage both standards to provide a better software development standard?

- The IEEE 12207 contains a comprehensive set of development life cycle process
  - *However, in making them life cycle model agnostic, it misses key practices for complex mission software (e.g, complex integration)*
- The SMC-S-012 contains more detailed unit test, integration, and validation processes
  - *However, is very brief in the implementation processes*
- What are other strengths from either standard?
- What are other weaknesses that could be improved?

Artificial Intelligence / Machine Learning in Software Development

Future impacts to Software Development Life Cycle

- What applications of AI/ML can be applied to a Software Development Life Cycle?

- Currently **ChatGPT** and **Copilot** have no problems doing menial coding tasks such as:
  - *Write me a binary search segment given a **[data structure]** and output the result as a string.*

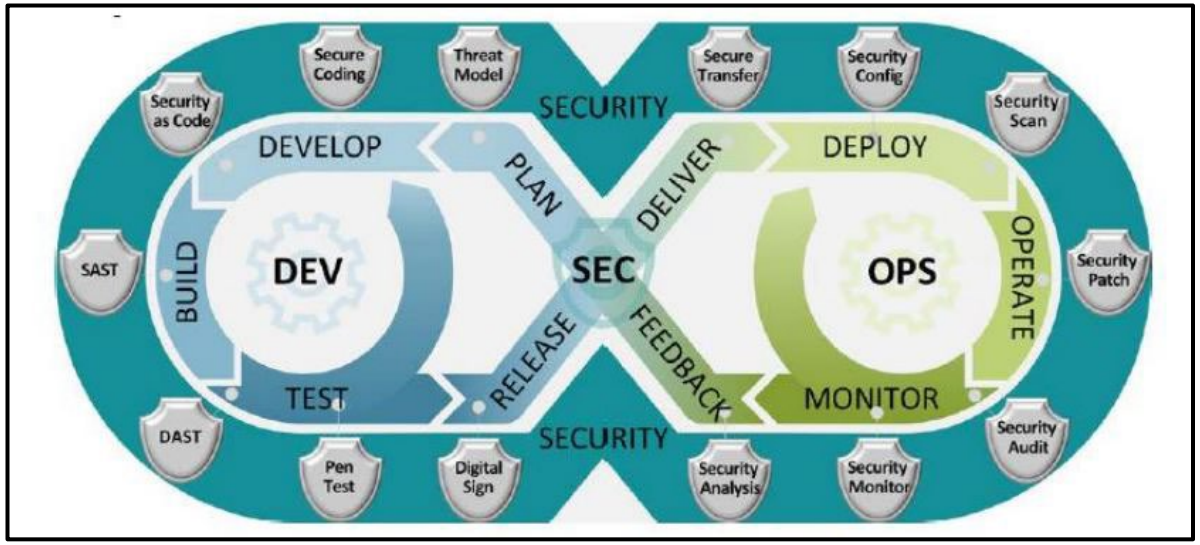
- Difficulties in writing complex architecture but may have improvements in the future. However… **how much do prompts to the language models matter?**
  - *If the goal is to automate as much code as possible in the future, then should there be more **emphasis** on **writing requirements?***

DevSecOps – Continuous Integration / Continuous Deployment

How does software architecture change?

