Carnegie Mellon University
Software Engineering Institute

DevSecOps Platform-Independent Model: Requirements and Capabilities

Timothy A. Chick
Brent Frye
Aaron Reffett
Natasha Shevchenko
Carol Woody
Joseph Yankel

August 2021

TECHNICAL REPORT
CMU/SEI-2021-TR-010

Program Name

http://www.sei.cmu.edu

Table of Contents

List of Figures

List of Tables

Abstract

Just as products evolve and adapt over time in order to continuously provide value to their users in a secure and cost-effective way, so too must the DevSecOps pipeline. The DevSecOps pipeline evolution is generally driven by changes to organizational business cases, stakeholder requirements, incremental process improvements, and risk mitigations. Given the socio-technical nature of a DevSecOps pipeline, an organization must be mindful in how it instantiates and evolves its DevSecOps pipeline in order to improve the pipeline's ability to effectively envelop participants, processes, and technologies in a secure way, while minimize any negative side effects. The DevSecOps platform-independent model (PIM), outlines the activities necessary to consciously and predictably evolve the pipeline, while providing a formal approach and methodology to building a pipeline tailored to an organization's specific requirements. The use of a DevSecOps platform-specific model (PSM) allows organizations to perform trade-off analyses among alternatives prior to changing the current pipeline instantiation, thus minimizing negative disruptions to the organization's ability to predictably deliver and maintain its products. It allows the organization to reason through the impact of change and to identify where the change should occur in order to provide the most value. To support the analysis and decision-making process, measures must be defined and corresponding data collected in order to provide insight into the decision-making challenges associated with incorporating new capabilities and enhancements into a DevSecOps pipeline.

While organizations, projects, and teams desire to reap the flexibility and speed expected through the implementation of DevSecOps principles, practices, and tools, missing reference material is needed to ensure that DevSecOps is implemented in a secure, safe, and sustainable way. The DevSecOps PIM has been created to address this need. It enables organizations, projects, teams and acquirers to

- specify the DevSecOps requirements to the lead system integrators who need to develop a platform-specific solution that includes the designed embedded system and continuous integration/continuous deployment (CI/CD) pipeline
- assess and analyze alternative pipeline functionality and feature changes as the embedded system evolves
- apply DevSecOps methods to complex products that do not follow well-established software architectural patterns used in industry
- provide a basis for threat and attack surface analysis to build a cyber assurance case in order to demonstrate that the product and the DevSecOps pipeline are sufficiently free from vulnerabilities and that they function only as intended

The DevSecOps PIM provides

- consistent guidance and modeling capability that ensure all proper layers and development concerns relevant to the organization's, project's, and team's needs are captured
- the basis for creating a DevSecOps PSM which can be incorporated into the product's model-based engineering approach as the DevSecOps model is included in the product's model. This allows proper modeling of DevSecOps design trades within a project's analysis of alternatives (AoA) processes, resulting in less costly and more secure products.
- the basis for metrics and documentation of trade-offs to be captured and analyzed through the model-based engineering approach. The model provides dynamic matrices of if those points were addressed, how they were addressed, and how well the corresponding (to the points) module is covered.
- the basis for performing risk modeling against decisions and DevSecOps model-based engineering to ensure security controls and processes are properly selected and deployed

1 Introduction

1.1 What is a DevSecOps Pipeline and How Does it Evolve?

A DevSecOps pipeline is a means for building products that support an organization's mission. To build a pipeline, the details that define what the various technologies used will address must first be prepared by developing business cases and requirements. These cases and requirements are further refined, feeding the pipeline and establishing the development cadence, as shown in Figure 1: Integrated Pipeline and Infrastructure.

Tools and infrastructure capabilities are then selected to allow designers, architects, developers, testers, verifiers, users, operators, and other relevant stakeholders to work together to produce the products needed to meet the objectives using the pipeline (as depicted in the Products box in Figure 1).

In addition, a parallel group of participants implements and supports the automation that allows product creators to build and facilitate management oversight (as depicted in the Capability Delivery box in Figure 1).

Each of these roles requires specialized technical expertise, and each branch relies on the same tools, repositories, and processes structured through the pipeline. The pipeline must be structured to allow each relevant stakeholder to access what they need to perform their role, and the processes must be arranged so that each activity flows through the pipeline and is easily handed off from one role to the next all the way from planning to delivery [Woody 2020].

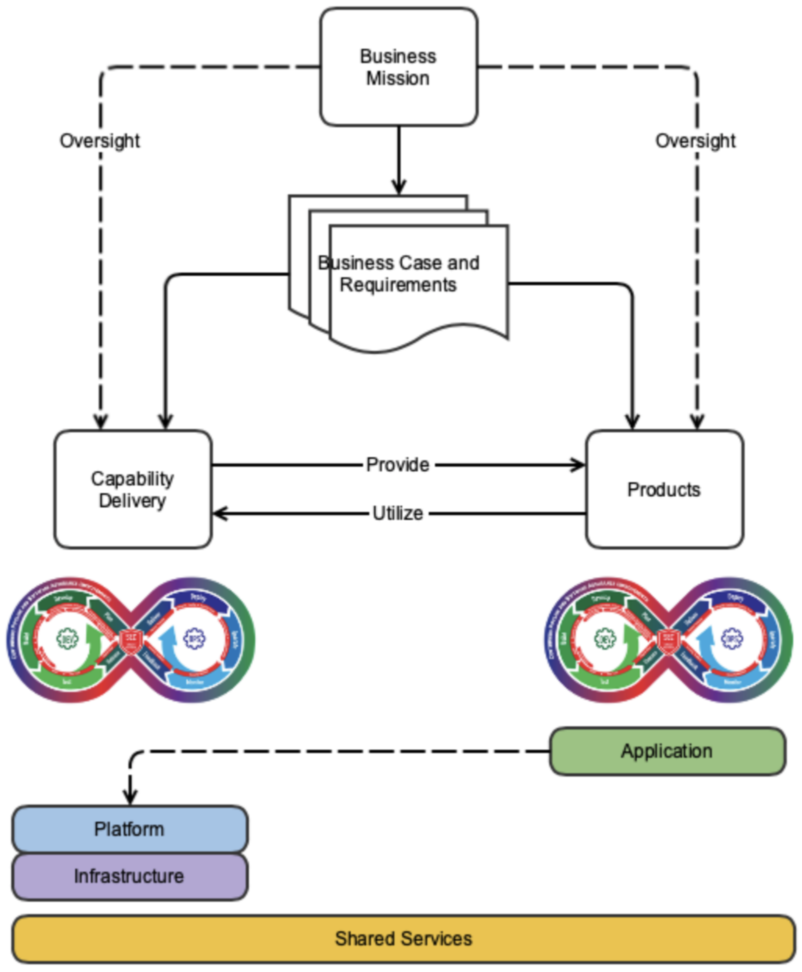


Figure 1: Integrated Pipeline and Infrastructure

Most literature discussing DevSecOps depicts it using some variation of the infinity diagram shown in Figure 2: DevSecOps Infinity Diagram. This is a high-level conceptual diagram since DevSecOps is a cultural and engineering practice that breaks down barriers and opens collaboration between the development, security, and operations organizations using automation to focus on rapid, frequent delivery of secure infrastructure and software to production.

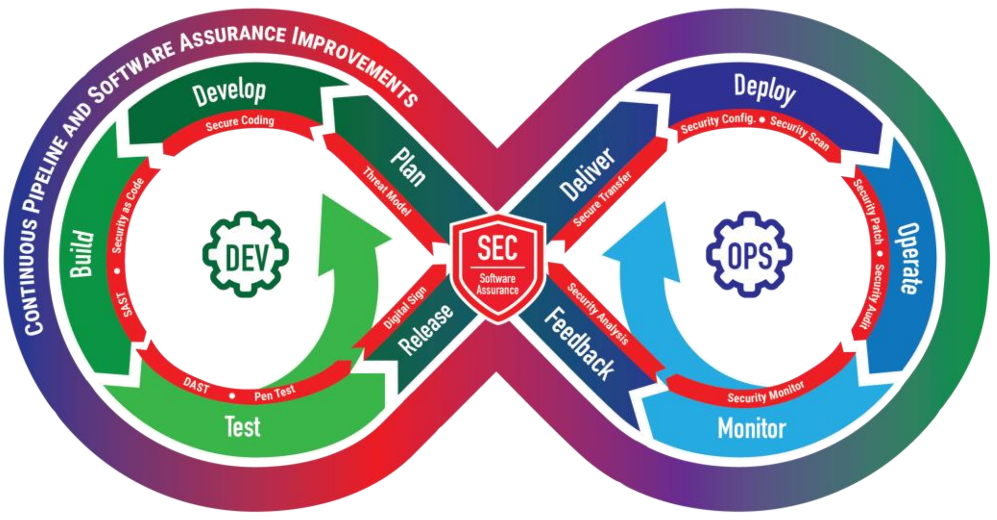


Figure 2: DevSecOps Infinity Diagram

The application and pipeline are built incrementally and are continuously updated to address changing business requirements as well as security and technology demands. It encompasses the intake to the release of software, and manages those flows predictably, transparently, and with minimal human intervention/effort [U.S. General Services Administration 2021]].

DevSecOps isn't simply a technology, a pipeline, or a system. It is an entire socio-technical environment that encompasses the people in certain roles, the processes that they are fulfilling, and the technology used to provide a capability that results in a relevant product or service being provided to meet a need.

Thus, an organization must be mindful of what it is building to instantiate a DevSecOps pipeline that fulfills its particular needs. Unfortunately, there is no one-size-fits-all pipeline. Each DevSecOps pipeline must be tailored to fulfill the needs of a particular program. In some cases, the capability delivery could be more complicated than the products themselves.

The DevSecOps pipeline isn't simply instantiated once and used throughout the product's lifecycle. It is continuously evolving, as the product evolves. The speed and rate of pipeline evolution is affected by the processes and roles that change at a much slower pace than technology, and most organizations don't start by fully automating everything. Instead, the automation of processes is realized over time.

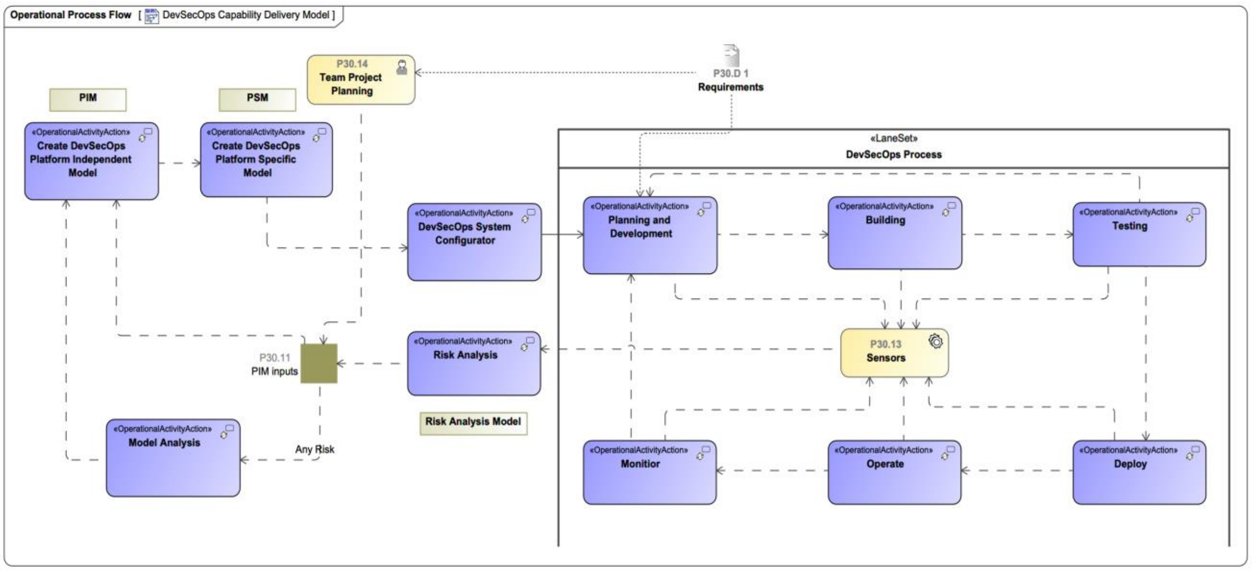

Operational Process Flow [DevSecOps Capability Delivery Model]
P30.14 Team Project Planning
P30.D 1 Requirements
PIM
PSM
«OperationalActivityAction» Create DevSecOps Platform Independent Model
«OperationalActivityAction» Create DevSecOps Platform Specific Model
«OperationalActivityAction» DevSecOps System Configurator
«LaneSet» DevSecOps Process
«OperationalActivityAction» Planning and Development
«OperationalActivityAction» Building
«OperationalActivityAction» Testing
P30.13 Sensors
P30.11 PIM inputs
«OperationalActivityAction» Risk Analysis
Risk Analysis Model
«OperationalActivityAction» Model Analysis
Any Risk
«OperationalActivityAction» Monitior
«OperationalActivityAction» Operate
«OperationalActivityAction» Deploy

Figure 3: DevSecOps Capability Delivery Model

The evolutionary aspects of the DevSecOps capability delivery pipeline are represented in Figure 3: DevSecOps Capability Delivery Model. The DevSecOps Capability Delivery Model adds several new activities to the traditional DevSecOps infinity diagram to represent the mindful nature of establishing and evolving a project’s capability delivery pipeline. The diagram details an activity flow that begins with product requirements which feed the teams’ project planning, and include the capability delivery needs of the product. This, in turn, feeds the DevSecOps PIM, which is used to create a DevSecOps PSM. The PSM is a representation of the current system and its planned updates, preferably maintained using a model-based system engineering tool.

This DevSecOps model captures all socio-technical aspects of the project’s specific capability delivery pipeline. It allows the organization to perform trade-off analyses among alternatives to ensure that the project’s capability delivery pipeline is operating in a cost-effective and secure way, while consistently meeting the needs of the product and all relevant stakeholders.

Based on the model, the capability delivery pipeline is configured and instantiated by the DevSecOps Configurator. The DevSecOps Configurator is analogous to the concept of Infrastructure as Code (IaC) and Configuration as Code (CaC). The product is developed, secured, and operationalized by using the instantiated capability delivery pipeline.

Throughout the lifecycle of the product, data is continuously collected via sensors. This data must be analyzed and evaluated via the Risk Analysis Model. If new risks are identified, such as security vulnerabilities or the possibility of not meeting contractual delivery dates, then the Model Analytics Engine is used to evaluate alternatives to the current capability delivery pipeline instantiation. Resulting changes are made to the DevSecOps Master Model and the process repeats.

Requirements changes require risk analysis as well as an evaluation of the capability delivery which may be impacted.

1.2 What is the DevSecOps Platform-independent Model and Why is it Needed?

Organizations struggle in applying DevSecOps practices and principles in heavily regulated and cybersecurity-constrained environments such as banking, healthcare, and government, because they lack a consistent basis for managing software intensive development, cybersecurity, and operations in a high-speed lifecycle. An authoritative reference is needed to enable organizations to fully design and execute an integrated DevSecOps strategy in which all stakeholder needs are addressed. An example is engineering security into all aspects of the DevSecOps pipeline in order to demonstrate and test the addressing of security concerns for both the pipeline and the product. While large organizations have successfully implemented some aspects of DevSecOps on smaller initiatives, they can struggle to implement these same techniques on large-scale projects. Even in small, relatively successful initiatives, substantial loss of productivity can occur when technical debt and insufficient security and operational practices are in place due to the lack of knowledge, experience, and reference material needed to fully design and execute an integrated DevSecOps strategy in which all stakeholder needs are addressed.

While organizations, projects, and teams desire to reap the flexibility and speed expected through the implementation of DevSecOps principles, practices, and tools, missing reference material is needed to ensure DevSecOps is implemented in a secure, safe, and sustainable way. The DevSecOps platform-independent model (PIM) has been created to address this need. It enables organizations, projects, teams and acquirers to

- specify the DevSecOps requirements to the lead system integrators tasked with developing a platform-specific solution that includes the designed system and continuous integration/continuous deployment (CI/CD) pipeline
- assess and analyze alternative pipeline functionality and feature changes as the system evolves
- apply DevSecOps methods to complex products that do not follow well-established software architectural patterns used in industry
- provide a basis for threat and attack surface analysis to build a cyber assurance case in order to demonstrate that the product and DevSecOps pipeline are sufficiently free from vulnerabilities and that they function only as intended.

While one can search “DevSecOps” on the internet and find a lot of literature that paints a picture of what DevSecOps could be or should be, this literature is not definitive and requires a considerable amount of interpretation, particularly for heavily regulated and cybersecurity-constrained environments. This results in

- DevSecOps perspectives not being fully integrated in organizational guidance and policy documents
- projects being unable to perform an analysis of alternatives (AoA) regarding the DevSecOps pipeline tools and processes
- multiple projects using similar infrastructure and pipelines in different and incompatible ways, even within the same organization
- suboptimal tools and security controls

The DevSecOps PIM provides

- consistent guidance and modeling capability that ensure all proper layers and development concerns relevant to the organization’s, project’s, and team’s needs are captured
- the basis for creating a DevSecOps platform-specific model (PSM) which can be incorporated into the product’s model-based engineering approach as the DevSecOps master model is included in the product’s model. This allows proper modeling of DevSecOps design trades within a project’s AoA processes, resulting in less costly and more secure products.
- the basis for metrics and documentation of trade-offs to be captured and analyzed through the model-based engineering approach. The model provides dynamic matrices of if those points were addressed, how they were addressed, and how well the corresponding (to the points) module is covered.

- the basis for performing risk modeling against decisions and DevSecOps model-based engineering to ensure security controls and processes are properly selected and deployed

Large, complex, heavily regulated, and cybersecurity-constrained projects have already embraced model-based engineering but have not applied the same techniques to their DevSecOps CI/CD pipelines. This limits a project’s ability to build a cyber-physical software factory that is fit for purpose. Establishing a DevSecOps PIM enables projects to develop a robust framework for creating a customized model where the system's architecture and the DevSecOps pipeline architecture are not in conflict and where they address the larger attack surface of the project. This allows DevSecOps to become a part of the enterprise architecture of the product being built, in contrast to current practices where DevSecOps is not included in the overall product architecture and does not effectively integrate with the compliance and operational context of the project.

1.3 What is a Platform-independent Model?

The goal of software system architecture is to align a (large) group of stakeholders in the same direction. For less complex software systems with well-established patterns, the importance of an architecture focus diminishes as one can simply follow the well-established patterns and associated solutions. However, many heavily regulated and cybersecurity-constrained software systems are more complex and require custom architectural patterns. Attention must be paid to maintaining the architecture and ensuring that the impact of requirement/feature changes are well understood and acceptable, especially with respect to impacts on security and operations. Otherwise, stakeholders will creatively solve their local problems that may then violate the overall structure.

A reference architecture is an authoritative source of information about a specific subject area that guides and constrains the instantiations of multiple architectures and solutions [U.S. Department of Defense 2018]. A reference architecture provides guidelines that ensure proper attention to, and management of, the system architecture. For more complex systems, there will be parts of the system architecture that are well understood and other parts that are not. This complexity can be better managed by separating the parts of the system that are well-known from those that are largely unknown. Then, it will be easier for relevant stakeholders to determine the right abstractions for the well-known parts that support future unknown (albeit expected) features and do not adversely impact security and operations.

As for the unknown parts of the system architecture, they are difficult to specify and, therefore, should not be specified. The likelihood of getting the specifications wrong is quite high. These parts should be made clear and visible in the system architecture. The unknown parts of the architecture will continuously (iteratively) evolve but this evolution must be carefully considered so as not to destroy or diminish the existing well-known parts of the architecture. Over time, the abstractions for the unknown parts will become better defined. Development, security, and operations teams and other stakeholders should be involved in this evolution to ensure that the proper balance between features, defensibility, and stability is maintained in a cost-effective manner.

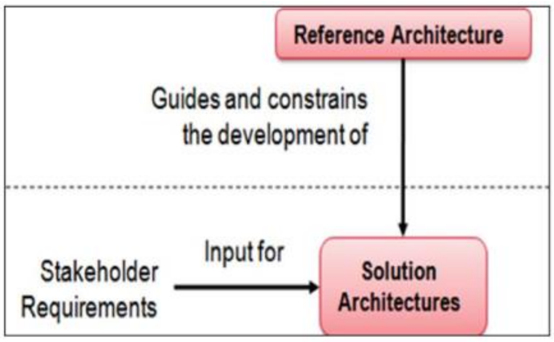


Figure 4: Reference and Solution Architecture Relationship

It is not uncommon for a project or team to go directly from a reference architecture to a platform-specific model (PSM) or even directly to a solution. A PSM is a model of a product or service being built by a project, or team, that is linked to a specific technology, such as specific programming languages, automated testing tools, issue tracking, etc. and how they are integrated and used together to meet a defined need. A platform-independent model (PIM) is a general and reusable model of a solution to a commonly occurring problem in software engineering within a given context that is independent of the specific technological platform used to implement it. It is more detailed than a reference architecture but stops short of specifying a specific technological implementation. The DevSecOps PIM bridges the gap between high-level theory and current DevSecOps instantiations. It provides the basis for consistently building and maintaining DevSecOps pipelines that are fit for purpose.