- Encourages **Continuous Integration** and **Continuous Deployment**

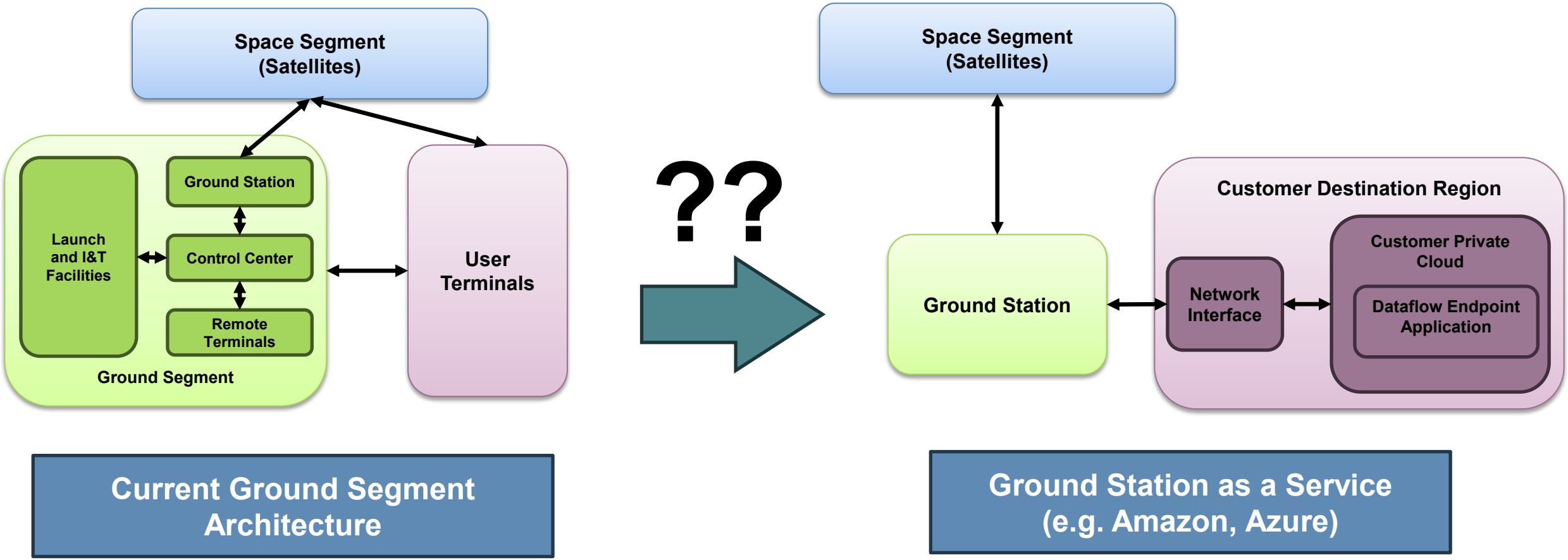
- Automated testing highly recommend – does this also encourage certain software architectures/patterns to make testing feasible and maintainable?
  - *MVC*
  - *Microservices*
  - *Anything else?*



[Source: OTR-2020-00382]

Ground Segment – Traditional to Cloud Infrastructure

Modernization, Open Architecture, and Agility



Ground Segment – Traditional to Cloud Infrastructure

Modernization, Open Architecture, and Agility

IEEE 12207-2017
Section 6.4.10 Transition process

- Traditional ground segments are contained within the system it is developed for; with a push for modernization to cloud infrastructure …
  - *How should ground software development differ in a cloud-based infrastructure?*
  - *If there's a shift from current ongoing operations, what are the implications and how does **transitioning of operations** work? (IEEE 12207-2017 Section 6.4.10)*

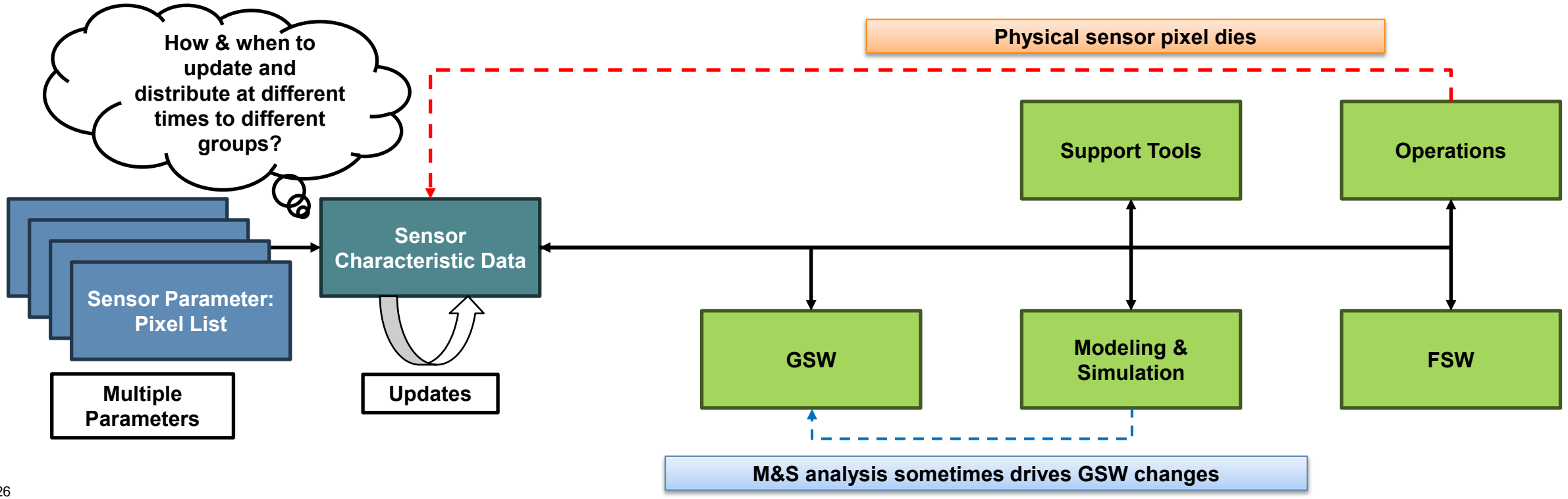
- How is software development impacted from new concepts such as **Ground Station as a Service**?