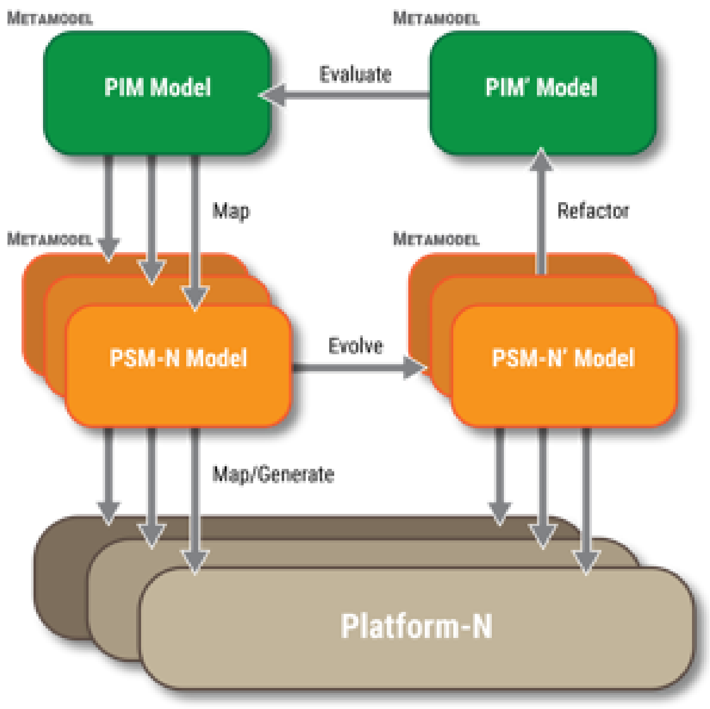


Figure 5: PIM and PSM Relationship to Platform Instantiations

The relationship between the PIM and PSM allows stakeholders to assess functionality and feature changes as the DevSecOps pipeline and product(s) under development evolve by providing a definitive definition of what a DevSecOps pipeline is and how it matures over time.

An Analogy: When cooking, there is a difference between following a simple, imprecise recipe and a complicated, detailed recipe. When preparing a simple dish, you can afford to use an unclear recipe, as even relatively major mistakes can be fixed and even avoided just by having some experience. But if you are making a complicated, time-intensive dish that requires not only general experience, but expertise and skills in very specific areas of knowledge and/or techniques (like development, security, and operations), the recipe should be as detailed as possible, guide you in every step, and give you instruments to check your progress and make adjustments along the way. Otherwise, even a few minor mistakes can ruin the dish. Today, projects and teams try to follow a simple recipe when developing their DevSecOps pipelines, which works fine for well-known patterns. However, for large, complex systems that do not conform to known patterns, projects need a more complete detailed recipe to avoid costly mistakes and to make informed adjustments along the way. The DevSecOps PIM is this detailed complete recipe.

1.4 Using the DevSecOps PIM

The DevSecOps pipeline isn't simply instantiated once and used throughout the product's lifecycle. It continuously evolves along with the product. The speed and rate of pipeline evolution is affected by the processes and roles that change at a much slower pace than technology, and most organizations don't start by fully automating everything. Instead, the automation of processes is realized over time. As a result, two views of the DevSecOps requirements were created along with the corresponding maturity and capability levels.

The first, and primary, perspective is a software lifecycle view of the requirements, as shown in Figure 6: DevSecOps System Requirements. In the figure, the Governance and System Infrastructure boxes represent the requirements focused on the enablement of the software development lifecycle, while the other boxes represent the engineering activities performed on a product under development as it iteratively evolves. Each requirement has a key that articulates the category and a unique number, referred to as "Id" in modeling diagrams. The numbers break down to show encapsulation. For example: Sys_5 is the higher requirement and Sys_5.1 and Sys_5.2 are more detailed requirements. Sys_5.1.1 is a child of Sys_5.1. At a minimum, all requirements have an Id, title, requirement statement (referred to as "text" in modeling diagrams), and a stereotype. Stereotypes are used to capture the maturity level and capability attributes of the requirements. In addition to these basic characteristics, many requirements have additional informative information (referred to as "documentation" in the modeling diagrams). This information is provided in order to help the model user understand the intent and expectation of the requirement.

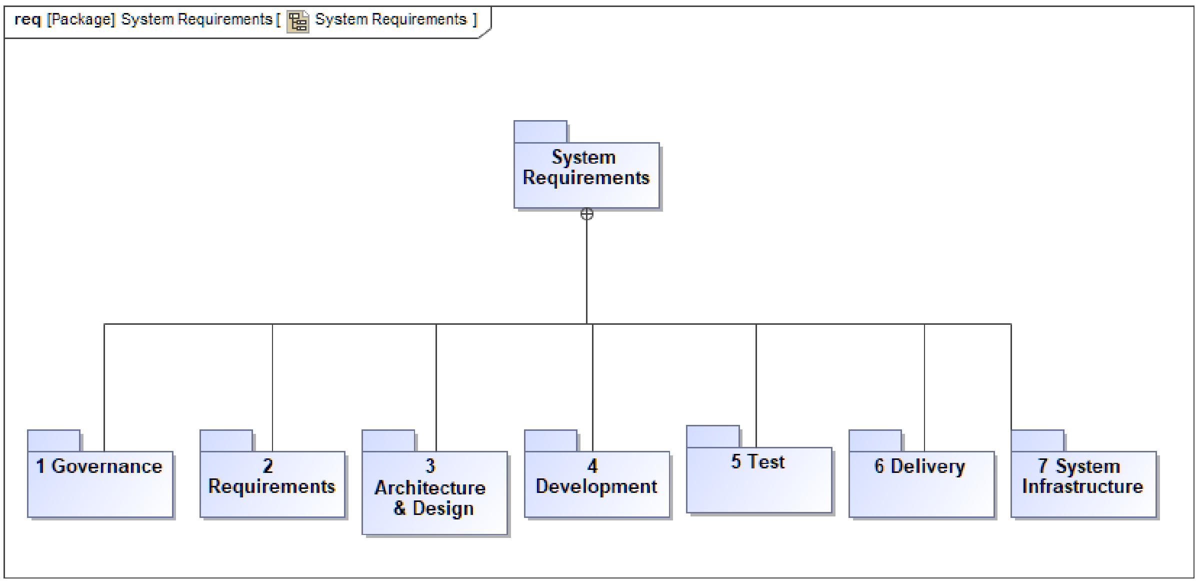


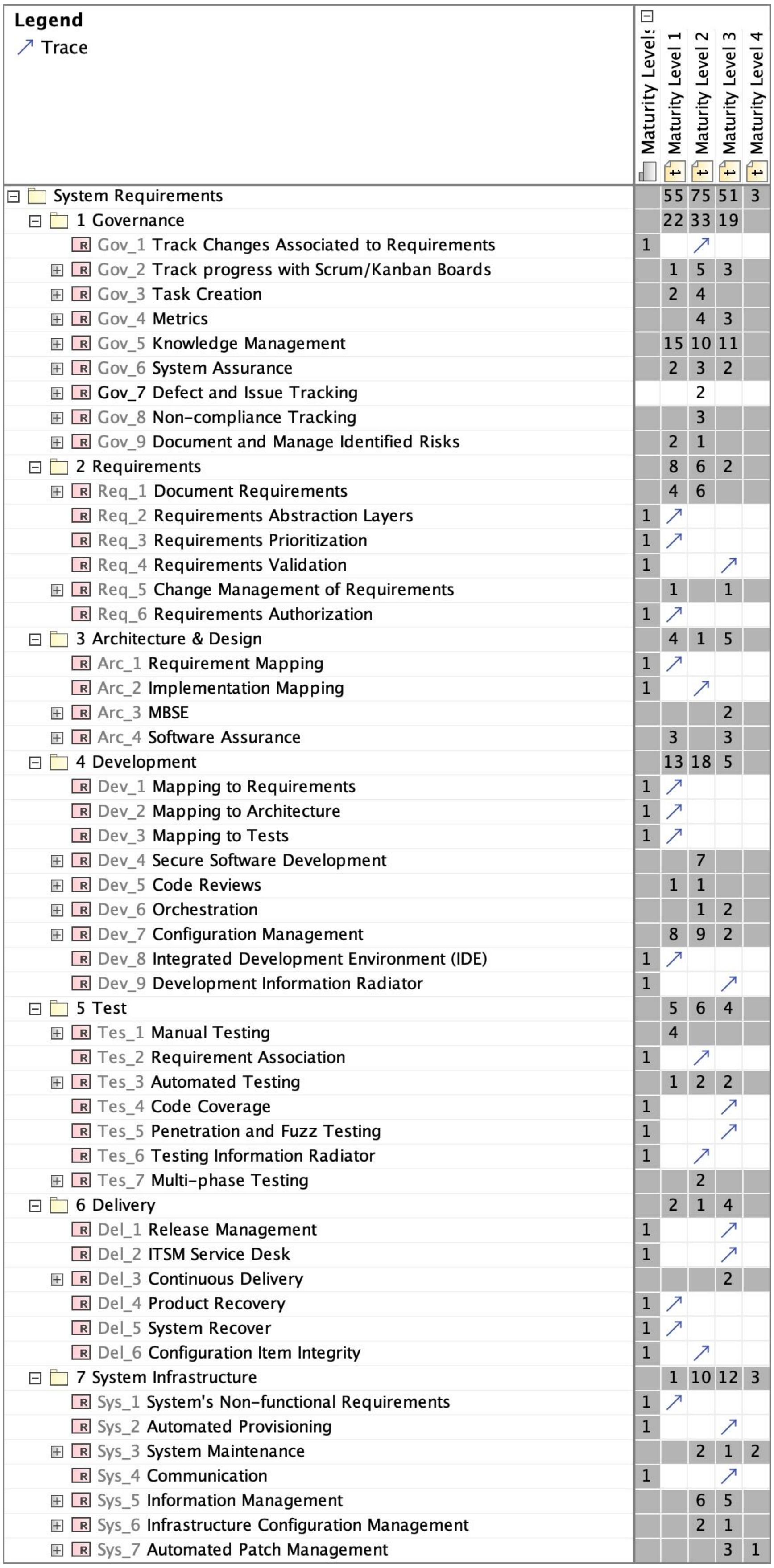
Figure 6: DevSecOps System Requirements

In addition to the encapsulation, or containment relationship between the requirements captured in the numbering system, there are trace relationships (i.e., requirement to requirement) among many requirements. A summary of these trace relationships can be found at the end of this document in Figure 58: Requirements to Requirements Relationship Matrix. In the detailed modelling diagrams, these relationships are shown via a dotted line with one of the following labels: “trace,” “copy,” “Depends on,” and “deriveReqt.”

- “Trace” is the most general form of a relationship as it indicates nothing more than the existence of a relationship between the two requirements.
- “Copy” relationships are contextual copies of two or more requirements. While the text of the requirements is identical, the title and requirement attributes differ in order to contextually articulate the various ways that the given requirement must be implemented to satisfy the needs of the system.
- “Depends on” is used to show the dependency between 2 requirements in which one requirement must be satisfied in order to achieve the other requirement.
- Finally, a “deriveReqt” relationship simply provides an origin or driver relationship for a given requirement.

The trace relationship among requirements, as shown in Figure 58: Requirements to Requirements Relationship Matrix is not an exhaustive relationship matrix as other requirements relationships do exist.

In general, the requirements are written to capture the ideal state of DevSecOps. However, as the authors started building the model it became clear that the ideal state of DevSecOps is too much for a person or group to adopt all at once. In fact, most of the DevSecOps adoptions the authors have observed have been instantiated and executed using an iterative approach that evolves into the ideal state. This insight led to the creation of maturity levels as defined in Table 2: DevSecOps Maturity Levels, where level 1 is the least mature and level 4 is the ideal state of DevSecOps. Each requirement has been mapped to a maturity level, as summarized in Figure 7: Requirements to Maturity Levels Matrix. The levels are accumulative in nature. For example, in order to address and meet the level 2 requirements, you must also address and meet all level 1 requirements.



Legend
↗ Trace

	Maturity Level	Maturity Level 1	Maturity Level 2	Maturity Level 3	Maturity Level 4
System Requirements		55	75	51	3
1 Governance		22	33	19	
Gov_1 Track Changes Associated to Requirements	1		↗		
Gov_2 Track progress with Scrum/Kanban Boards		1	5	3	
Gov_3 Task Creation		2	4		
Gov_4 Metrics			4	3	
Gov_5 Knowledge Management		15	10	11	
Gov_6 System Assurance		2	3	2	
Gov_7 Defect and Issue Tracking			2		
Gov_8 Non-compliance Tracking			3		
Gov_9 Document and Manage Identified Risks		2	1		
2 Requirements		8	6	2	
Req_1 Document Requirements		4	6		
Req_2 Requirements Abstraction Layers	1	↗			
Req_3 Requirements Prioritization	1	↗			
Req_4 Requirements Validation	1			↗	
Req_5 Change Management of Requirements		1		1	
Req_6 Requirements Authorization	1	↗			
3 Architecture & Design		4	1	5	
Arc_1 Requirement Mapping	1	↗			
Arc_2 Implementation Mapping	1		↗		
Arc_3 MBSE				2	
Arc_4 Software Assurance		3		3	
4 Development		13	18	5	
Dev_1 Mapping to Requirements	1	↗			
Dev_2 Mapping to Architecture	1	↗			
Dev_3 Mapping to Tests	1	↗			
Dev_4 Secure Software Development			7		
Dev_5 Code Reviews		1	1		
Dev_6 Orchestration			1	2	
Dev_7 Configuration Management		8	9	2	
Dev_8 Integrated Development Environment (IDE)	1	↗			
Dev_9 Development Information Radiator	1			↗	
5 Test		5	6	4	
Tes_1 Manual Testing		4			
Tes_2 Requirement Association	1		↗		
Tes_3 Automated Testing		1	2	2	
Tes_4 Code Coverage	1			↗	
Tes_5 Penetration and Fuzz Testing	1			↗	
Tes_6 Testing Information Radiator	1		↗		
Tes_7 Multi-phase Testing			2		
6 Delivery		2	1	4	
Del_1 Release Management	1			↗	
Del_2 ITSM Service Desk	1			↗	
Del_3 Continuous Delivery				2	
Del_4 Product Recovery	1	↗			
Del_5 System Recover	1	↗			
Del_6 Configuration Item Integrity	1		↗		
7 System Infrastructure		1	10	12	3
Sys_1 System's Non-functional Requirements	1	↗			
Sys_2 Automated Provisioning	1			↗	
Sys_3 System Maintenance			2	1	2
Sys_4 Communication	1			↗	
Sys_5 Information Management			6	5	
Sys_6 Infrastructure Configuration Management			2	1	
Sys_7 Automated Patch Management				3	1

Figure 7: Requirements to Maturity Levels Matrix

As a DevSecOps system matures, so will its capabilities. The second perspective that requirements are mapped to is a capability view. All requirements are mapped to the 10 capabilities shown in Figure 8: DevSecOps Capabilities, and grouped under a top-level capability referred to as the DevSecOps Pipeline Capability.

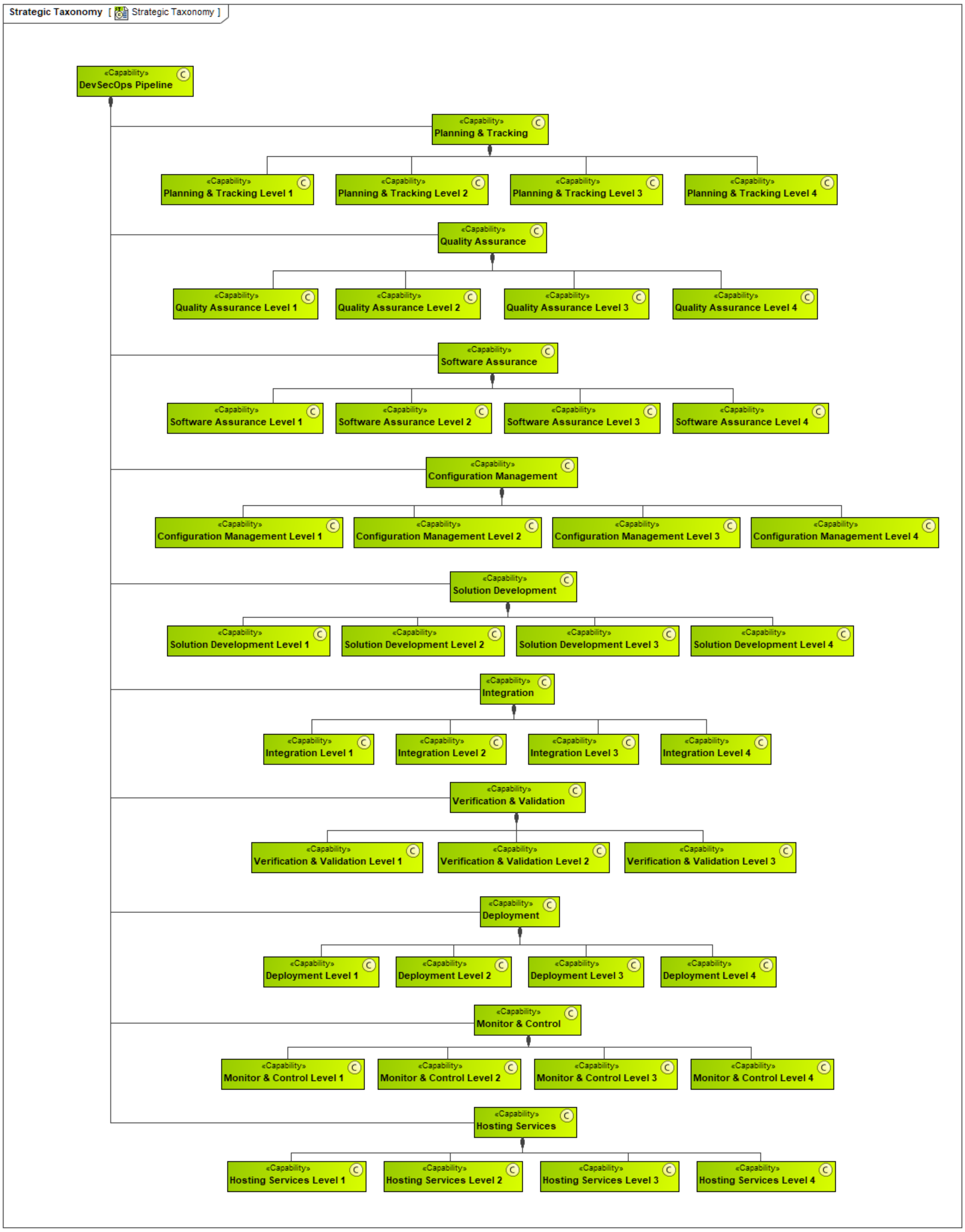

Strategic Taxonomy [Strategic Taxonomy]
«Capability» DevSecOps Pipeline
«Capability» Planning & Tracking
«Capability» Planning & Tracking Level 1
«Capability» Planning & Tracking Level 2
«Capability» Planning & Tracking Level 3
«Capability» Planning & Tracking Level 4
«Capability» Quality Assurance
«Capability» Quality Assurance Level 1
«Capability» Quality Assurance Level 2
«Capability» Quality Assurance Level 3
«Capability» Quality Assurance Level 4
«Capability» Software Assurance
«Capability» Software Assurance Level 1
«Capability» Software Assurance Level 2
«Capability» Software Assurance Level 3
«Capability» Software Assurance Level 4
«Capability» Configuration Management
«Capability» Configuration Management Level 1
«Capability» Configuration Management Level 2
«Capability» Configuration Management Level 3
«Capability» Configuration Management Level 4
«Capability» Solution Development
«Capability» Solution Development Level 1
«Capability» Solution Development Level 2
«Capability» Solution Development Level 3
«Capability» Solution Development Level 4
«Capability» Integration
«Capability» Integration Level 1
«Capability» Integration Level 2
«Capability» Integration Level 3
«Capability» Integration Level 4
«Capability» Verification & Validation
«Capability» Verification & Validation Level 1
«Capability» Verification & Validation Level 2
«Capability» Verification & Validation Level 3
«Capability» Deployment
«Capability» Deployment Level 1
«Capability» Deployment Level 2
«Capability» Deployment Level 3
«Capability» Deployment Level 4
«Capability» Monitor & Control
«Capability» Monitor & Control Level 1
«Capability» Monitor & Control Level 2
«Capability» Monitor & Control Level 3
«Capability» Monitor & Control Level 4
«Capability» Hosting Services
«Capability» Hosting Services Level 1
«Capability» Hosting Services Level 2
«Capability» Hosting Services Level 3
«Capability» Hosting Services Level 4

Figure 8: DevSecOps Capabilities

These capabilities are groupings of requirements that, when combined, define a collective competency in performing a set of functional activities across the product lifecycle. These capabilities are defined in Table 1: DevSecOps Capability Definitions. The capability levels represent the measure of consistency and completeness, which is usually achieved through increased automation, in which functional activities are performed. The requirements maturity level represents the minimal capability level the given requirement is expected to meet. For example, if a given requirement is mapped to the Planning & Tracking capability and maturity level 3, then in order to consider the requirement met, it would need to achieve the capability expectations of Planning & Tracking level 3. The requirement's maturity can grow as the level of capability improves. For example, one could implement a maturity level 3 requirement at capability level 4.

Table 1: DevSecOps Capability Definitions

Name	Documentation
Configuration Management	Configuration Management is the set of activities used to establish and maintain the integrity of the system and product under development, and associated supporting artifacts, throughout their useful lives. Different levels of control are appropriate for different supporting artifacts and implementation elements and for different points in time. For some supporting artifacts and implementation elements, it may be sufficient to maintain version control of the artifact or element that is traced to a specific instance of the system or product under development in use at a given time, past or present, so that all information related to a given instance, or version, of the system or product under development is known. In which case, all other variations of the artifacts and elements can be discarded as subsequent iterations are generated or updated. Other supporting artifacts and implementation elements may require formal configuration in which case baselines are defined and established at predetermined points in the lifecycle. Baselines, and subsequent changes, are formally reviewed and approved, and serve as the basis for future efforts. The configuration management capability of a system matures as the consistency and completeness of the integrity controls are put in place to capture all supporting artifacts and implementation elements associated with the system and product under development while keeping pace with the DevSecOps pipeline through automation and integration with all aspects of the lifecycle. This includes; (1) the relationship between artifacts and elements for a given instance, or version, of the system or product under development, (2) capturing sufficient information to identify and maintain configuration items, even if those who created them are no longer available, (3) defining the level of control each artifacts and elements requires based on technical and business needs, (4) systematically controlling and monitoring changes to configuration items, and (5) enforcement and logging of all required relevant stakeholder reviews and approvals, based on the organization, project, and team policies and procedures.
Configuration Management Level 1	• All supporting artifacts and implementation elements that require configuration control are identified and documented. • The level of configuration control for each supporting artifact and implementation element is defined. • While the configuration management of supporting artifacts may be a fully manual process, an automated version control system, or set of systems, must be in place to track current and historical versions of files used to create implementation elements.
Configuration Management Level 2	• Automated configuration management system(s) are in place for all identified supporting artifacts and implementation elements. • Immutable logging of all changes to configuration items and associated metadata, such as who made the change, when the change occurred, and what was changed. • Changes to the system and product under development is associated with an approved requirement or change request. • All relevant stakeholders are notified when changes to configuration items are requested. • Some integration between the automated version control system used for file tracking and other aspects of the DevSecOps pipeline has occurred in order to enable the automatic triggering of other activities. • The automated version control system traces relationships between test artifacts and requirements, and test results and associated artifacts to a specific instance of the system or product under development in use at a given time, past or present.
Configuration Management Level 3	• Manage and control the volatility of change. Be able to identify impacted supporting artifacts and implementation elements a given change request will impact. • Use automatic discovery tools to scan current instance of system and product under development, and associated configurations, to identify mismatches between current instance and approved versions under configuration management in order to ensure integrity of the instantiated instances. Automatically report all mismatches to relevant stakeholders. • The system shall automatically maintain an audit trail of all system configuration changes to include what was changed, who/what changed it, when the change occurred. • System only allows authorized individuals, or entities, to make specific types of changes to the product under development based on the individual's role, or entity's purpose, and where they are in the DevSecOps pipeline.

Name	Documentation
Configuration Management Level 4	• Automatically correct any misconfiguration of the currently instantiated system and product under development based on approved supporting artifacts and implementation elements under configuration control. • The system shall monitor user activities and actively identify security-related actions and system configuration changes that are uncharacteristic of the given user and notify relevant stakeholders of the uncharacteristic behavior to validate the change w as appropriate and to avoid insider threats. • A fully automated change proposal process, w here changes are proposed and automatically routed to relevant stakeholders for approval and implemented by the system.
Deployment	Deployment is the set of processes related to the delivery or release of the product under development into the environment in w hich users of the product interact with the product. The deployment capabilities of the system mature w ith increased levels of automation, advanced rollback and release functionality, along w ith disaster recovery, speed, and accuracy.
Deployment Level 1	• The system can manually recover if a failure occurs in a deployed product, deploying the product at the last know n acceptable state.
Deployment Level 2	• A quality criteria for the deployment of the system and product under development is defined. While monitoring for failures can be a combination of manual and automated detection processes: • The system can automatically recover if a failure occurs in a deployed product, deploying the product at the last know n acceptable state. • The system can automatically recover the product to a previously w orking state in the event of system failure. • The system can track the changes betw een deployed products, and the personnel and reasoning involved in the change.
Deployment Level 3	• Both the system and product under development are fully automated in terms of orchestration and deployment into target environments • Various release strategies are supported to include canary, Blue-Green, multiple service, batch, rolling, and A/B Testing. • The product under development is deployed continuously, supported by sufficient automation in w hich no human intervention is required to release the product to its users. • The system shall automatically collect the necessary data to monitor the system and product under development for failures and quality issues, and alert relevant stakeholders w hen corrective actions are required. • In the event that a failure or cancellation occurs during deployment of the product or system, the system w ill automatically restore a the most recent w orking version. • Automated updating or patching of softw are used by the system. Patches are rolled out automatically to the various parts of the system.
Deployment Level 4	• Continuous improvement of the testing procedures is performed based on the data collected from the system and product under development tests. • The system shall automatically identify and track w hen the defined quality criteria has not been met and the automated quality controls have been bypassed. All relevant stakeholders w ill be automatically notified and the non-compliance issue w ill be tracked to closure.
DevSecOps Pipeline	The DevSecOps pipeline is a socio-technical system composed of both softw are tools and processes. As the capability matures it seamlessly integrates three traditional factions that sometimes have opposing interests: development values features, security values defensibility, and operations values stability. A DevSecOps pipeline emerges w hen continuous integration of these three factions is used to meet organizational, project, and team objectives and commitments.
Hosting Services	Hosting services are made up of the underlying infrastructure and platforms that both the system and product under development operate upon. This includes the various cloud providers, on premises bare-metal and virtualization, netw orks, and other SaaS that is utilized along w ith the management, configuration, access control, ow nership, and personnel involved.
Hosting Services Level 1	• The hosting services adequately support the scalability, reliability, regulatory, and security requirements to operate, maintain, and build an organizations product. • The hosting services provide compatibility w ith the testing framew orks and tools utilized throughout system and product development lifecycles.
Hosting Services Level 2	• Logs from hosting services are aggregated, auditable, and analyzable. • System transaction logs are available and immutable. • Performance metrics can be visualized, analyzed for hardw are, softw are, database and netw ork components. • Role-based access control is utilized throughout. • All information collected uses proper techniques to maintain privacy and sensitivity concerns, and can be properly disposed of w hen necessary. • All configuration items are identified and resources are planned and executed in order to maintain

Name	Documentation
	configuration integrity of the given item. • Disaster recovery processes are documented and supported.
Hosting Services Level 3	• The system infrastructure is provisioned using IaC and is automated. • Captured metrics can generate alerts based off of defined values. • Ability to automatically alert and communicate metrics associated with security risks of the underlying infrastructure to stakeholders so they can manage risk and make decisions regarding risk and impact to software applications. • Automatic upgrading of operating system software, and supporting services.
Hosting Services Level 4	• Qualities such as performance, capacity, security, compliance and risk tolerance are continuously being monitored using automated tools. Results from the automated tools are automatically reported to all relevant stakeholders to ensure the quality of the automated process and to identify and track improvements to quality attributes. • System configuration and performance are continuously being monitored using automated tools to identify and report all anomalies. Results from the automated tools are automatically reported to all relevant stakeholders so they can manage risk and make decisions. • Infrastructure is immutable and can be automatically replaced vs update in place.
Integration	Integration is the process of merging changes from multiple developers made to a single code base. Integration can be made manually on a periodic basis, typically by a senior or lead engineer, or it can be made continuously by automated processes as individual changes are made to the code base. In either case the purpose of integration is to assemble a series of changes, merge and deconflict them, build the product and ensure that it functions as intended and that no change broke the w hole product, even if those changes w orked in isolation.
Integration Level 1	• Documented, repeatable, processes exist which may be manual, automated, or some combination of the tw o • Some individual processes (e.g., merging changes) may require expert subjective judgement • Processes may require manual intervention betw een phases and/or to coordinate steps betw een disparate systems • Some human-human and human-process contact occurs outside the orchestration pipeline • Process initiation is manual and irregular
Integration Level 2	• Most individual processes are scripted and repeatable • Expert subjectivity has been removed from all processes by adopting processes with objective criteria for success • An orchestrated integration pipeline exists; however, it may not be fully automated • Some human-human and human-process contact occurs outside the orchestration pipeline • Integration process initiation is regular w hether manual or automated
Integration Level 3	• All individual processes are scripted and fully automated • An orchestrated integration pipeline controls all processes from start to finish • All human-process contact occurs from w ithin the context of the orchestration pipeline (e.g., approvals captured in ticketing system, SCM, etc. and orchestration continues)
Integration Level 4	• The entire integration pipeline is fully automated requiring no manual intervention • The entire integration pipeline runs in near real time as changes are committed to the code base • Alerts, notifications and results of integration are sent to relevant engineers automatically • A successfully integrated product is ready for delivery with no additional manual processes required
Monitor & Control	Monitor and Control involves continuously monitoring activities, communicating status, and taking corrective action in order to proactively address issues and to consistently improve performance. More mature projects automate as much of this as possible. Appropriate visibility enables timely corrective action to be taken w hen performance deviates significantly from what was expected. A deviation is significant, if w hen left unresolved, it precludes the project from meeting its objectives. Items that should be monitored include cost, schedule, effort, commitments, risks, data, stakeholder involvement, corrective action progress as w ell as task & w ork product attributes like size, complexity, w eight, form, fit or function.
Monitor & Control Level 1	• All supporting artifacts and implementation elements that require monitoring and control are identified and documented. • The level of monitor and control for each supporting artifact and implementation element is defined. • A policy and plan for planning and performing the monitor and control capability is established and maintained. • The w ork products of the monitor and control capability are placed under appropriate levels of control.
Monitor & Control Level 2	• The people performing or supporting the monitor and control capability are trained as needed. • Automated monitor and control system(s) are in place for all identified supporting artifacts and implementation elements. • Automated collection of w ork products, measures, and measurement results are in place. • Automated comparison of actual measurements to expected measurements is performed and deviations

Name	Documentation
	are quantified. • Automated alerting w hen significant deviations occur.
Monitor & Control Level 3	• The relevant stakeholders of the monitor and control capability are identified, involved, and are obtaining the information they need to make decisions. • Automated sharing of monitor and control information to relevant stakeholders. • Stakeholders can tailor the visualizations of the information provided to meet their needs.
Monitor & Control Level 4	• The monitor and control capability is itself subject to monitored and controlled and corrective action is taken w hen necessary. • Automated collection of monitor control capability w ork products, measures, measurement results and improvement information including records of significant deviation, criteria for significant deviation, and corrective action results are in place. • Root causes of defects and other problems in the monitor and control capability are identified and corrected. • Monitor and control capability is itself subject to continuous improvement.
Planning & Tracking	Planning and Tracking is the set of practices used to define tasks and activities, along w ith the resources needed to perform them, required to achieve an objective, or commitment, and track progress, or lack thereof, tow ards achieving the given objective. It provides the mechanisms required to inform relevant stakeholders w here an effort currently is within the process and whether it is on track to provide the expected outcomes. These mechanisms allow relevant stakeholders to determine w hat has been accomplished and w hat adjustments or corrective actions need to occur to account for impediments and other unforeseen issues. Ideally, impediments and issues are proactively identified and addressed. Practices include documenting activities and breaking them dow n into actionable w ork in w hich resources can be assigned, capturing dependence, forecasting, mapping w ork to requirements, data collection, tracking progress to commitments and reporting status. The planning and tracking capability of a system matures as the automation and integration of associated practices increases.
Planning & Tracking Level 1	Manual practices, with possible use of some rudimental tools, that collect and store information used to track and report status and outputs from planning and tracking activities.
Planning & Tracking Level 2	• Planning and tracking tools are used to define tasks and activities, along w ith the resources needed to perform them, required to achieve an objective, or commitment, and track progress, or lack thereof, tow ards achieving the given objective. • The tools provide the ability to capture and associate planning and tracking metadata, such as estimates, assumptions, prioritization, assignment, status, commitments, assets, association to implementation elements and supporting artifacts, and agreements. Metadata may consist of mostly manually collected information, w ith minimal automation. • Automated visualization techniques are used to organize activities, understand dependencies, coordinate multi-team efforts, and road mapping future commitments. The automated system is used to relevant stakeholders to share project plans and status of current activities with relevant stakeholders.
Planning & Tracking Level 3	• The planning and tracking tools are able to coordinate multiple value streams at the organizational level. Planning and tracking activities are integrated to include both technical and non-technical activities, such as quality assurance, documentation, testing and configuration management. Dependencies betw een technical and non-technical activities can be visualized in order to coordinate efforts and identify issues. • Metadata is used to support estimation, projections and w hat-if scenarios simulations. Organizations, projects and teams are able customized metadata, and associated use, in order to meet relevant stakeholder needs. • The planning and tracking tools are integrated w ith other tools in order to automatically collect metadata associated w ith various value stream activities. This includes defect, issues, and non-compliance efforts as they are automatically discovered and subsequently addressed and tracked to closure and asset management. • Automated stakeholder notification and status reporting, and associated visualizations, are used to notify relevant stakeholders of changes to plan or commitments, status of current activities, deviations from defined thresholds, and asset renew als and maintenance.
Planning & Tracking Level 4	Data is used to: • apply statistical analytical methods to planning and tracking practices in order to improve and optimize the team's, project's, and organization's ability to meet objectives and commitments • provide objective quantitative status to relevant stakeholders • automatically generate tasking and execute processes based on plan.
Quality Assurance	Quality Assurance is a set of independent activities (i.e., free from technical, managerial, and financial influences, intentional or unintentional) designed to provide confidence to relevant stakeholders that the DevSecOps processes and tools are appropriate for and produce products and services of suitable quality for their intended purposes. It assumes that the organization's, team's, and project's policies and procedures have been defined based on all relevant stakeholder needs w hich will result in a value stream that consistently produces products and services that meet all relevant stakeholder expectations. The

Name	Documentation
	quality assurance capability of a System matures as its ability to assess adherence to, and the adequacy of the defined policies and procedures.
Quality Assurance Level 1	• All relevant stakeholders associated with the products and services associated with the product under development and the system that support it have been identified. • All relevant stakeholder expectations and regulatory requirements are documented. • Policies and procedures are developed and documented to describe how the DevSecOps processes and tools are required to be used in order to meet all relevant stakeholder requirements. • Documented policies and procedures may use traditional document centered approach and dissemination may be a manual process. • All current policies and procedure are readily available to all personnel
Quality Assurance Level 2	• Automated tools are used to maintain configuration control of policies and procedures • All relevant stakeholders are automatically notified of changes to policies and procedures • Independent resources have been identified and a plan exists to review or audit activities that have been defined within the documented policies and procedures • DevSecOps processes and tools are periodically audited based on the plan to identify non-compliance with policies and procedures and inadequacies regarding the value stream's ability to consistently produce products and services which meet all relevant stakeholders' expectation and regulatory requirements. The audits may be conducted manually, use automation, or a combination pf both. • All identified non-compliance and inadequacies are independently documented, reported to relevant stakeholders, and tracked to closure.
Quality Assurance Level 3	• DevSecOps tools are configured to automatically enforce policies and procedures as a product under development progresses through the system. • Automated processes are monitored by an independent resource in order to detect and report noncompliance issues to all relevant stakeholders • Non-compliance and inadequacy issues identified through automated, or manual, auditing are documented and tracked to closure using an automated issue tracking system that is consistent with the tools used for all other planning and tracking purposes, in order to integrate all efforts that must be planned and tracked to completion. • All quality assurance tools, such as origin and static analysis tools, are fully integrated into the system's pipeline and associated policies are automatically enforced as the product under development progresses through the system. • The System automatically monitors and enforces compliance to defined quality criteria as defined for both the product under development and the system regarding the implementation of enhancements and modifications.
Quality Assurance Level 4	• All automated activities are continuously being audit for non-compliance issues through the use of automated tools, with regards to both the System and Product under development. • Results from the automated auditing tools are automatically reported to all relevant stakeholders to ensure the quality of the automated auditing process, in addition to tracking non-compliance issues to resolution. • The system shall automatically identify and track when the defined quality criteria has not been met or the automated quality controls have been bypassed. All relevant stakeholders will be automatically notified and the non-compliance issue will be tracked to closure.
Software Assurance	Software Assurance is the level of confidence that software functions only as intended and is free from vulnerabilities, either intentional or unintentionally designed or inserted as part of the software, throughout the full software lifecycle. It consists of two independent, but interrelated, assertions: 1. The software functions only as intended. It exhibits only functionality intended by its design and does not exhibit functionality not intended. 2. The software is free from vulnerabilities, whether intentionally or unintentionally present in the software, including software incorporated into the final system. It is the responsibility of the DevSecOps system to ensure that software that meets the organization's threshold for software assurance is allowed to be deployed and operated.
Software Assurance Level 1	• All relevant stakeholders and expectations with regards to the products and services associated with the product under development and the system that support it have been identified. • System functional and non-functional requirements are documented. • A comprehensive software bill of materials (SBOM) is compiled detailing all components that make up the DevSecOps system. • All relevant system constraints and regulatory requirements are documented. • Software assurance processes and tools are inventoried and policies and procedures written setting out how they are to be used to meet assurance requirements. • Documented policies and procedures may use traditional document centered approach and dissemination may be a manual process.

Name	Documentation
Softw are Assurance Level 2	• Softw are assurance related DevSecOps metrics are defined and collected. • Baseline and threshold levels for software assurance are established. • Metrics are tracked over time and made available to all stakeholders as needed. • Results of system functional testing are collected and periodically analyzed. • Know n vulnerabilities in all components that make up the DevSecOps system are periodically collected and analyzed. • Processes and polices are in place to periodically compare present metrics to past and make adjustments as necessary. • Processes and policies are in place and review ed periodically review reports from all software assurance products. • Processes and policies are in place to identify w hen the level of softw are assurance implied by captured metrics and reports exceeds the organization's threshold and to make adjustments as necessary.
Softw are Assurance Level 3	• The organization has established a comprehensive risk analysis and management program. • Softw are assurance metrics, reporting, and analysis are incorporated into the risk management process. • Results of the risk management process are incorporated into softw are assurance policies and procedures. • Softw are assurance metrics and thresholds are periodically updated as a result of risk management activities. • The organization prioritizes softw are assurance tasks based on the level of risk to the organization.
Softw are Assurance Level 4	• All softw are assurance tools, or as many as are feasible, are run continuously and reports disseminated automatically to all relevant stakeholders. • Softw are that fails to meet the organization's softw are assurance thresholds is automatically prevented from being delivered or deployed. • Automated procedures are in place to remediate softw are assurance issues found within the operating DevSecOps system.
Solution Development	Solutions development determines the best w ay of satisfying the requirements to achieve an outcome. Its goals are to: evaluate baseline requirements and alternative solutions to achieve them; select the optimum solution; create a specification for the solution. Each development value stream develops one or more solutions, w hich are products, services, or systems delivered to the customer, w hether internal or external to the Enterprise.
Solution Development Level 1	• All development activities and tools have been identified and documented • Provide tools to enable users to edit, compile, and review source code • Provide the ability for developers to trace links betw een requirements, architectural elements, and implementation elements • Provide a repository for all requirements and associated metadata • Provide manual processes for assuring security and privacy compliance • Processes for transitioning betw een development components are defined and documented • Individual processes are scripted and repeatable • Processes may require manual intervention betw een phases and/or to coordinate steps • Process initiation may be manual or automated
Solution Development Level 2	• Transitions betw een implementation elements, and supporting artifacts, are automated, possibly manually triggered • Identify and document secure coding practices and development coding standards • Provide traceability of softw are code origins to provide a SBOM and verify use of most recent third-party components
Solution Development Level 3	• Transitioning betw een development components are fully automated, either triggered on a periodic schedule or automatically triggered based upon completion of another component's activity • Support determination of requirement feasibility and validation analysis • Support model-based softw are engineering in order to provide continuous, iterative, and traceable requirements model • Support policy as code (e.g., STIG enforcement)
Solution Development Level 4	• Transitioning betw een development components is performed continuously w ithout human intervention • Continuously audit code commits, w ith alerts to relevant stakeholders • Enable "digital tw in" modeling of the production system • Support advanced analysis to ensure compliance
Verification & Validation	Verification and validation are the set of activities and evidence that the system or application under development has met the requirements and criteria that is expected. It includes the general realm of testing, verifying, and validation activities and matures as automation, feedback, and integration w ith other elements increase.
Verification & Validation Level 1	• All relevant stakeholders w ith regards to the products and services associated with the product under development and the system that support it have been identified.

Name	Documentation
	• All testing cases, procedures, and their artifacts are can be configured, stored, and maintained for a given instance of a product under development. • The system and product under development supports the necessary technologies to execute tests.
Verification & Validation Level 2	• Automated tools are used to trace tests to requirements. • Automated tools are used to trace tests cases and artifacts to specific versions of a product under development. • Automated tools are used to configure, store, and execute tests. • Test coverage reports are generated and captured for a specific instance of the system or product under development. • Tests are performed across multiple phases of the software lifecycle such as development, test, and operations providing feedback continuously. • Security patching is automatically tested, resulting in automated report generation and delivery. • Both functional and non-function tests are manually or automatically executed.
Verification & Validation Level 3	• Tests are executed automatically using a continuous integration technique. • A MBSE approach is used to plan and execute testing of the system and product under development. • The system and product under development automatically executes quality tests that either passes or fails the appropriate component under test based on quality metrics for any change being made. Appropriate monitoring of the system and product under development enforces the quality metrics. • The system provides the necessary environment to perform advanced security testing such as Fuzz, and Penetration testing activities.

The PIM defines DevSecOps pipeline capability as a socio-technical system composed of both software tools and processes. As the capability matures it seamlessly integrates three traditional factions that sometimes have opposing interests: development values features, security values defensibility, and operations values stability. A DevSecOps pipeline emerges when continuous integration of these three factions is used to meet organizational, project, and team objectives and commitments. Figure 9: DevSecOps Pipeline shows another view of the DevSecOps pipeline and the iterative processes and interactions it must support. Figure 59: Capabilities to Requirements Relationship Matrix provides a summary mapping of capabilities to requirements.