Interface Interoperability with Increased Modularity

Dealing with dependent modules, and data dependent driven changes between individual entities

- **How can processes help interfaces be truly defined, integrated, and (configure) managed?**
- Increased modularity, either by …
  - *Creating more and smaller software items*
  - *Simulation / Engineering Test Beds (and STE)*
  - *Modular / Open Architectures*
- ***Simplified*** Example (Sensor characteristic data):

IEEE 12207-2017
Section 6.3.6 Information Management
Section 6.4.10 Integration
Section 6.4.4 Architecture Definition

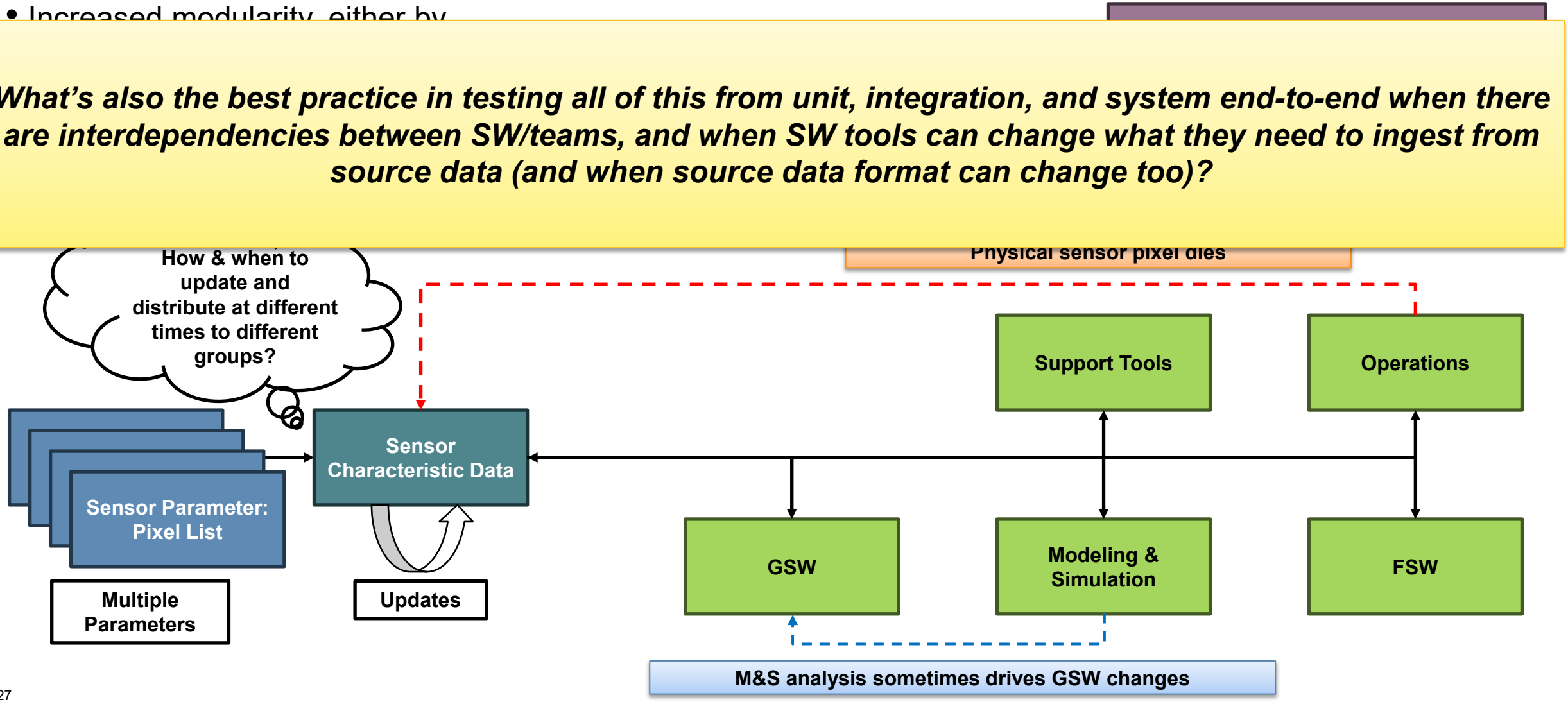


Interface Interoperability with Increased Modularity

Dealing with dependent modules, and data dependent driven changes between individual entities