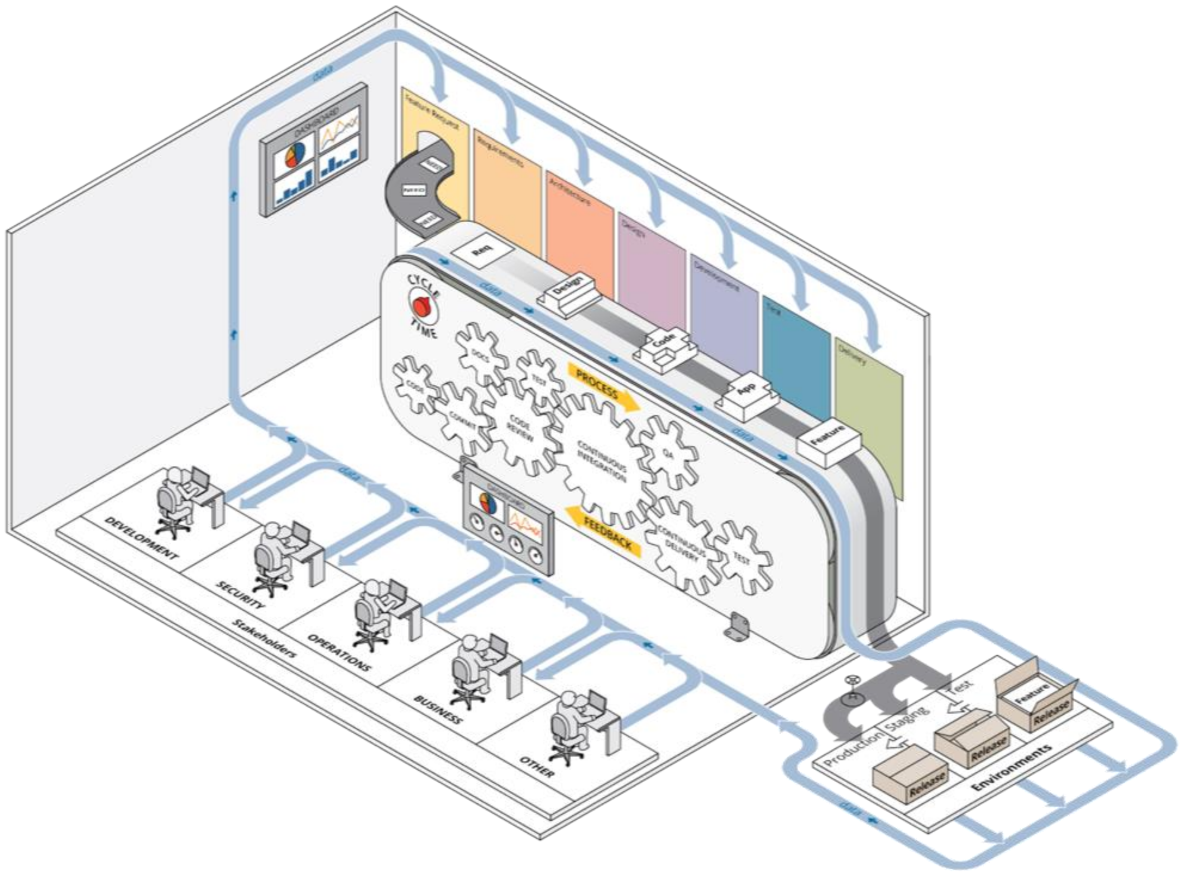


Figure 9: DevSecOps Pipeline

1.4.1 Getting Started

When using the model, it is important to start with Table 3: Glossary. The terms defined in the Glossary have specific definitions within the context of the model that go beyond the standard English definition of the word or phrase. Two key terms used throughout the model are "system" and "product under development." Once familiar with the terms in the glossary, the next step is to consider your use case for using the model. While there are several potential uses of the model, two of the most common are: (1) the creation of a new DevSecOps pipeline in support of a new product or in modernizing an existing product, and (2) evaluating an existing DevSecOps pipeline for areas of improvement. In either case it is important to first understand the vision of the product to be built and maintained as well as any regulatory or environmental constraints that will be put on the DevSecOps pipeline instantiation, as this will drive many of your decisions.

In use case 1, the software lifecycle view of the requirements, as shown in Figure 6: DevSecOps System Requirements, along with the maturity ratings will probably best suit your initial needs. Start with the maturity level 1 requirements. In general, maturity level 1 is focused on the basic engineering, security, and operational practices needed to start producing a product, even if done with minimal automation and integrated tooling (i.e., relying on manual processes). Level 2 is about actually being able to claim you are doing DevSecOps, as it is when automation and integrated tools and associated processes really start to come into play. With that said, it is important not to skip level 1 and go straight to level 2 or any higher level. Level 1 allows you to understand what you need, and it will guide your tool selection, configuration, automation, and integration decisions going forward. Remember, most organizations don't start by fully automating everything. Instead, the automation of processes is realized over time. The model's DevSecOps system requirements perspective organizes the requirements in a way that instinctively maps them to a product lifecycle, thus forming a natural grouping of concerns based on traditional roles within an organization and the tools used in supporting associated activities.

For use case 2, the focus is on improving an existing DevSecOps pipeline which means improving existing capabilities and identifying missing or deficient capabilities. In this case, the DevSecOps Capabilities view as shown in Figure 8: DevSecOps Capabilities, along with the capability levels, will probably best suit your needs. This will allow you to look holistically at the pipeline and determine which lower-level capabilities are lacking (i.e., which are the weakest link) and potentially holding back the higher-level capability in terms of throughput, quality, security, and other quality attributes.

2 Dictionaries

2.1 DevSecOps Maturity Levels

Table 2: DevSecOps Maturity Levels

Term	Documentation
Maturity Level 1	Performed Basic Practices: This represents the minimum set of engineering, security, and operational practices that is required to begin supporting a product under development, even if only performed in an ad-hoc manner with minimal automation, documentation, or process maturity. This level is focused on minimal development, security and operational hygiene.
Maturity Level 2	Documented/Automated Intermediate Practices: In addition to meeting the level 1 practices. This level represents the transition from manual ad-hoc practices to the automated and consistent execution of defined processes. This set of practices represents the next evolution the maturity of the product under development's pipeline by providing the capability needed to automate the practices that are most often executed or produce the most unpredictable results. These practices include defining process that enable individuals to perform activities in a repeatable manner.
Maturity Level 3	Managed Pipeline Execution: In addition to meeting the level 1 and 2 practices. This level focuses on consistently meeting the information needs of all relevant stakeholders associated with the product under development so that they can make informed decisions as work items progress through a defined process.
Maturity Level 4	Proactive Reviewing and Optimizing DevSecOps: In addition to meeting the level 1-3 practices. This level is focused on reviewing the effectiveness of the system so that corrective actions are taken, when necessary, as well as quantitively improving the system's performance as it relates to the consistent development and operation of the product under development.

2.2 Glossary

Table 3: Glossary

Term	Description
Chain of Custody	Everything that happens in the system as it relates to a configuration items (CI). A record of chain of custody provides evidence of what is in the CI, how it has been changed or modified, by whom, and when in sufficient details that the CI could be recreated. The evidence is immutable and provides a sufficient level of detail to satisfy audit and regulatory requirements.
Code Coverage	code coverage is a measure used to describe the degree to which the source code of a program is executed when a particular test suite runs [Wikipedia 2021].
Common Vulnerabilities and Exposures (CVE)	Identifies, defines, and categorizes publicly disclosed cybersecurity vulnerabilities [The MITRE Corporation 2021a].
Common Weakness Enumeration (CWE)	Common Weakness Enumeration (CWE) is a community-developed list of software and hardware weakness types. It serves as a common language, a measuring stick for security tools, and as a baseline for weakness identification, mitigation, and prevention efforts [The MITRE Corporation 2021b].
Continuous Integration	Integration is the process of merging changes from multiple developers made to a single code base. Continuous Integration is the process of merging changes from multiple developers, conducting integration tests, staging the system for acceptance testing, and potentially staging the product for final delivery, in an automated fashion in real or near real time as changes are made.
DevSecOps	A cultural and engineering practice that breaks down barriers and opens collaboration between development, security, and operations organizations using automation to focus on rapid, frequent delivery of secure infrastructure and software to production. It encompasses intake to release of software and manages those flows predictably, transparently, and with minimal human intervention/effort [U.S. General Services Administration 2021].
Implementation Elements	Physical or digital components of the realized system or product under development used to achieve a capability or set of capabilities. Elements of the implementation are distinct from supporting artifacts and elements, such as architecture, design, test, analysis, requirements, reports, etc.
ITSM service desk	The Information Technology Service Management (ITSM) service desk is the single point of contact between the service provider and the service consumer or user.
Kanban board	A Kanban board is used to manage work at a personal, team or organization level. They visually depict work at various stages of a process using cards to represent work items and columns to represent each stage of the process. Cards are moved from left to right to show progress and to help coordinate teams performing the work. A Kanban board may be divided into horizontal "swimlanes" representing different kinds of work or different individuals, teams or organizations performing the work [Wikipedia 2021d].
Knowledge Management	The process of creating, sharing, using and managing the knowledge and information of an organization [Girard 2015].
MBSE	Model-based systems engineering (MBSE) is a formalized methodology that is used to support the requirements, design, analysis, verification, and validation associated with the development of complex systems [Shevchenko 2020]
Minimally Viable Capability Release	The initial set of features suitable to be fielded to an operational environment that provides value to the end user in a rapid time line. The minimally viable capability release (MVCR) delivers initial end user capabilities to enhance some mission outcomes. The MVCR is analogous to a minimum marketable product [U.S. Department of Defense 2020].
Minimum Viable Product	An early version of the software to deliver or field basic capabilities to users to evaluate and provide feedback on. Insights from minimum viable products (MVPs) help shape scope, requirements, and design [U.S. Department of Defense 2020].
Monitor and Control	Monitoring and Control is the continuous monitoring of project activities and the enactment of corrective action(s). Measures are used to determine progress by comparing current status to expected status or behavior. When the project deviates significantly from what was expected, appropriate corrective actions are taken. Moreover, learning and innovation occur and improvements are made from leveraging knowledge learned from continuous feedback.
Orchestration System	A tool or collection of tools used to automatically coordinate and execute many tasks together in order to streamline and optimize frequent, repeatable processes, with an expected level of assurance.
Organization	An administrative structure in which people collectively manage one or more projects or workgroups as a whole, share a senior manager, and operate under the same policies [Chrissis 2011]

Term	Description
Product Under Development	For the scope of this model, product under development is defined as the specialized product, component, application, or bundled applications being built and/or maintained by the system in order to meet the needs of a specific end use.
Project	For the scope of this model, project applies broadly to any managed set of interrelated activities and resources, including people, that delivers one or more products or services to a customer or end user [Chrissis 2011]
Quality Assurance	A strategic and systematic approach to monitoring the engineering tools, practices, and processes used to ensure the quality of a product under development in order to assure relevant stakeholders that the product under development will fulfill relevant stakeholders' expectations and regulatory requirements. Expectations are ideally explicitly stated through service level agreements, requirements, goals, etc., and not simply implied.
Relevant Stakeholders	A Stakeholder is a group or individual that is affected by or is in some way accountable for the outcome of an undertaking. A relevant stakeholder is a stakeholder that is identified for involvement in specific activities and is included in a plan [Chrissis 2011]. It includes technical staff of various domains, operational users and representatives, and business units such as legal, contracts, finance, compliance, privacy, and security. The group or individual can be internal or external to the organization.
Requirements	1) A condition or capability needed by a stakeholder to solve a problem or achieve an objective. 2) A condition or capability that must be met or possessed by a solution or solution component to satisfy a contract, standard, specification or other formally imposed documents. 3) A documented representation of a condition or capability as in (1) or (2).
Scrum Board	A Scrum board is a tool used to visually display current project work, specifically work that has been taken into the current sprint. At a high level, it shows what has not been started, what is currently being worked on, and what has been completed. Virtual boards use software designed to look like the physical boards, but they are viewed and changed electronically. Virtual board layouts can be customized based on the target audience (i.e. the people doing the work or relevant stakeholders who want to know the progress of the effort) [Study.com 2020].
Security and Privacy Engineering Principles	A set of principles defined in the System and Services Acquisition (SA-8) control family [Joint Task Force Transformation Initiative Interagency Working Group 2020] that are used in the specification, design, development, implementation, and modification of the system, product under development, and associated sub-components.
Software Assurance	Software Assurance is the practice of ensuring that a piece of software, or complete software-centric system, functions only as intended and is free of known vulnerabilities. It is a holistic practice that incorporates software requirements, architecture, testing, vulnerability management, risk management and operational policies and procedures to reduce, and ideally eliminate, risk to the organization's mission as a result of software defects.
System	For the scope of this model, system is defined as the set of people, processes, tools and technology working together as part of an interconnected DevSecOps network designed to collect, process, store, evaluate, deliver, deploy, and monitor a product under development and all associated artifacts.
Team	A group of people with complementary skills and expertise who work together to accomplish specified objectives [Chrissis 2011]
Test Artifacts	Tangible by-products that are generated automatically, manually, or a combination of both, while planning, performing, and reporting on testing activities in order to (1) verify and validate the product under development, (2) monitor and verify the DevSecOps process, and (3) establish transparency between members of a project team and all relevant stakeholders. Thus, all by-products must contain accurate information and details.
Trace	A logical link that specifies a relationship between two or more entities.
Traceability	The ability to determine a set of logical links between two or more entities across the development lifecycle.
Value Streams	Value streams represent the series of steps that an organization uses to implement solutions that provide a continuous flow of value to a customer [Scaled Agile, Inc. 2021].

3 System Requirements

3.1 1 Governance

3.1.1 Gov_1 Track Changes Associated to Requirements

The system shall be able to track and associate any changes to the system or product under development to a given requirement, either functional or non-functional, and, if applicable, change request (CR) through the development lifecycle.

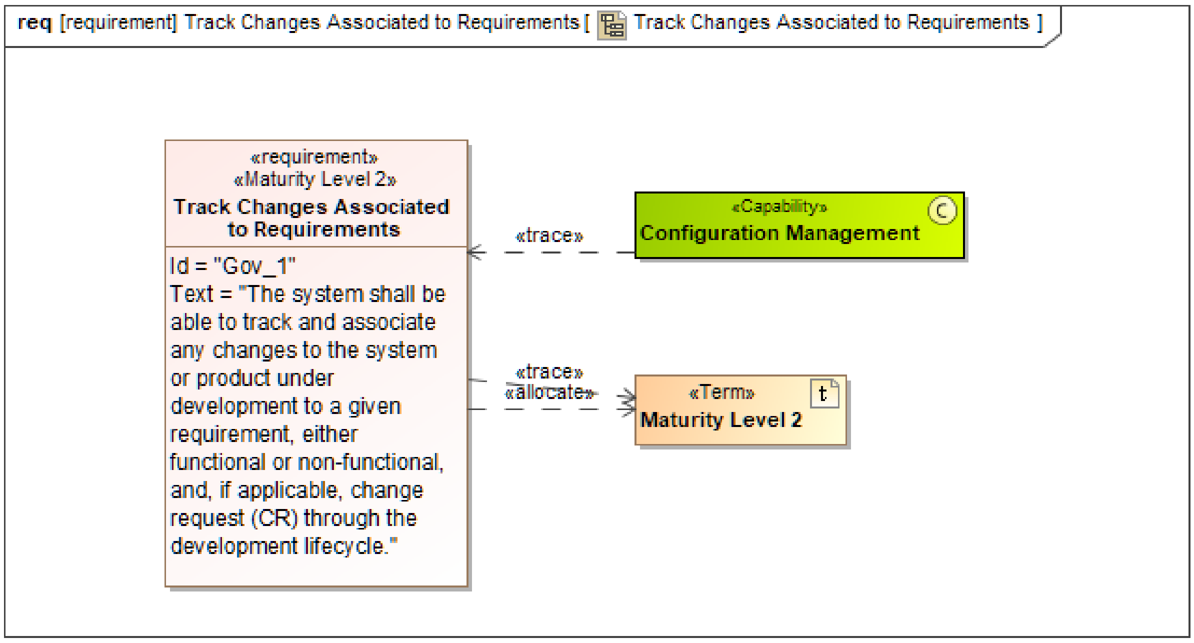
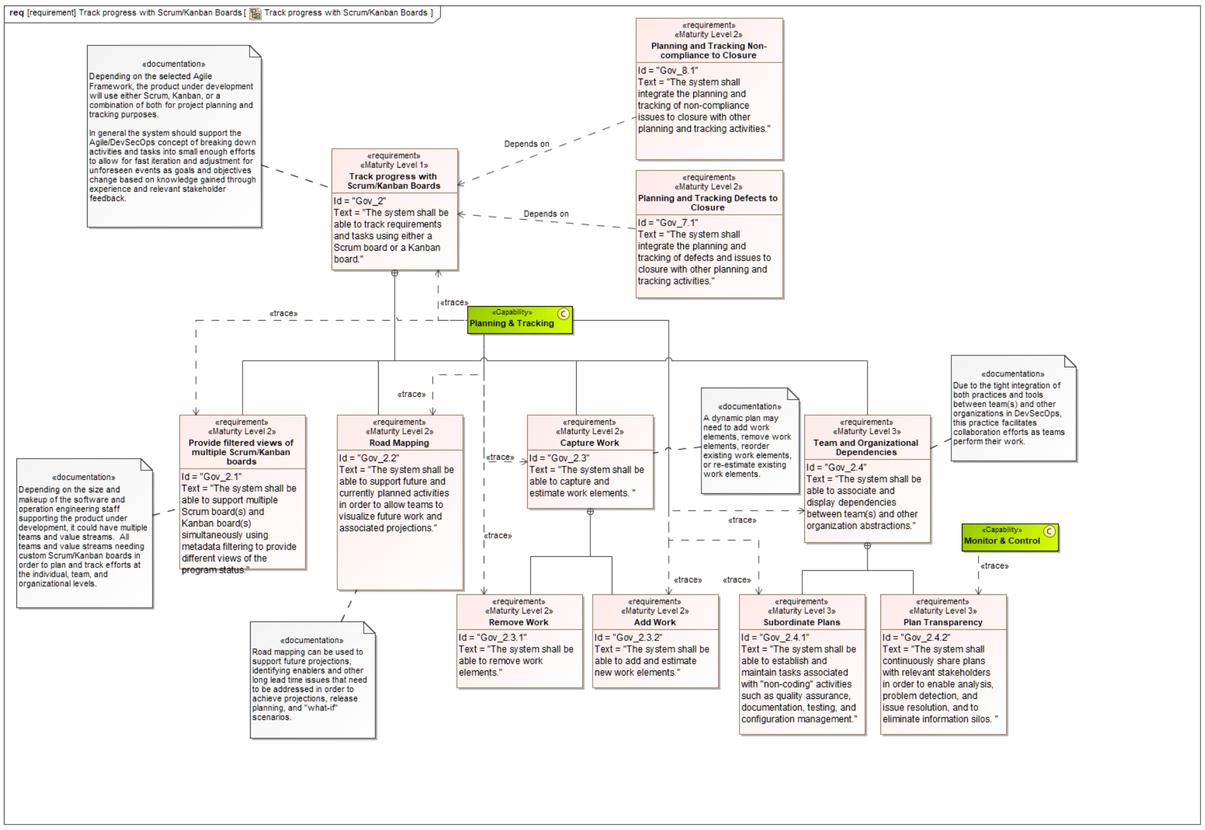

req [requirement] Track Changes Associated to Requirements [Track Changes Associated to Requirements]
«requirement»
«Maturity Level 2»
Track Changes Associated to Requirements
Id = "Gov_1"
Text = "The system shall be able to track and associate any changes to the system or product under development to a given requirement, either functional or non-functional, and, if applicable, change request (CR) through the development lifecycle."
«trace»
«Capability»
Configuration Management
«trace»
«allocate»
«Term»
Maturity Level 2

Figure 10: Track Changes Associated to Requirements

3.1.2 Gov_2 Track progress with Scrum/Kanban Boards

The system shall be able to track requirements and tasks using either a Scrum board or a Kanban board.


req [requirement] Track progress with Scrum/Kanban Boards [Track progress with Scrum/Kanban Boards]
Depends on
Depends on
«trace»
«trace»
«trace»
«trace»
«trace»
«trace»
«trace»
«trace»
«trace»
«Capability» Planning & Tracking
«Capability» Monitor & Control

«documentation»
Depending on the selected Agile Framework, the product under development will use either Scrum, Kanban, or a combination of both for project planning and tracking purposes.

In general the system should support the Agile/DevSecOps concept of breaking down activities and tasks into small enough efforts to allow for fast iteration and adjustment for unforeseen events as goals and objectives change based on knowledge gained through experience and relevant stakeholder feedback.

«requirement»
«Maturity Level 1»
Track progress with Scrum/Kanban Boards
Id = "Gov_2"
Text = "The system shall be able to track requirements and tasks using either a Scrum board or a Kanban board."

«requirement»
«Maturity Level 2»
Planning and Tracking Non-compliance to Closure
Id = "Gov_8.1"
Text = "The system shall integrate the planning and tracking of non-compliance issues to closure with other planning and tracking activities."

«requirement»
«Maturity Level 2»
Planning and Tracking Defects to Closure
Id = "Gov_7.1"
Text = "The system shall integrate the planning and tracking of defects and issues to closure with other planning and tracking activities."

«requirement»
«Maturity Level 2»
Provide filtered views of multiple Scrum/Kanban boards
Id = "Gov_2.1"
Text = "The system shall be able to support multiple Scrum board(s) and Kanban board(s) simultaneously using metadata filtering to provide different views of the program status."

«documentation»
Depending on the size and makeup of the software and operation engineering staff supporting the product under development, it could have multiple teams and value streams. All teams and value streams needing custom Scrum/Kanban boards in order to plan and track efforts at the individual, team, and organizational levels.

«requirement»
«Maturity Level 2»
Road Mapping
Id = "Gov_2.2"
Text = "The system shall be able to support future and currently planned activities in order to allow teams to visualize future work and associated projections."

«documentation»
Road mapping can be used to support future projections, identifying enablers and other long lead time issues that need to be addressed in order to achieve projections, release planning, and "what-if" scenarios.

«requirement»
«Maturity Level 2»
Capture Work
Id = "Gov_2.3"
Text = "The system shall be able to capture and estimate work elements. "

«documentation»
A dynamic plan may need to add work elements, remove work elements, reorder existing work elements, or re-estimate existing work elements.

«requirement»
«Maturity Level 3»
Team and Organizational Dependencies
Id = "Gov_2.4"
Text = "The system shall be able to associate and display dependencies between team(s) and other organization abstractions."

«documentation»
Due to the tight integration of both practices and tools between team(s) and other organizations in DevSecOps, this practice facilitates collaboration efforts as teams perform their work.

«requirement»
«Maturity Level 2»
Remove Work
Id = "Gov_2.3.1"
Text = "The system shall be able to remove work elements."

«requirement»
«Maturity Level 2»
Add Work
Id = "Gov_2.3.2"
Text = "The system shall be able to add and estimate new work elements."

«requirement»
«Maturity Level 3»
Subordinate Plans
Id = "Gov_2.4.1"
Text = "The system shall be able to establish and maintain tasks associated with "non-coding" activities such as quality assurance, documentation, testing, and configuration management."

«requirement»
«Maturity Level 3»
Plan Transparency
Id = "Gov_2.4.2"
Text = "The system shall continuously share plans with relevant stakeholders in order to enable analysis, problem detection, and issue resolution, and to eliminate information silos. "

Figure 11: Track Progress with Scrum/Kanban Boards

Documentation:

Depending on the selected Agile Framework, the product under development will use either Scrum, Kanban, or a combination of both for project planning and tracking purposes.

In general, the system should support the Agile/DevSecOps concept of breaking down activities and tasks into small enough efforts to allow for fast iteration and adjustment for unforeseen events as goals and objectives change based on knowledge gained through experience and relevant stakeholder feedback.

3.1.2.1 Gov_2.1 Provide filtered views of multiple Scrum/Kanban boards

The system shall be able to support multiple Scrum board(s) and Kanban board(s) simultaneously using metadata filtering to provide different views of the program status.

Documentation:

Depending on the size and makeup of the software and operation engineering staff supporting the product under development, it could have multiple teams and value streams. All teams and value streams needing custom Scrum/Kanban boards in order to plan and track efforts at the individual, team, and organizational levels.

3.1.2.2 Gov_2.2 Road Mapping

The system shall be able to support future and currently planned activities in order to allow teams to visualize future work and associated projections.

Documentation:

Road mapping can be used to support future projections, identifying enablers and other long lead time issues that need to be addressed in order to achieve projections, release planning, and "what-if" scenarios.

3.1.2.3 Gov_2.3 Capture Work

The system shall be able to capture and estimate work elements.

Documentation:

A dynamic plan may need to add work elements, remove work elements, reorder existing work elements, or re-estimate existing work elements.

3.1.2.3.1 Gov_2.3.1 Remove Work

The system shall be able to remove work elements.

3.1.2.3.2 Gov_2.3.2 Add Work

The system shall be able to add and estimate new work elements.

3.1.2.4 Gov_2.4 Team and Organizational Dependencies

The system shall be able to associate and display dependencies between team(s) and other organization abstractions.

Documentation:

Due to the tight integration of both practices and tools between team(s) and other organizations in DevSecOps, this practice facilitates collaboration efforts as teams perform their work.

3.1.2.4.1 Gov_2.4.1 Subordinate Plans

The system shall be able to establish and maintain tasks associated with "non-coding" activities such as quality assurance, documentation, testing, and configuration management.

3.1.2.4.2 Gov_2.4.2 Plan Transparency

The system shall continuously share plans with relevant stakeholders in order to enable analysis, problem detection, and issue resolution, and to eliminate information silos.

3.1.3 Gov_3 Task Creation

The system shall be able to capture tasks for doing work.

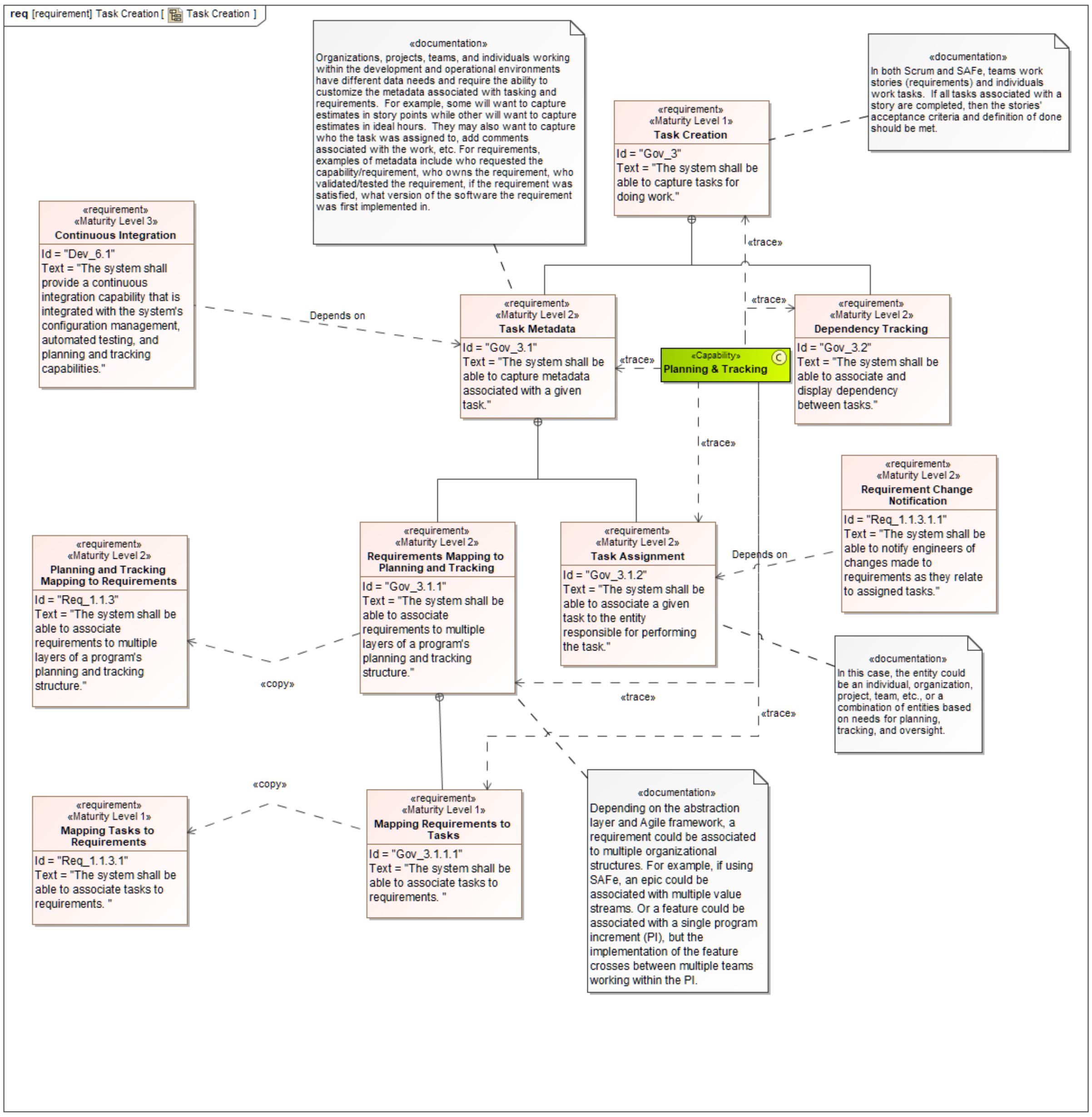

req [requirement] Task Creation [Task Creation]
«documentation» Organizations, projects, teams, and individuals working within the development and operational environments have different data needs and require the ability to customize the metadata associated with tasking and requirements. For example, some will want to capture estimates in story points while other will want to capture estimates in ideal hours. They may also want to capture who the task was assigned to, add comments associated with the work, etc. For requirements, examples of metadata include who requested the capability/requirement, who owns the requirement, who validated/tested the requirement, if the requirement was satisfied, what version of the software the requirement was first implemented in.
«requirement» «Maturity Level 1» Task Creation Id = "Gov_3" Text = "The system shall be able to capture tasks for doing work."
«documentation» In both Scrum and SAFe, teams work stories (requirements) and individuals work tasks. If all tasks associated with a story are completed, then the stories' acceptance criteria and definition of done should be met.
«requirement» «Maturity Level 3» Continuous Integration Id = "Dev_6.1" Text = "The system shall provide a continuous integration capability that is integrated with the system's configuration management, automated testing, and planning and tracking capabilities."
Depends on
«requirement» «Maturity Level 2» Task Metadata Id = "Gov_3.1" Text = "The system shall be able to capture metadata associated with a given task."
«trace»
«Capability» Planning & Tracking
«requirement» «Maturity Level 2» Dependency Tracking Id = "Gov_3.2" Text = "The system shall be able to associate and display dependency between tasks."
«requirement» «Maturity Level 2» Requirement Change Notification Id = "Req_1.1.3.1.1" Text = "The system shall be able to notify engineers of changes made to requirements as they relate to assigned tasks."
«requirement» «Maturity Level 2» Planning and Tracking Mapping to Requirements Id = "Req_1.1.3" Text = "The system shall be able to associate requirements to multiple layers of a program's planning and tracking structure."
«requirement» «Maturity Level 2» Requirements Mapping to Planning and Tracking Id = "Gov_3.1.1" Text = "The system shall be able to associate requirements to multiple layers of a program's planning and tracking structure."
«requirement» «Maturity Level 2» Task Assignment Id = "Gov_3.1.2" Text = "The system shall be able to associate a given task to the entity responsible for performing the task."
«documentation» In this case, the entity could be an individual, organization, project, team, etc., or a combination of entities based on needs for planning, tracking, and oversight.
«copy»
«requirement» «Maturity Level 1» Mapping Tasks to Requirements Id = "Req_1.1.3.1" Text = "The system shall be able to associate tasks to requirements. "
«requirement» «Maturity Level 1» Mapping Requirements to Tasks Id = "Gov_3.1.1.1" Text = "The system shall be able to associate tasks to requirements. "
«documentation» Depending on the abstraction layer and Agile framework, a requirement could be associated to multiple organizational structures. For example, if using SAFe, an epic could be associated with multiple value streams. Or a feature could be associated with a single program increment (PI), but the implementation of the feature crosses between multiple teams working within the PI.

Figure 12: Task Creation

Documentation:

In both Scrum and SAFe, teams work stories (requirements) and individuals work tasks. If all tasks associated with a story are completed, then the stories' acceptance criteria and definition of done should be met.

3.1.3.1 Gov_3.1 Task Metadata

The system shall be able to capture metadata associated with a given task.

Documentation:

Organizations, projects, teams, and individuals working within the development and operational environments have different data needs and require the ability to customize the metadata associated with tasking and requirements. For example, some will want to capture estimates in story points while other will want to capture estimates in ideal hours. They may also want to capture who the task was assigned to, add comments associated with the work, etc. For requirements, examples of metadata include who requested the capability/requirement, who owns the requirement, who validated/tested the requirement, if the requirement was satisfied, what version of the software the requirement was first implemented in.

3.1.3.1.1 Gov_3.1.1 Requirements Mapping to Planning and Tracking

The system shall be able to associate requirements to multiple layers of a program's planning and tracking structure.

Documentation:

Depending on the abstraction layer and Agile framework, a requirement could be associated to multiple organizational structures. For example, if using SAFe, an epic could be associated with multiple value streams. Or a feature could be associated with a single program increment (PI), but the implementation of the feature crosses between multiple teams working within the PI.

3.1.3.1.1.1 Gov_3.1.1.1 Mapping Requirements to Tasks

The system shall be able to associate tasks to requirements.

3.1.3.1.2 Gov_3.1.2 Task Assignment

The system shall be able to associate a given task to the entity responsible for performing the task.

Documentation:

In this case, the entity could be an individual, organization, project, team, etc., or a combination of entities based on needs for planning, tracking, and oversight.

3.1.3.2 Gov_3.2 Dependency Tracking

The system shall be able to associate and display dependency between tasks.

3.1.4 Gov_4 Metrics

The system shall be able to automatically collect, correlate and display metrics associated with productivity, reliability, quality, security, and operations of both the system and the product under development and associated technologies, approaches, and methods, and models used throughout the product's life in order to achieve an organization's objectives and business needs.

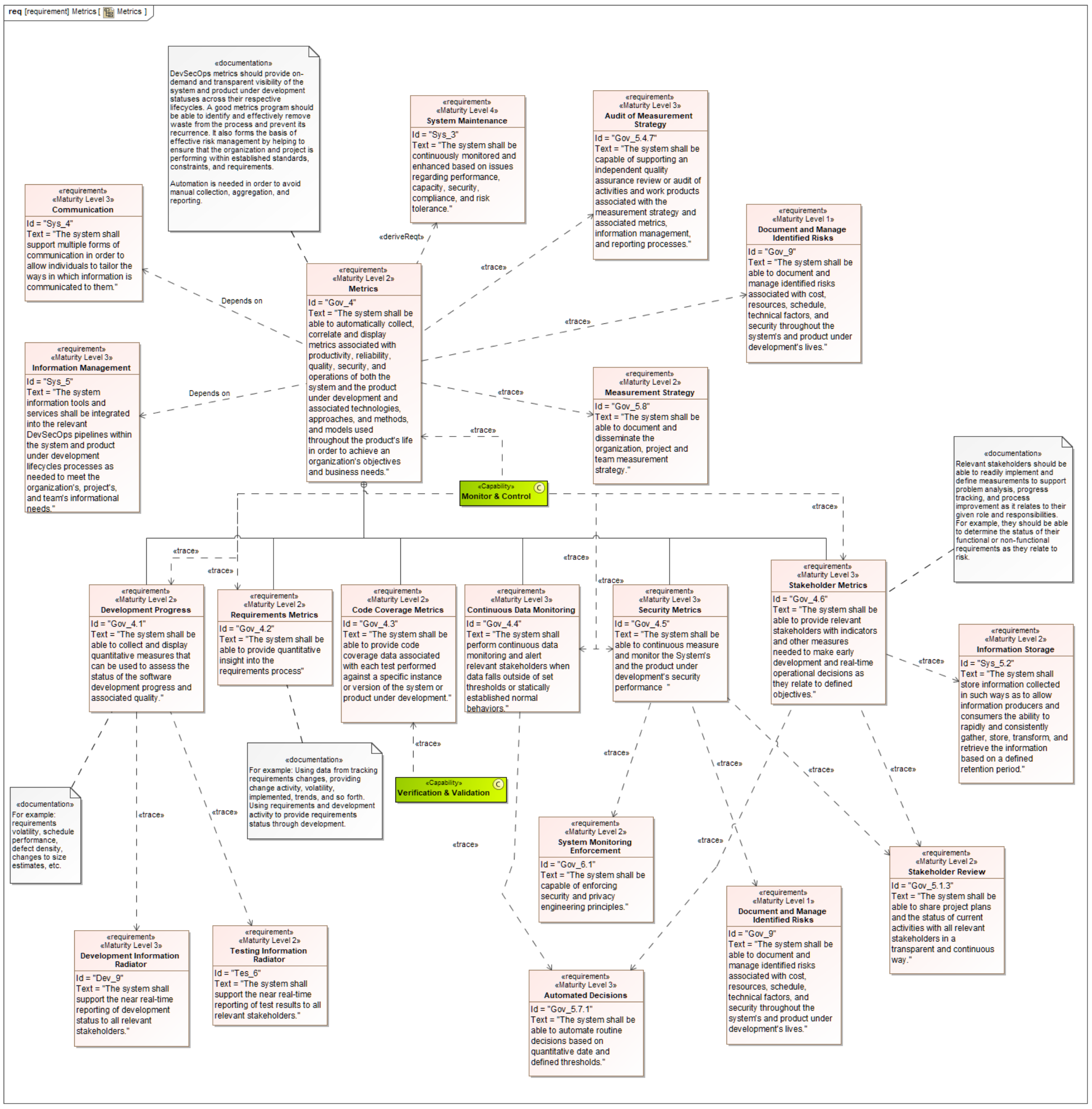

req [requirement] Metrics [Metrics]
«documentation» DevSecOps metrics should provide on-demand and transparent visibility of the system and product under development statuses across their respective lifecycles. A good metrics program should be able to identify and effectively remove waste from the process and prevent its recurrence. It also forms the basis of effective risk management by helping to ensure that the organization and project is performing within established standards, constraints, and requirements. Automation is needed in order to avoid manual collection, aggregation, and reporting.
«requirement» «Maturity Level 4» System Maintenance Id = "Sys_3" Text = "The system shall be continuously monitored and enhanced based on issues regarding performance, capacity, security, compliance, and risk tolerance."
«requirement» «Maturity Level 3» Audit of Measurement Strategy Id = "Gov_5.4.7" Text = "The system shall be capable of supporting an independent quality assurance review or audit of activities and work products associated with the measurement strategy and associated metrics, information management, and reporting processes."
«requirement» «Maturity Level 3» Communication Id = "Sys_4" Text = "The system shall support multiple forms of communication in order to allow individuals to tailor the ways in which information is communicated to them."
«requirement» «Maturity Level 1» Document and Manage Identified Risks Id = "Gov_9" Text = "The system shall be able to document and manage identified risks associated with cost, resources, schedule, technical factors, and security throughout the system's and product under development's lives."
«deriveReqt»
«trace»
Depends on
«requirement» «Maturity Level 2» Metrics Id = "Gov_4" Text = "The system shall be able to automatically collect, correlate and display metrics associated with productivity, reliability, quality, security, and operations of both the system and the product under development and associated technologies, approaches, and methods, and models used throughout the product's life in order to achieve an organization's objectives and business needs."
«requirement» «Maturity Level 3» Information Management Id = "Sys_5" Text = "The system information tools and services shall be integrated into the relevant DevSecOps pipelines within the system and product under development lifecycles processes as needed to meet the organization's, project's, and team's informational needs."
«requirement» «Maturity Level 2» Measurement Strategy Id = "Gov_5.8" Text = "The system shall be able to document and disseminate the organization, project and team measurement strategy."
«Capability» Monitor & Control
«documentation» Relevant stakeholders should be able to readily implement and define measurements to support problem analysis, progress tracking, and process improvement as it relates to their given role and responsibilities. For example, they should be able to determine the status of their functional or non-functional requirements as they relate to risk.
«requirement» «Maturity Level 2» Development Progress Id = "Gov_4.1" Text = "The system shall be able to collect and display quantitative measures that can be used to assess the status of the software development progress and associated quality."
«requirement» «Maturity Level 2» Requirements Metrics Id = "Gov_4.2" Text = "The system shall be able to provide quantitative insight into the requirements process"
«requirement» «Maturity Level 2» Code Coverage Metrics Id = "Gov_4.3" Text = "The system shall be able to provide code coverage data associated with each test performed against a specific instance or version of the system or product under development."
«requirement» «Maturity Level 3» Continuous Data Monitoring Id = "Gov_4.4" Text = "The system shall perform continuous data monitoring and alert relevant stakeholders when data falls outside of set thresholds or statically established normal behaviors."
«requirement» «Maturity Level 3» Security Metrics Id = "Gov_4.5" Text = "The system shall be able to continuous measure and monitor the System's and the product under development's security performance "
«requirement» «Maturity Level 3» Stakeholder Metrics Id = "Gov_4.6" Text = "The system shall be able to provide relevant stakeholders with indicators and other measures needed to make early development and real-time operational decisions as they relate to defined objectives."
«requirement» «Maturity Level 2» Information Storage Id = "Sys_5.2" Text = "The system shall store information collected in such ways as to allow information producers and consumers the ability to rapidly and consistently gather, store, transform, and retrieve the information based on a defined retention period."
«documentation» For example: Using data from tracking requirements changes, providing change activity, volatility, implemented, trends, and so forth. Using requirements and development activity to provide requirements status through development.
«Capability» Verification & Validation
«documentation» For example: requirements volatility, schedule performance, defect density, changes to size estimates, etc.
«requirement» «Maturity Level 2» System Monitoring Enforcement Id = "Gov_6.1" Text = "The system shall be capable of enforcing security and privacy engineering principles."
«requirement» «Maturity Level 2» Stakeholder Review Id = "Gov_5.1.3" Text = "The system shall be able to share project plans and the status of current activities with all relevant stakeholders in a transparent and continuous way."
«requirement» «Maturity Level 1» Document and Manage Identified Risks Id = "Gov_9" Text = "The system shall be able to document and manage identified risks associated with cost, resources, schedule, technical factors, and security throughout the system's and product under development's lives."
«requirement» «Maturity Level 3» Development Information Radiator Id = "Dev_9" Text = "The system shall support the near real-time reporting of development status to all relevant stakeholders."
«requirement» «Maturity Level 2» Testing Information Radiator Id = "Tes_6" Text = "The system shall support the near real-time reporting of test results to all relevant stakeholders."
«requirement» «Maturity Level 3» Automated Decisions Id = "Gov_5.7.1" Text = "The system shall be able to automate routine decisions based on quantitative date and defined thresholds."

Figure 13: Metrics

Documentation:

DevSecOps metrics should provide on-demand and transparent visibility of the system and product under development statuses across their respective lifecycles. A good metrics program should be able to identify and effectively remove waste from the process and prevent its recurrence. It also forms the basis of effective risk management by helping to ensure that the organization and project is performing within established standards, constraints, and requirements.

Automation is needed in order to avoid manual collection, aggregation, and reporting.

3.1.4.1 Gov_4.1 Development Progress

The system shall be able to collect and display quantitative measures that can be used to assess the status of the software development progress and associated quality.

Documentation:

For example: requirements volatility, schedule performance, defect density, changes to size estimates, etc.

3.1.4.2 Gov_4.2 Requirements Metrics

The system shall be able to provide quantitative insight into the requirements process

Documentation:

For example: Using data from tracking requirements changes, providing change activity, volatility, implemented, trends, and so forth. Using requirements and development activity to provide requirements status through development.