- **How can processes help interfaces be truly defined, integrated, and (configure) managed?**
- Increased modularity, either by

What's also the best practice in testing all of this from unit, integration, and system end-to-end when there are interdependencies between SW/teams, and when SW tools can change what they need to ingest from source data (and when source data format can change too)?

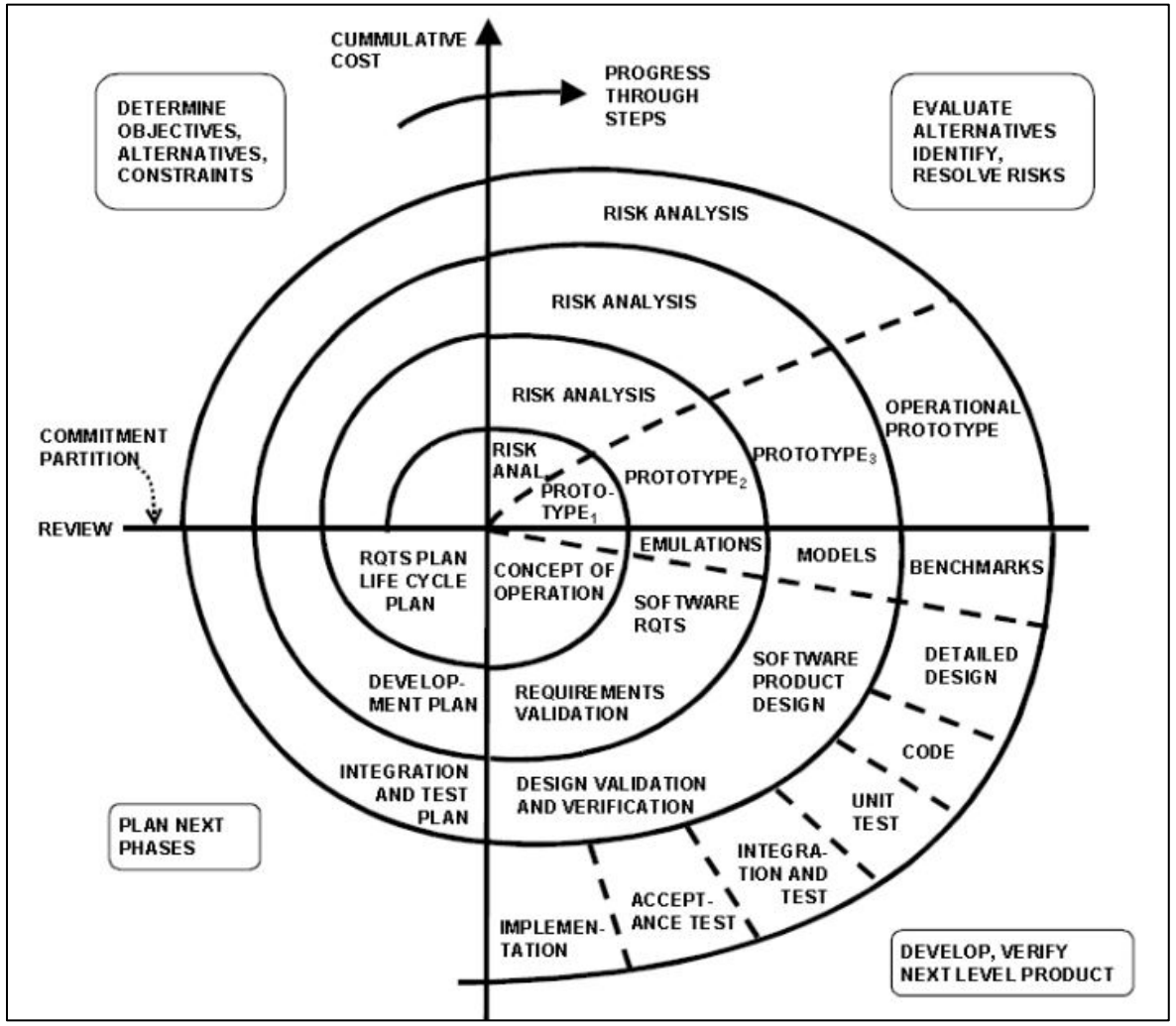


Back Up Charts

Notes on Agile – IEEE 12207-2017

What's different?