3.1.4.3 Gov_4.3 Code Coverage Metrics

The system shall be able to provide code coverage data associated with each test performed against a specific instance or version of the system or product under development.

3.1.4.4 Gov_4.4 Continuous Data Monitoring

The system shall perform continuous data monitoring and alert relevant stakeholders when data falls outside of set thresholds or statically established normal behaviors.

3.1.4.5 Gov_4.5 Security Metrics

The system shall be able to continuous measure and monitor the System's and the product under development's security performance

3.1.4.6 Gov_4.6 Stakeholder Metrics

The system shall be able to provide relevant stakeholders with indicators and other measures needed to make early development and real-time operational decisions as they relate to defined objectives.

Documentation:

Relevant stakeholders should be able to readily implement and define measurements to support problem analysis, progress tracking, and process improvement as it relates to their given role and responsibilities. For example, they should be able to determine the status of their functional or non-functional requirements as they relate to risk.

3.1.5 Gov_5 Knowledge Management

The system shall provide integrated knowledge management

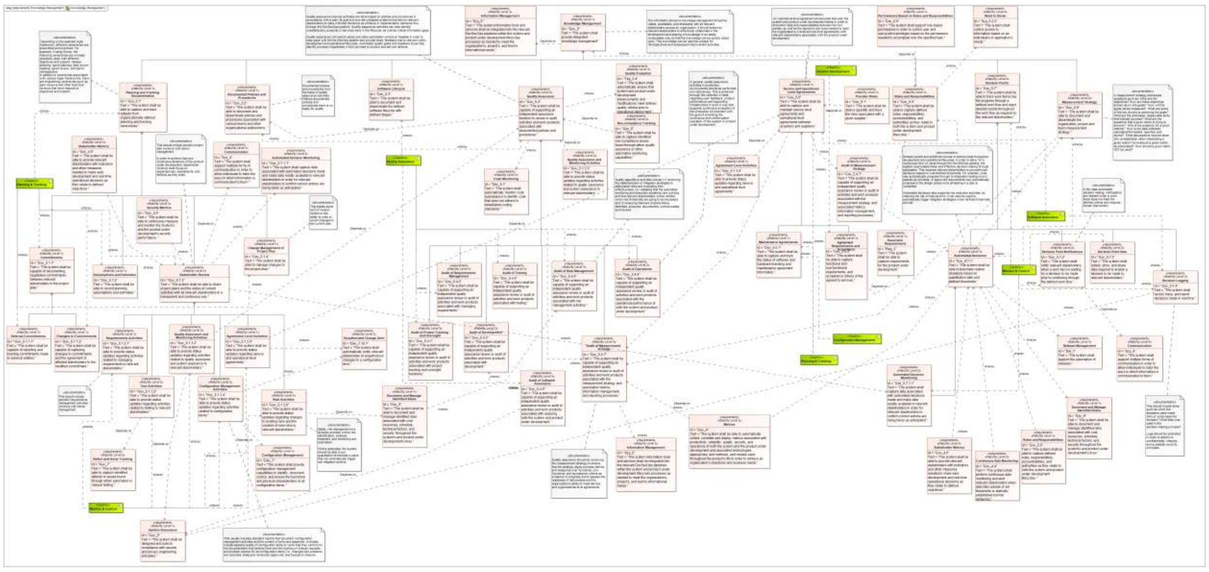


Figure 14: Knowledge Management

Documentation:

The information stored in knowledge management should be visible, accessible, and shareable with all relevant stakeholders across an organization. It should empower relevant stakeholders to effectively collaborate in the development and sharing of knowledge in an easily consumable way so that the knowledge can be quickly acted upon. This knowledge can be used as a basis for retrospectives and subsequent improvement activities.

3.1.5.1 Gov_5.1 Planning and Tracking Documentation

The system shall be able to capture and track outputs from organizationally-defined planning and tracking ceremonies.

Documentation:

Depending on the selected Agile framework, different ceremonies are prescribed and practiced. For example, if using Scrum, the following ceremonies are normally practiced, each with different objectives and outputs: release planning, sprint planning, daily scrum meeting, sprint review, and sprint retrospective.

In addition to ceremonies associated with various Agile frameworks, there are engineering ceremonies such as peer reviews and other technical reviews that have respective objectives and outputs.

3.1.5.1.1 Gov_5.1.1 Commitments

The system shall be capable of documenting negotiated commitments between relevant stakeholders in the project plan.

3.1.5.1.1.1 Gov_5.1.1.1 External Commitments

The system shall be capable of reporting and tracking commitments made to external entities.

3.1.5.1.1.2 Gov_5.1.1.2 Changes to Commitments

The system shall be capable of capturing changes to commitments and the agreement of affected stakeholders to the modified commitment.

3.1.5.1.2 Gov_5.1.2 Assumptions and Estimates

The system shall be able to record planning assumptions and estimates.

3.1.5.1.3 Gov_5.1.3 Stakeholder Review

The system shall be able to share project plans and the status of current activities with all relevant stakeholders in a transparent and continuous way.

Documentation:

This should include periodic project plan reviews with senior management.

In order to achieve fast and continuous iterations of the product under development, stakeholder reviews must focus on dependencies, impediments, and defects as they arise.

3.1.5.1.3.1 Gov_5.1.3.1 Requirements Activities

The system shall be able to provide status updates regarding activities related to managing requirements to relevant stakeholders.

Documentation:

This should include periodic requirements management activities reviews with senior management.

3.1.5.1.3.2 Gov_5.1.3.2 Test Activities

The system shall be able to provide status updates regarding activities related to testing to relevant stakeholders.

3.1.5.1.3.3 Gov_5.1.3.3 Quality Assurance and Monitoring Activities

The system shall be able to provide status updates regarding activities related to quality assurance and system assurance to relevant stakeholders.

3.1.5.1.3.4 Gov_5.1.3.4 Configuration Management Activities

The system shall be able to provide status updates regarding activities related to configuration items.

Documentation:

This usually includes standard reports that document configuration management activities and the content of software baselines. Activities include baseline audits of configuration items to verify that they conform to the documentation that defines them and the tracking of change requests and problem reports for all configuration items (i.e., changes and problems are recorded, analyzed, reviewed, approved, and tracked to closure).

3.1.5.1.3.5 Gov_5.1.3.5 Agreement Level Activities

The system shall be able to provide status updates regarding service and operational level agreements.

3.1.5.1.3.6 Gov_5.1.3.6 Risk Activities

The system shall be able to provide status updates regarding changes to existing risks and the creation of new risks to relevant stakeholders.

Documentation:

Ideally, risk management is a dynamic process where risk identification, analysis, treatment, and monitoring are automated.

Where applicable, the system should be able to put quantitative thresholds in place that can automatically trigger risk mitigation actions.

3.1.5.1.4 Gov_5.1.4 Change Management of Project Plan

The system shall be able to manage changes to the project plan.

Documentation:

This implies some sort of version control or the ability to undo or revert changes to the current plan.

3.1.5.2 Gov_5.2 Documented Policies and Procedures

The system shall be able to document and disseminate policies and procedures associated with various teams and other organizational abstractions.

Documentation:

Documented policies and procedures form the basis of quality assurance activities. Without documented policies and procedures there is no basis for audits.

3.1.5.3 Gov_5.3 Software Lifecycle

The system shall be able to document and disseminate the defined software lifecycle with defined stages.

3.1.5.4 Gov_5.4 Quality Assurance

The system shall be capable of supporting an independent assurance function to review or audit activities and work products associated with documented policies and procedures.

Documentation:

Quality assurance ensures activities are done based on policies and procedures in accordance with a plan. Its goal is to provide unbiased evidence that allows relevant stakeholders to make informed decisions as artifacts or implementation elements flow through the DevSecOps pipeline. Quality assurance activities can help identify unsatisfactory products or services early in the lifecycle, as well as critical information gaps.

Quality assurance will need to adopt and utilize automation wherever feasible in order to keep pace with the DevSecOps pipeline and provide timely feedback that is relevant within development and operational lifecycles. Automated quality gates and feedback loops help identify process irregularities which can lead to product and service defects.

3.1.5.4.1 Gov_5.4.1 Audit of Project Tracking and Oversight

The system shall be capable of supporting an independent quality assurance review or audit of activities and work products associated with project tracking and oversight functions.

3.1.5.4.2 Gov_5.4.2 Audit of Requirements Management

The system shall be capable of supporting an independent quality assurance review or audit of activities and work products associated with managing requirements.

3.1.5.4.3 Gov_5.4.3 Audit of Development

The system shall be capable of supporting an independent quality assurance review or audit of activities and work products associated with development.

3.1.5.4.4 Gov_5.4.4 Audit of Testing

The system shall be capable of supporting an independent quality assurance review or audit of activities and work products associated with testing.

3.1.5.4.5 Gov_5.4.5 Audit of Software Assurance

The system shall be capable of supporting an independent quality assurance review or audit of activities and work products associated with assuring both the system and product under development.

3.1.5.4.6 Gov_5.4.6 Audit of Risk Management

The system shall be capable of supporting an independent quality assurance review or audit of activities and work products associated with risk management activities.

Documentation:

Quality assurance activities include (1) reviewing the implementation of mitigation strategies to associated risks and evaluating their effectiveness, (2) validating that the automated monitoring and execution system(s) are working and that relevant stakeholders will be notified when risk thresholds are going to be exceeded, and (3) ensuring risks are properly being identified, analyzed, documented, communicated, and tracked.

3.1.5.4.7 Gov_5.4.7 Audit of Measurement Strategy

The system shall be capable of supporting an independent quality assurance review or audit of activities and work products associated with the measurement strategy and associated metrics, information management, and reporting processes.

Documentation:

Quality assurance should be reviewing the measurement strategy to ensure that the strategy aligns process metrics and measures over functional, non-functional, and acceptance criteria as evidence of progress and to assess the readiness of deliverables and the organization's ability to meet service and organizational level agreements.

3.1.5.4.8 Gov_5.4.8 Audit of Operations

The system shall be capable of supporting an independent quality assurance review or audit of activities and work products associated with the operational performance of both the system and product under development.

Documentation:

In general, quality assurance activities in production environments should be performed non-obtrusively. This is achieved through the collection of data regarding user feedback, product performance and supporting infrastructure in such a way that supports continuous evaluation of the production environment with the goal of promoting the continuous and uninterrupted operation of the system or product under development.

3.1.5.5 Gov_5.5 Service and Operational Level Agreements

The system shall be able to capture and maintain service level agreements and operational level agreements between acquirers and suppliers.

Documentation:

An operational level agreement should exist between the system and product under development teams in order to document roles and responsibilities between the two parties, as well as the agreed to services needed to meet the organization's overall service level agreements with relevant stakeholders associated with the product under development.

3.1.5.5.1 Gov_5.5.1 Maintenance Agreements

The system shall be able to capture and track the status of software and hardware licensing and maintenance agreement information.

3.1.5.5.2 Gov_5.5.2 Agreement Requirements and Acceptance

The system shall be able to capture functional and non-functional requirements, and acceptance criteria of the agreed to services.

3.1.5.6 Gov_5.6 Roles and Responsibilities

The system shall be able to capture defined roles, responsibilities, accountabilities, and authorities as they relate to both the system and product under development lifecycles.

3.1.5.7 Gov_5.7 Decision Points

The system shall be able to track work items has the progress through a defined work flow and inject decision points throughout the work flow as required by the relevant stakeholders.

3.1.5.7.1 Gov_5.7.1 Automated Decisions

The system shall be able to automate routine decisions based on quantitative date and defined thresholds.

Documentation:

Decision points and preferred course of actions exist throughout development and operational lifecycles. In order to allow for a continuous flow of value through the DevSecOps pipeline, the system must enable timely and effective decision-making through automation. This requires relevant stakeholders to pre-authorize decisions based on well-defined thresholds. For example, code may automatically progress through to integration testing once it passes all unit tests, or approved requirements may automatically progress to the design phase once all tasking in a plan is completed.

Automated decisions also support risk reduction activities, by reducing the risk of manual error. It can also be used to automatically trigger mitigation strategies when defined thresholds are met.

3.1.5.7.1.1 Gov_5.7.1.1 Automated Decision Monitoring

The system shall capture data associated with automated decisions made and make data readily available to relevant stakeholders in order for relevant stakeholders to confirm correct actions are being taken as anticipated.

3.1.5.7.2 Gov_5.7.2 Decision Point Notifications

The system shall notify relevant stakeholders when a work item is waiting for a decision to be made prior to continuing through the defined work flow.

Documentation:

In the case automated decision making, notifications are needed when a work items does not meet the defined criteria and requires human intervention.

3.1.5.7.3 Gov_5.7.3 Decision Point Data

The system shall collect, store, and share data required to enable a decision to be made by relevant stakeholders.

3.1.5.7.4 Gov_5.7.4 Decision Logging

The system shall record, trace, and report decisions made in real time.

Documentation:

This should include items such as when the decisions were made. Who or what made the decision? What data was used in the decision-making process?

Logs should be protected in order to adhere to confidentiality, integrity, and availability security principles.

3.1.5.8 Gov_5.8 Measurement Strategy

The system shall be able to document and disseminate the organization, project and team measurement strategy.

Documentation:

A measurement strategy addresses questions such as: What are the objectives? How are these objectives broken down into goals? How will the goals will be measured? What are some of the key drivers to achieving the goals? What are the actionable, measurable items that indicate success? What are the questions that a given metric is trying to answer? Who is the audience for a given metrics? How is the data collected, calculated/formulated, reported, and stored? What assumptions must be taken into consideration when interpreting a given metric? How should a given metric be interpreted? How should a given metric NOT be used?

3.1.6 Gov_6 System Assurance

The system shall be designed and build in compliance with security and privacy engineering principles.

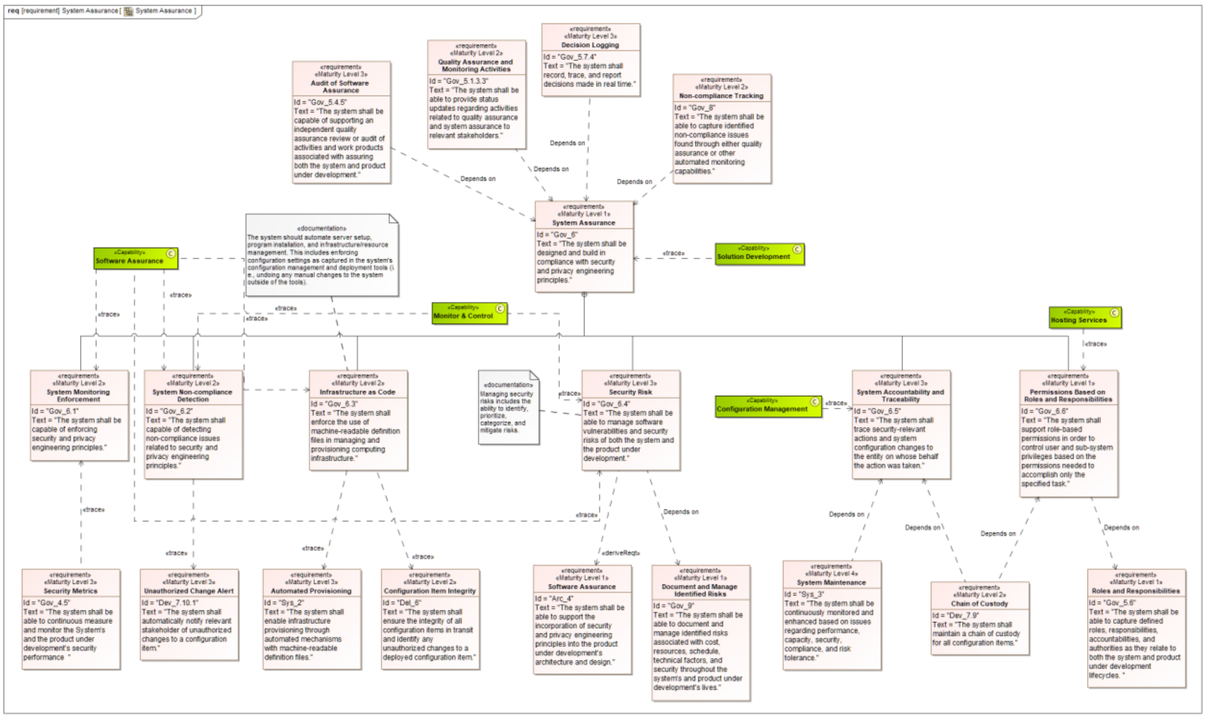


Figure 15: System Assurance

3.1.6.1 Gov_6.1 System Monitoring Enforcement

The system shall be capable of enforcing security and privacy engineering principles.

3.1.6.2 Gov_6.2 System Non-compliance Detection

The system shall capable of detecting non-compliance issues related to security and privacy engineering principles.

3.1.6.3 Gov_6.3 Infrastructure as Code

The system shall enforce the use of machine-readable definition files in managing and provisioning computing infrastructure.

Documentation:

The system should automate server setup, program installation, and infrastructure/resource management. This includes enforcing configuration settings as captured in the system's configuration management and deployment tools (i.e., undoing any manual changes to the system outside of the tools).

3.1.6.4 Gov_6.4 Security Risk

The system shall be able to manage software vulnerabilities and security risks of both the system and the product under development.

Documentation:

Managing security risks includes the ability to identify, prioritize, categorize, and mitigate risks.

3.1.6.5 Gov_6.5 System Accountability and Traceability

The system shall trace security-relevant actions and system configuration changes to the entity on whose behalf the action was taken.

3.1.6.6 Gov_6.6 Permissions Based on Roles and Responsibilities

The system shall support role-based permissions in order to control user and sub-system privileges based on the permissions needed to accomplish only the specified task.

3.1.7 Gov_7 Defect and Issue Tracking

The system shall be able to capture identified defects or issues found through either automated or manual testing.

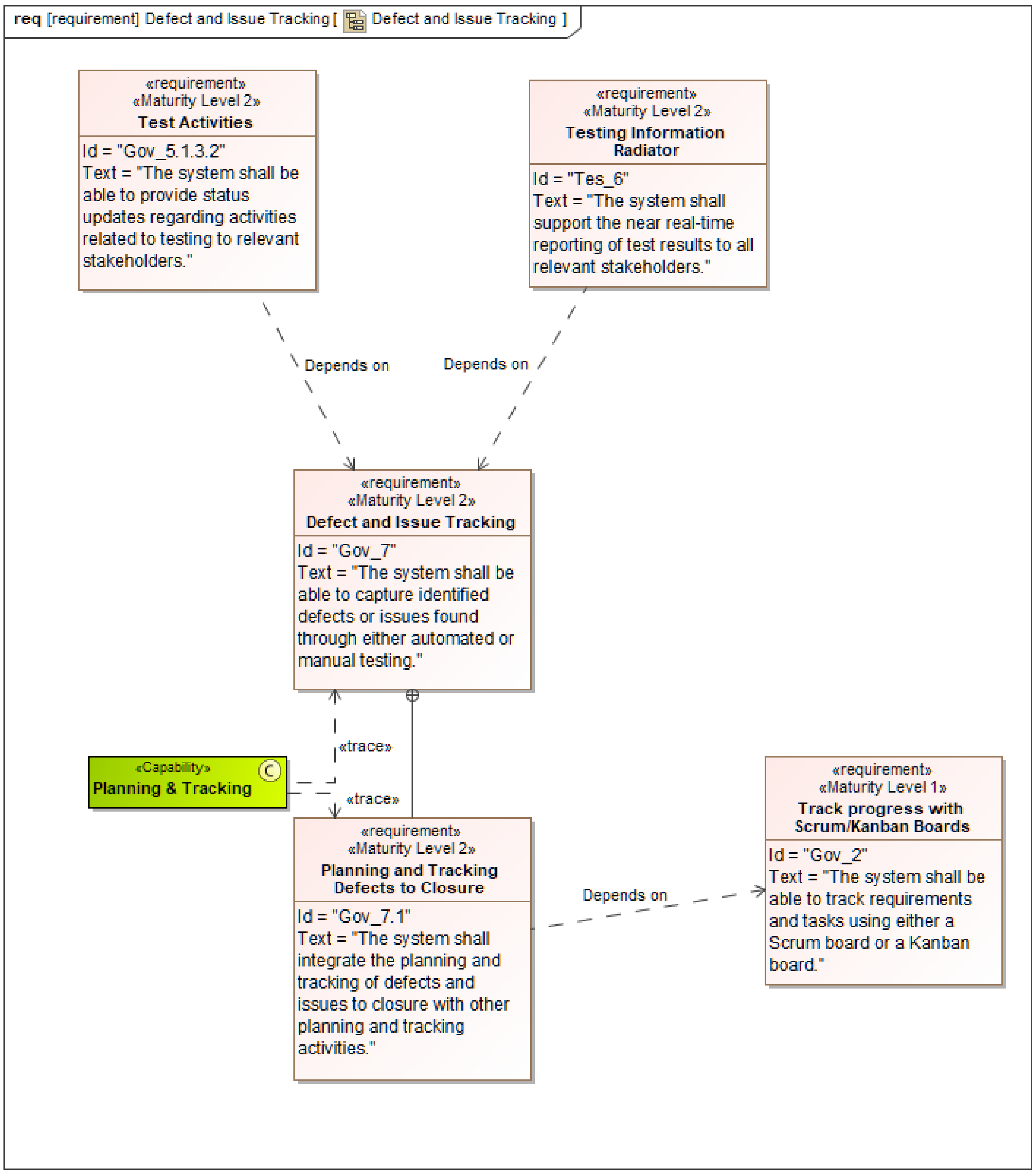

req [requirement] Defect and Issue Tracking [Defect and Issue Tracking]
«requirement» «Maturity Level 2» Test Activities
Id = "Gov_5.1.3.2"
Text = "The system shall be able to provide status updates regarding activities related to testing to relevant stakeholders."
«requirement» «Maturity Level 2» Testing Information Radiator
Id = "Tes_6"
Text = "The system shall support the near real-time reporting of test results to all relevant stakeholders."
Depends on
Depends on
«requirement» «Maturity Level 2» Defect and Issue Tracking
Id = "Gov_7"
Text = "The system shall be able to capture identified defects or issues found through either automated or manual testing."
«trace»
«trace»
«Capability» Planning & Tracking
«requirement» «Maturity Level 2» Planning and Tracking Defects to Closure
Id = "Gov_7.1"
Text = "The system shall integrate the planning and tracking of defects and issues to closure with other planning and tracking activities."
Depends on
«requirement» «Maturity Level 1» Track progress with Scrum/Kanban Boards
Id = "Gov_2"
Text = "The system shall be able to track requirements and tasks using either a Scrum board or a Kanban board."

Figure 16: Defect and Issue Tracking

3.1.7.1 Gov_7.1 Planning and Tracking Defects to Closure

The system shall integrate the planning and tracking of defects and issues to closure with other planning and tracking activities.

3.1.8 Gov_8 Non-compliance Tracking

The system shall be able to capture identified non-compliance issues found through either quality assurance or other automated monitoring capabilities.

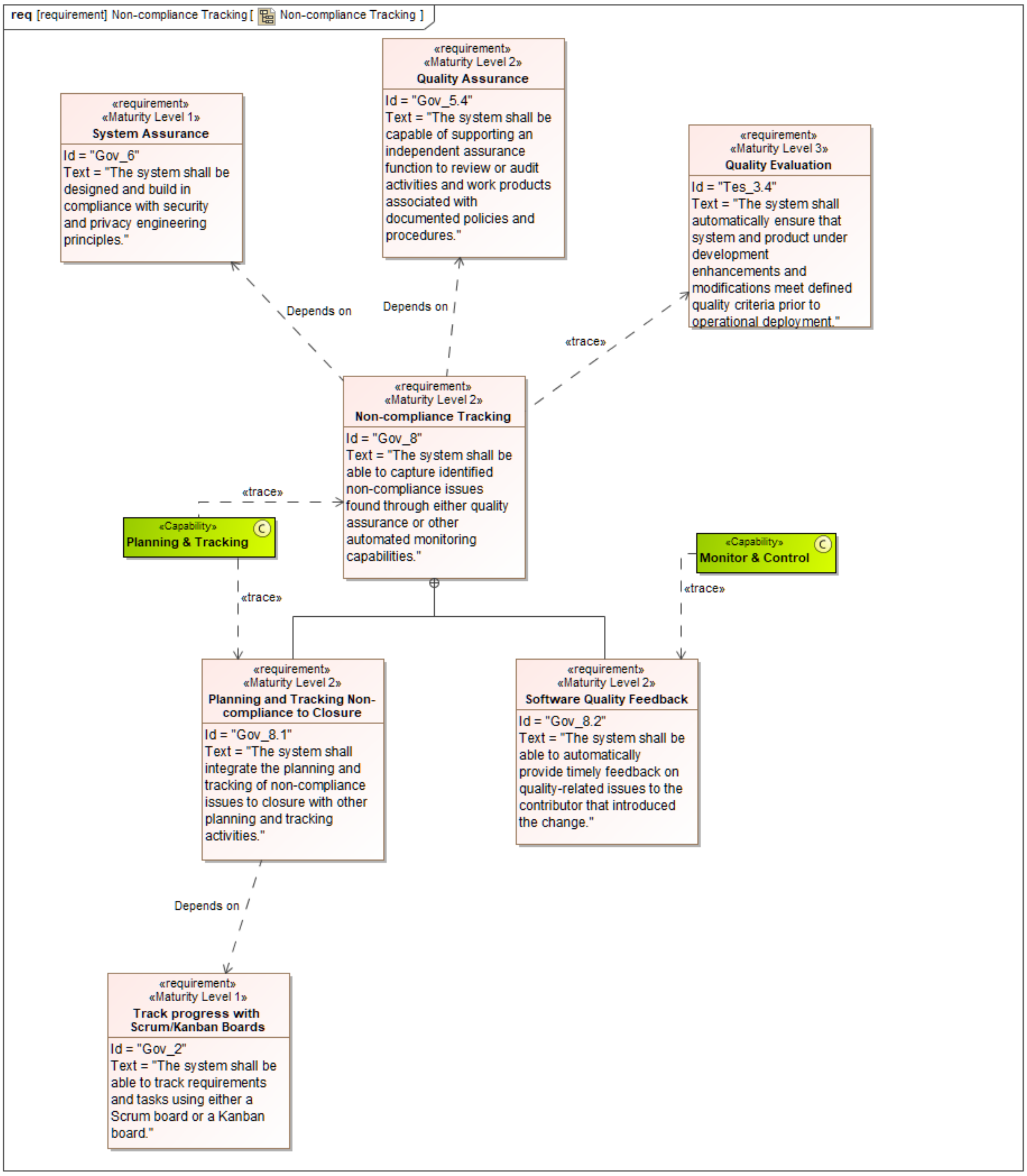

req [requirement] Non-compliance Tracking [Non-compliance Tracking]
«requirement» «Maturity Level 1» System Assurance
Id = "Gov_6"
Text = "The system shall be designed and build in compliance with security and privacy engineering principles."
«requirement» «Maturity Level 2» Quality Assurance
Id = "Gov_5.4"
Text = "The system shall be capable of supporting an independent assurance function to review or audit activities and work products associated with documented policies and procedures."
«requirement» «Maturity Level 3» Quality Evaluation
Id = "Tes_3.4"
Text = "The system shall automatically ensure that system and product under development enhancements and modifications meet defined quality criteria prior to operational deployment."
Depends on
Depends on
«trace»
«requirement» «Maturity Level 2» Non-compliance Tracking
Id = "Gov_8"
Text = "The system shall be able to capture identified non-compliance issues found through either quality assurance or other automated monitoring capabilities."
«trace»
«Capability» Planning & Tracking
«Capability» Monitor & Control
«trace»
«trace»
«requirement» «Maturity Level 2» Planning and Tracking Non-compliance to Closure
Id = "Gov_8.1"
Text = "The system shall integrate the planning and tracking of non-compliance issues to closure with other planning and tracking activities."
«requirement» «Maturity Level 2» Software Quality Feedback
Id = "Gov_8.2"
Text = "The system shall be able to automatically provide timely feedback on quality-related issues to the contributor that introduced the change."
Depends on
«requirement» «Maturity Level 1» Track progress with Scrum/Kanban Boards
Id = "Gov_2"
Text = "The system shall be able to track requirements and tasks using either a Scrum board or a Kanban board."

Figure 17: Non-compliance Tracking

3.1.8.1 Gov_8.1 Planning and Tracking Non-compliance to Closure

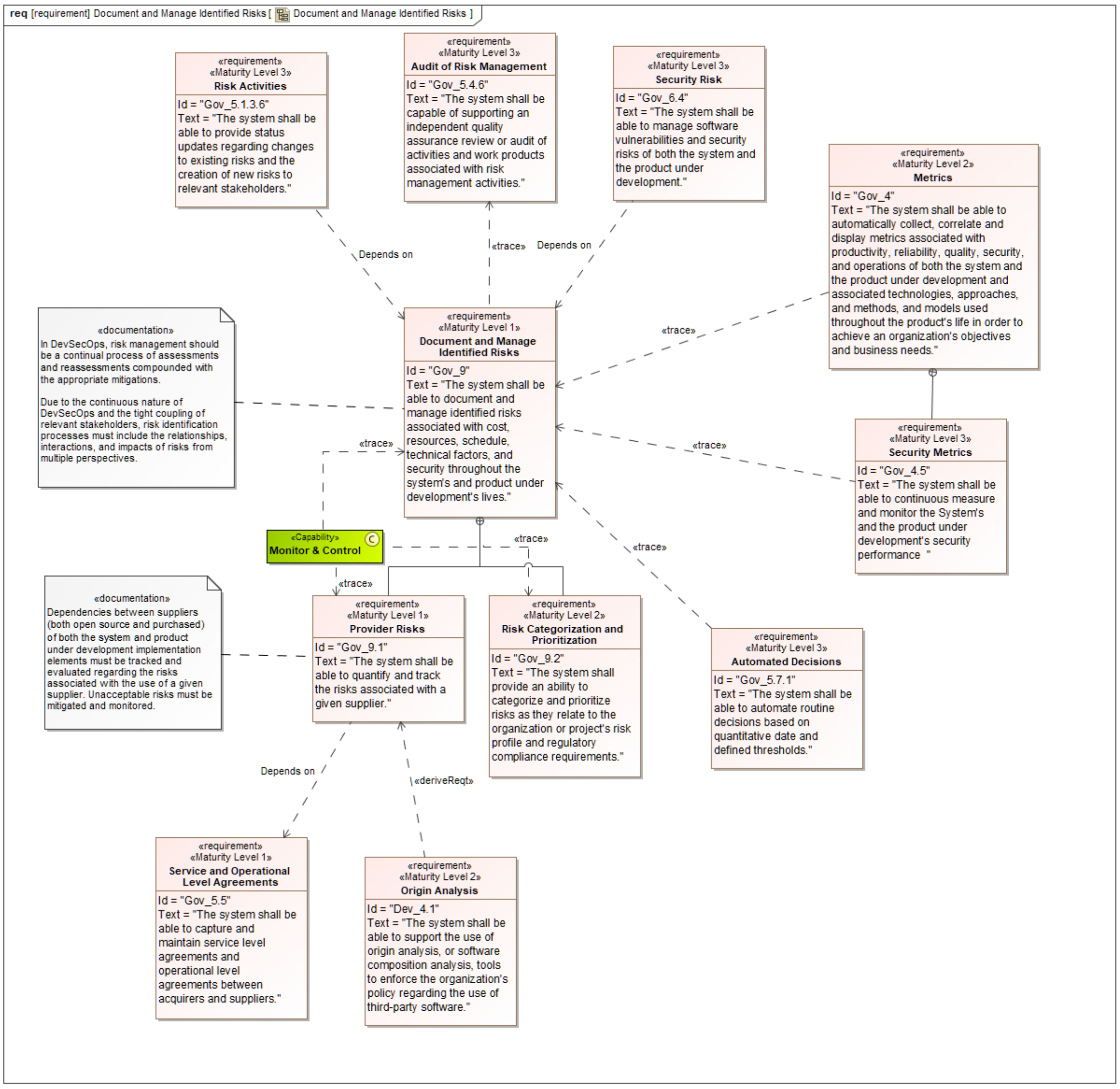
The system shall integrate the planning and tracking of non-compliance issues to closure with other planning and tracking activities.

3.1.8.2 Gov_8.2 Software Quality Feedback

The system shall be able to automatically provide timely feedback on quality-related issues to the contributor that introduced the change.

3.1.9 Gov_9 Document and Manage Identified Risks

The system shall be able to document and manage identified risks associated with cost, resources, schedule, technical factors, and security throughout the system's and product under development's lives.



«requirement» «Maturity Level 3» **Risk Activities**
Id = "Gov_5.1.3.6"
Text = "The system shall be able to provide status updates regarding changes to existing risks and the creation of new risks to relevant stakeholders."

«requirement» «Maturity Level 3» **Audit of Risk Management**
Id = "Gov_5.4.6"
Text = "The system shall be capable of supporting an independent quality assurance review or audit of activities and work products associated with risk management activities."

«requirement» «Maturity Level 3» **Security Risk**
Id = "Gov_6.4"
Text = "The system shall be able to manage software vulnerabilities and security risks of both the system and the product under development."

«requirement» «Maturity Level 2» **Metrics**
Id = "Gov_4"
Text = "The system shall be able to automatically collect, correlate and display metrics associated with productivity, reliability, quality, security, and operations of both the system and the product under development and associated technologies, approaches, and methods, and models used throughout the product's life in order to achieve an organization's objectives and business needs."

«documentation»
In DevSecOps, risk management should be a continual process of assessments and reassessments compounded with the appropriate mitigations.

Due to the continuous nature of DevSecOps and the tight coupling of relevant stakeholders, risk identification processes must include the relationships, interactions, and impacts of risks from multiple perspectives.

«requirement» «Maturity Level 1» **Document and Manage Identified Risks**
Id = "Gov_9"
Text = "The system shall be able to document and manage identified risks associated with cost, resources, schedule, technical factors, and security throughout the system's and product under development's lives."

«requirement» «Maturity Level 3» **Security Metrics**
Id = "Gov_4.5"
Text = "The system shall be able to continuous measure and monitor the System's and the product under development's security performance "

«documentation»
Dependencies between suppliers (both open source and purchased) of both the system and product under development implementation elements must be tracked and evaluated regarding the risks associated with the use of a given supplier. Unacceptable risks must be mitigated and monitored.

«requirement» «Maturity Level 1» **Provider Risks**
Id = "Gov_9.1"
Text = "The system shall be able to quantify and track the risks associated with a given supplier."

«requirement» «Maturity Level 2» **Risk Categorization and Prioritization**
Id = "Gov_9.2"
Text = "The system shall provide an ability to categorize and prioritize risks as they relate to the organization or project's risk profile and regulatory compliance requirements."

«requirement» «Maturity Level 3» **Automated Decisions**
Id = "Gov_5.7.1"
Text = "The system shall be able to automate routine decisions based on quantitative date and defined thresholds."

«requirement» «Maturity Level 1» **Service and Operational Level Agreements**
Id = "Gov_5.5"
Text = "The system shall be able to capture and maintain service level agreements and operational level agreements between acquirers and suppliers."

«requirement» «Maturity Level 2» **Origin Analysis**
Id = "Dev_4.1"
Text = "The system shall be able to support the use of origin analysis, or software composition analysis, tools to enforce the organization's policy regarding the use of third-party software."

Figure 18: Document and Manage Identified Risks

Documentation:

In DevSecOps, risk management should be a continual process of assessments and reassessments compounded with the appropriate mitigations.

Due to the continuous nature of DevSecOps and the tight coupling of relevant stakeholders, risk identification processes must include the relationships, interactions, and impacts of risks from multiple perspectives.

3.1.9.1 Gov_9.1 Provider Risks

The system shall be able to quantify and track the risks associated with a given supplier.

Documentation:

Dependencies between suppliers (both open source and purchased) of both the system and product under development implementation elements must be tracked and evaluated regarding the risks associated with the use of a given supplier. Unacceptable risks must be mitigated and monitored.

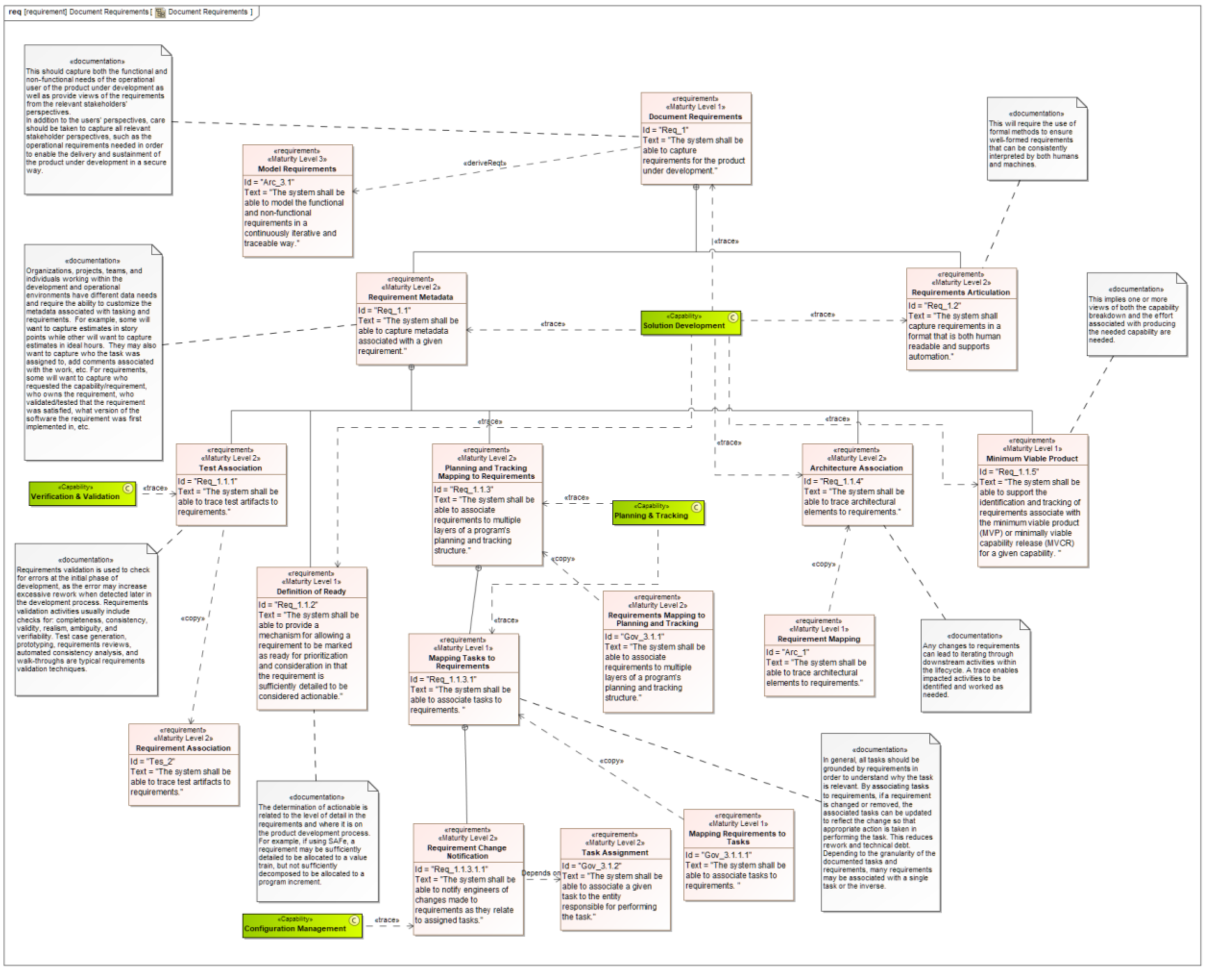
3.1.9.2 Gov_9.2 Risk Categorization and Prioritization

The system shall provide an ability to categorize and prioritize risks as they relate to the organization or project's risk profile and regulatory compliance requirements.

3.2 2 Requirements

3.2.1 Req_1 Document Requirements

The system shall be able to capture requirements for the product under development.


req [requirement] Document Requirements [Document Requirements]
«deriveReqt»
«trace»
«copy»
Depends on
«Capability» Solution Development
«Capability» Verification & Validation
«Capability» Planning & Tracking
«Capability» Configuration Management

«documentation»
This should capture both the functional and non-functional needs of the operational user of the product under development as well as provide views of the requirements from the relevant stakeholders' perspectives.
In addition to the users' perspectives, care should be taken to capture all relevant stakeholder perspectives, such as the operational requirements needed in order to enable the delivery and sustainment of the product under development in a secure way.

«requirement» «Maturity Level 1» **Document Requirements**
Id = "Req_1"
Text = "The system shall be able to capture requirements for the product under development."

«documentation»
This will require the use of formal methods to ensure well-formed requirements that can be consistently interpreted by both humans and machines.

«requirement» «Maturity Level 3» **Model Requirements**
Id = "Arc_3.1"
Text = "The system shall be able to model the functional and non-functional requirements in a continuously iterative and traceable way."

«documentation»
Organizations, projects, teams, and individuals working within the development and operational environments have different data needs and require the ability to customize the metadata associated with tasking and requirements. For example, some will want to capture estimates in story points while other will want to capture estimates in ideal hours. They may also want to capture who the task was assigned to, add comments associated with the work, etc. For requirements, some will want to capture who requested the capability/requirement, who owns the requirement, who validated/tested that the requirement was satisfied, what version of the software the requirement was first implemented in, etc.

«requirement» «Maturity Level 2» **Requirement Metadata**
Id = "Req_1.1"
Text = "The system shall be able to capture metadata associated with a given requirement."

«requirement» «Maturity Level 2» **Requirements Articulation**
Id = "Req_1.2"
Text = "The system shall capture requirements in a format that is both human readable and supports automation."

«documentation»
This implies one or more views of both the capability breakdown and the effort associated with producing the needed capability are needed.

«requirement» «Maturity Level 2» **Test Association**
Id = "Req_1.1.1"
Text = "The system shall be able to trace test artifacts to requirements."

«requirement» «Maturity Level 2» **Planning and Tracking Mapping to Requirements**
Id = "Req_1.1.3"
Text = "The system shall be able to associate requirements to multiple layers of a program's planning and tracking structure."

«requirement» «Maturity Level 2» **Architecture Association**
Id = "Req_1.1.4"
Text = "The system shall be able to trace architectural elements to requirements."

«requirement» «Maturity Level 1» **Minimum Viable Product**
Id = "Req_1.1.5"
Text = "The system shall be able to support the identification and tracking of requirements associate with the minimum viable product (MVP) or minimally viable capability release (MVCR) for a given capability. "

«documentation»
Requirements validation is used to check for errors at the initial phase of development, as the error may increase excessive rework when detected later in the development process. Requirements validation activities usually include checks for: completeness, consistency, validity, realism, ambiguity, and verifiability. Test case generation, prototyping, requirements reviews, automated consistency analysis, and walk-throughs are typical requirements validation techniques.

«requirement» «Maturity Level 1» **Definition of Ready**
Id = "Req_1.1.2"
Text = "The system shall be able to provide a mechanism for allowing a requirement to be marked as ready for prioritization and consideration in that the requirement is sufficiently detailed to be considered actionable."