- "Iterative" development and prototyping isn't new. (https://en.wikipedia.org/wiki/Spiral_model)
  - *First described by Barry Boehm in his 1986 paper, "A Spiral Model of Software Development and Enhancement"*
  - *In later publications, Boehm describes the spiral model as a "process model generator," where choices based on a project's risks generate an appropriate process model for the project. Thus, the incremental, waterfall, prototyping, and other process models are special cases of the spiral model that fit the risk patterns of certain projects.*
- First IEEE 12207 first published in 1996, 12207-2017 mentions possibilities of various life cycle implementations



[Source: OTR-2010-0107064833-0]

Notes on Agile – IEEE 12207-2017

Expand Upon Agile and DevSecOps?

- IEEE 12207-2017 cognizant of iterative processes
  - *Example* ***6.4.3.3 System/Software requirements definition process – Activities and Task***

> NOTE 1 Requirements definition involves iterative and recursive steps in parallel with other life cycle processes. Depending on the life cycle model that is being employed, it is useful to compare the resources to be spent in assuring initial correctness of requirements versus the resource needed to evolve requirements based on verification and validation results.

  - *Example:* ***6.4.11.1 Validation Process – Purpose***

> The Validation process is typically used at key points in a product's life cycle to demonstrate that the product's requirements for stakeholder intended operational use have been met. Validation is also applicable to the software engineering artifacts (viewed as software system elements). Different domains and engineering or development communities can identify the milestones, validation strategies and criteria differently.
>
> For software systems, highly iterative life cycle models often feature frequent involvement by the acquirer, user representative, or other stakeholders to validate, e.g., the priority of requirements for inclusion in an iteration, the usability of the software interface through prototypes, and the suitability of the software for performing business tasks and fulfilling the operational concept.

- On top of iteration and prototyping, Agile frameworks typically highlight customer collaboration and feedback.

Opportunity to expand upon these topics given Agile/DevSecOps is defined as the chosen lifecycle.

Notes on DevSecOps – IEEE 12207-2017

Security

- **Security** is referenced in 12207 by referring implementation via ISO/IEC Standards. **This is sufficient.**
  - *Complex that cannot be singled to a general standard. Two types:*
    - Platform
    - IT
  - *DoD Programs already have:*
    - DODI 8500.01, 8510.01, 8570
    - Program Protection Plan (PPP) that requires contractors to develop and implement a Program Protection Implementation Plan (PPIP)
    - RMF
    - NIST 800-53 Security Controls
    - … and more.

6.3.1.3 Activities and tasks

The project shall implement the following activities and tasks in accordance with applicable organization policies and procedures with respect to the Project Planning process.

a) **Define the project.** This activity consists of the following tasks:

1) Identify the project objectives and constraints.

NOTE 1 Objectives and constraints include performance and other quality aspects, cost, time and customer and user satisfaction. Each objective is identified with a level of detail that permits selection, tailoring and implementation of the appropriate processes and activities.

NOTE 2 ISO/IEC 15026 *Systems and software assurance,* ISO/IEC 27001 *Information Security Management System* and ISO/IEC 27036, *Information Security for Supplier Relationships,* provide additional guidance on objectives and constraints related to assurance and security.

Notes on DevSecOps – IEEE 12207-2017

Continuous Integration / Continuous Delivery

- Continuous Integration is already mentioned
  - *Can expand in Software Engineering Environment to set up infrastructure to support CI/CD*
- Continuous Delivery not always possible
  - *Can't always deploy to operational environments (space platforms), can expand upon how maintenance/transition/delivery are for Mission Critical Software that do not have a feasible method of updating*

6.4.8 Integration process

6.4.8.1 Purpose

The purpose of the Integration process is to synthesize a set of system elements into a realized system (product or service) that satisfies system/software requirements, architecture, and design.

This process assembles the implemented system elements. Interfaces are identified and activated to enable interoperation of the system elements as intended. This process integrates the enabling systems with the system-of-interest to facilitate interoperation.

Software system integration iteratively combines implemented software system elements to form complete or partial system configurations in order to build a product or service. Software integration is typically performed daily or continuously during development and maintenance stages, using automated tools. Continuous integration involves frequent inclusion or replacement and archiving of items in software libraries under CM control.