«requirement» «Maturity Level 2» **Requirements Mapping to Planning and Tracking**
Id = "Gov_3.1.1"
Text = "The system shall be able to associate requirements to multiple layers of a program's planning and tracking structure."

«requirement» «Maturity Level 1» **Requirement Mapping**
Id = "Arc_1"
Text = "The system shall be able to trace architectural elements to requirements."

«documentation»
Any changes to requirements can lead to iterating through downstream activities within the lifecycle. A trace enables impacted activities to be identified and worked as needed.

«requirement» «Maturity Level 1» **Mapping Tasks to Requirements**
Id = "Req_1.1.3.1"
Text = "The system shall be able to associate tasks to requirements. "

«requirement» «Maturity Level 2» **Requirement Association**
Id = "Tes_2"
Text = "The system shall be able to trace test artifacts to requirements."

«documentation»
The determination of actionable is related to the level of detail in the requirements and where it is on the product development process. For example, if using SAFe, a requirement may be sufficiently detailed to be allocated to a value train, but not sufficiently decomposed to be allocated to a program increment.

«requirement» «Maturity Level 2» **Requirement Change Notification**
Id = "Req_1.1.3.1.1"
Text = "The system shall be able to notify engineers of changes made to requirements as they relate to assigned tasks."

«requirement» «Maturity Level 2» **Task Assignment**
Id = "Gov_3.1.2"
Text = "The system shall be able to associate a given task to the entity responsible for performing the task."

«requirement» «Maturity Level 1» **Mapping Requirements to Tasks**
Id = "Gov_3.1.1.1"
Text = "The system shall be able to associate tasks to requirements. "

«documentation»
In general, all tasks should be grounded by requirements in order to understand why the task is relevant. By associating tasks to requirements, if a requirement is changed or removed, the associated tasks can be updated to reflect the change so that appropriate action is taken in performing the task. This reduces rework and technical debt. Depending to the granularity of the documented tasks and requirements, many requirements may be associated with a single task or the inverse.

Figure 19: Document Requirements

Documentation:

This should capture both the functional and non-functional needs of the operational user of the product under development as well as provide views of the requirements from the relevant stakeholders' perspectives.

In addition to the users' perspectives, care should be taken to capture all relevant stakeholder perspectives, such as the operational requirements needed in order to enable the delivery and sustainment of the product under development in a secure way.

3.2.1.1 Req_1.1 Requirement Metadata

The system shall be able to capture metadata associated with a given requirement.

Documentation:

Organizations, projects, teams, and individuals working within the development and operational environments have different data needs and require the ability to customize the metadata associated with tasking and requirements. For example, some will want to capture estimates in story points while other will want to capture estimates in ideal hours. They may also want to capture who the task was assigned to, add comments associated with the work, etc. For requirements, some will want to capture who requested the capability/requirement, who owns the requirement, who validated/tested that the requirement was satisfied, what version of the software the requirement was first implemented in, etc.

3.2.1.1.1 Req_1.1.1 Test Association

The system shall be able to trace test artifacts to requirements.

Documentation:

Requirements validation is used to check for errors at the initial phase of development, as the error may increase excessive rework when detected later in the development process. Requirements validation activities usually include checks for: completeness, consistency, validity, realism, ambiguity, and verifiability. Test case generation, prototyping, requirements reviews, automated consistency analysis, and walk-throughs are typical requirements validation techniques.

3.2.1.1.2 Req_1.1.2 Definition of Ready

The system shall be able to provide a mechanism for allowing a requirement to be marked as ready for prioritization and consideration in that the requirement is sufficiently detailed to be considered actionable.

Documentation:

The determination of actionable is related to the level of detail in the requirements and where it is on the product development process. For example, if using SAFe, a requirement may be sufficiently detailed to be allocated to a value train, but not sufficiently decomposed to be allocated to a program increment.

3.2.1.1.3 Req_1.1.3 Planning and Tracking Mapping to Requirements

The system shall be able to associate requirements to multiple layers of a program's planning and tracking structure.

3.2.1.1.3.1 Req_1.1.3.1 Mapping Tasks to Requirements

The system shall be able to associate tasks to requirements.

Documentation:

In general, all tasks should be grounded by requirements in order to understand why the task is relevant. By associating tasks to requirements, if a requirement is changed or removed, the associated tasks can be updated to reflect the change so that appropriate action is taken in performing the task. This reduces rework and technical debt. Depending to the granularity of the documented tasks and requirements, many requirements may be associated with a single task or the inverse.

3.2.1.1.3.2 Req_1.1.3.1.1 Requirement Change Notification

The system shall be able to notify engineers of changes made to requirements as they relate to assigned tasks.

3.2.1.1.4 Req_1.1.4 Architecture Association

The system shall be able to trace architectural elements to requirements.

Documentation:

Any changes to requirements can lead to iterating through downstream activities within the lifecycle. A trace enables impacted activities to be identified and worked as needed.

3.2.1.1.5 Req_1.1.5 Minimum Viable Product

The system shall be able to support the identification and tracking of requirements associate with the minimum viable product (MVP) or minimally viable capability release (MVCR) for a given capability.

Documentation:

This implies one or more views of both the capability breakdown and the effort associated with producing the needed capability are needed.

3.2.1.2 Req_1.2 Requirements Articulation

The system shall capture requirements in a format that is both human readable and supports automation.

Documentation:

This will require the use of formal methods to ensure well-formed requirements that can be consistently interpreted by both humans and machines.

3.2.2 Req_2 Requirements Abstraction Layers

The system shall be able to capture different levels of requirement abstractions and associated requirements between the abstraction layers.

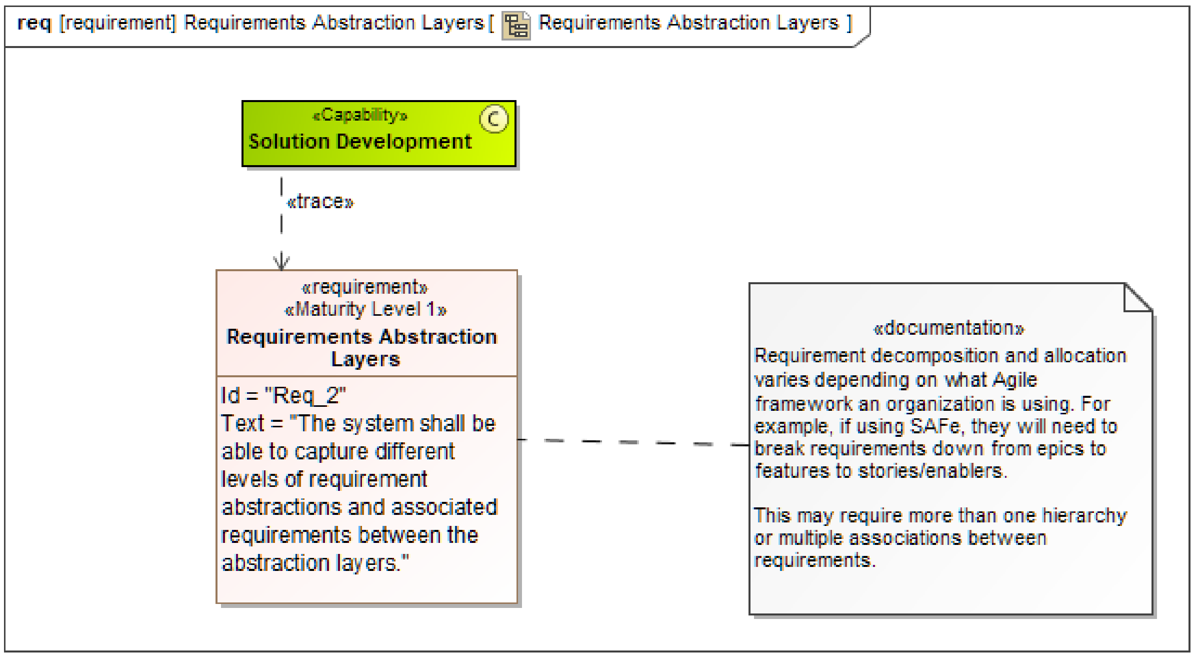

req [requirement] Requirements Abstraction Layers [Requirements Abstraction Layers]
«Capability»
Solution Development
«trace»
«requirement»
«Maturity Level 1»
Requirements Abstraction Layers
Id = "Req_2"
Text = "The system shall be able to capture different levels of requirement abstractions and associated requirements between the abstraction layers."
«documentation»
Requirement decomposition and allocation varies depending on what Agile framework an organization is using. For example, if using SAFe, they will need to break requirements down from epics to features to stories/enablers.
This may require more than one hierarchy or multiple associations between requirements.

Figure 20: Requirements Abstraction Layers

Documentation:

Requirement decomposition and allocation varies depending on what Agile framework an organization is using. For example, if using SAFe, they will need to break requirements down from epics to features to stories/enablers.

This may require more than one hierarchy or multiple associations between requirements.

3.2.3 Req_3 Requirements Prioritization

The system shall be able to prioritize requirements and capture the rationale associated with the prioritization process used.

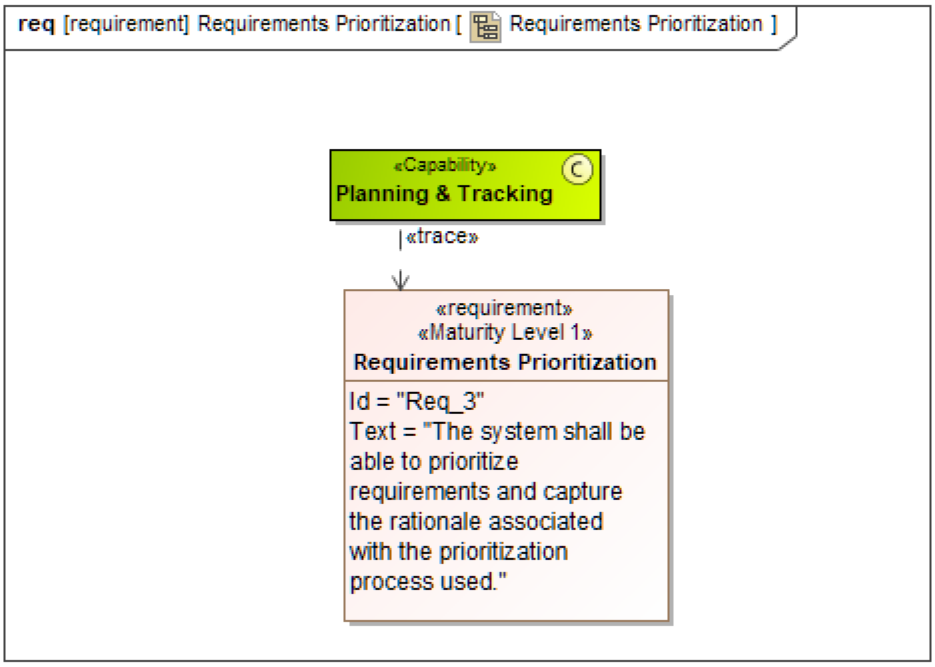

req [requirement] Requirements Prioritization [Requirements Prioritization]
«Capability»
Planning & Tracking
«trace»
«requirement»
«Maturity Level 1»
Requirements Prioritization
Id = "Req_3"
Text = "The system shall be able to prioritize requirements and capture the rationale associated with the prioritization process used."

Figure 21: Requirements Prioritization

3.2.4 Req_4 Requirements Validation

The system shall be able to support requirements feasibility and validation analysis, in which identified problems and agreed on actions of detected problems are collected and tracked to completion.

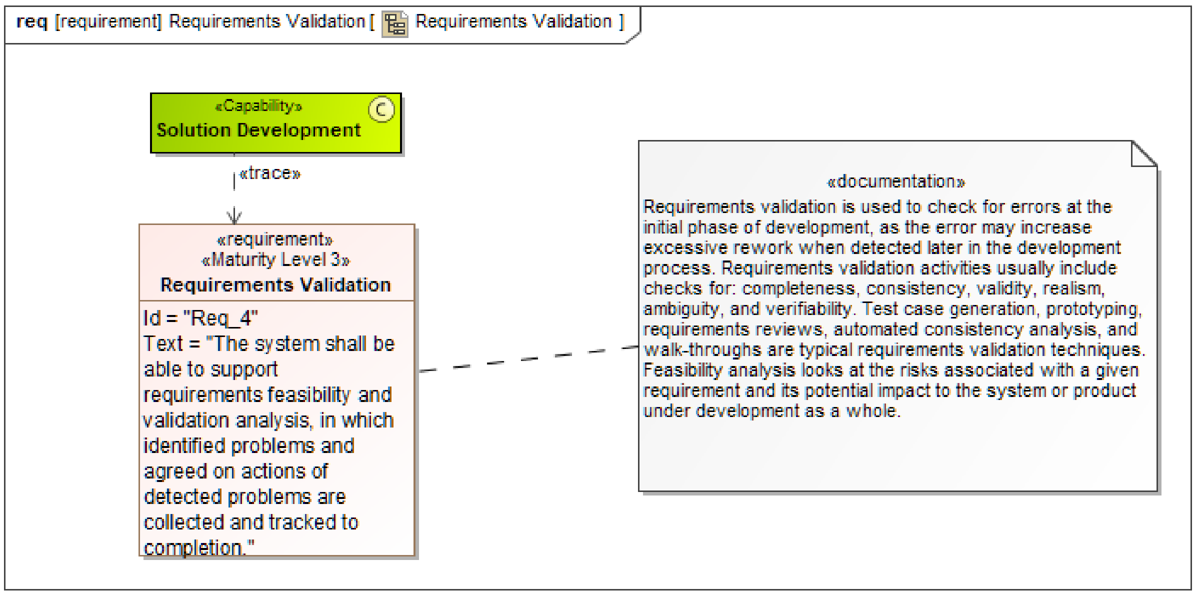

req [requirement] Requirements Validation [Requirements Validation]
«Capability»
Solution Development
«trace»
«requirement»
«Maturity Level 3»
Requirements Validation
Id = "Req_4"
Text = "The system shall be able to support requirements feasibility and validation analysis, in which identified problems and agreed on actions of detected problems are collected and tracked to completion."
«documentation»
Requirements validation is used to check for errors at the initial phase of development, as the error may increase excessive rework when detected later in the development process. Requirements validation activities usually include checks for: completeness, consistency, validity, realism, ambiguity, and verifiability. Test case generation, prototyping, requirements reviews, automated consistency analysis, and walk-throughs are typical requirements validation techniques. Feasibility analysis looks at the risks associated with a given requirement and its potential impact to the system or product under development as a whole.

Figure 22: Requirements Validation

Documentation:

Requirements validation is used to check for errors at the initial phase of development, as the error may increase excessive rework when detected later in the development process. Requirements validation activities usually include checks for: completeness, consistency, validity, realism, ambiguity, and verifiability. Test case generation, prototyping, requirements reviews, automated consistency analysis, and walk-throughs are typical requirements validation techniques.

Feasibility analysis looks at the risks associated with a given requirement and its potential impact to the system or product under development as a whole.

3.2.5 Req_5 Change Management of Requirements

The system shall be able to manage changes to the requirements.

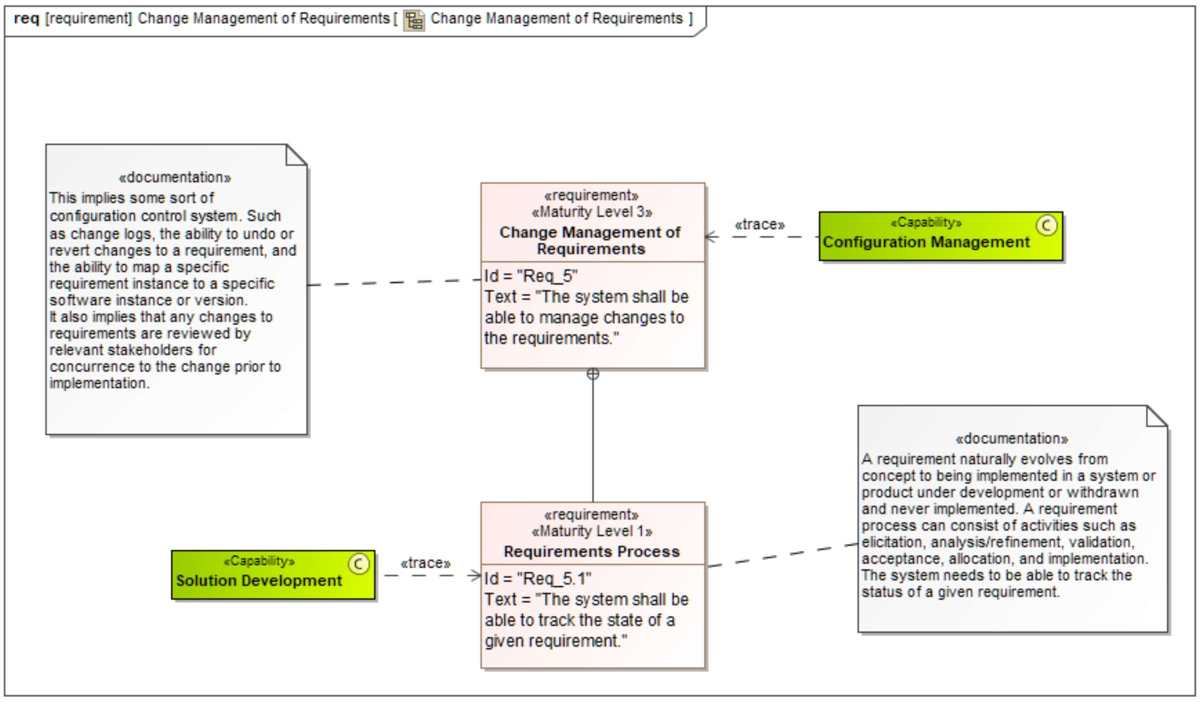

req [requirement] Change Management of Requirements [Change Management of Requirements]
«documentation»
This implies some sort of configuration control system. Such as change logs, the ability to undo or revert changes to a requirement, and the ability to map a specific requirement instance to a specific software instance or version.
It also implies that any changes to requirements are reviewed by relevant stakeholders for concurrence to the change prior to implementation.
«requirement» «Maturity Level 3» Change Management of Requirements
Id = "Req_5"
Text = "The system shall be able to manage changes to the requirements."
«trace»
«Capability» Configuration Management
«Capability» Solution Development
«trace»
«requirement» «Maturity Level 1» Requirements Process
Id = "Req_5.1"
Text = "The system shall be able to track the state of a given requirement."
«documentation»
A requirement naturally evolves from concept to being implemented in a system or product under development or withdrawn and never implemented. A requirement process can consist of activities such as elicitation, analysis/refinement, validation, acceptance, allocation, and implementation. The system needs to be able to track the status of a given requirement.

Figure 23: Change Management of Requirements

Documentation:

This implies some sort of configuration control system. Such as change logs, the ability to undo or revert changes to a requirement, and the ability to map a specific requirement instance to a specific software instance or version.

It also implies that any changes to requirements are reviewed by relevant stakeholders for concurrence to the change prior to implementation.

3.2.5.1 Req_5.1 Requirements Process

The system shall be able to track the state of a given requirement.

Documentation:

A requirement naturally evolves from concept to being implemented in a system or product under development or withdrawn and never implemented. A requirement process can consist of activities such as elicitation, analysis/refinement, validation, acceptance, allocation, and implementation. The system needs to be able to track the status of a given requirement.

3.2.6 Req_6 Requirements Authorization

The system shall be able to assign role-based authorization to view, modify, and/or create requirements to individual users.

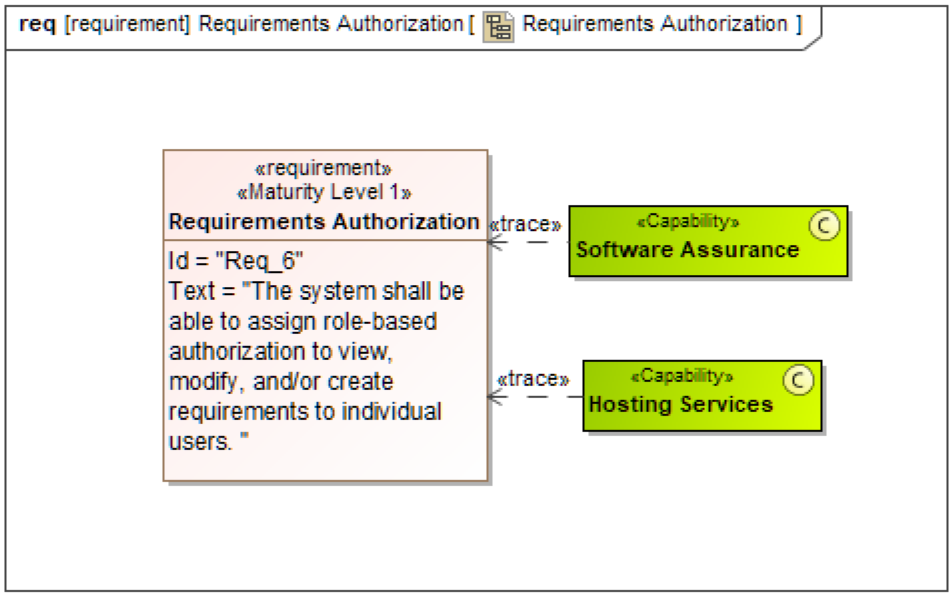

req [requirement] Requirements Authorization [Requirements Authorization]
«requirement»
«Maturity Level 1»
Requirements Authorization
Id = "Req_6"
Text = "The system shall be able to assign role-based authorization to view, modify, and/or create requirements to individual users. "
«trace»
«Capability»
Software Assurance
«trace»
«Capability»
Hosting Services

Figure 24: Requirements Authorization

3.3 3 Architecture & Design

3.3.1 Arc_1 Requirement Mapping

The system shall be able to trace architectural elements to requirements.

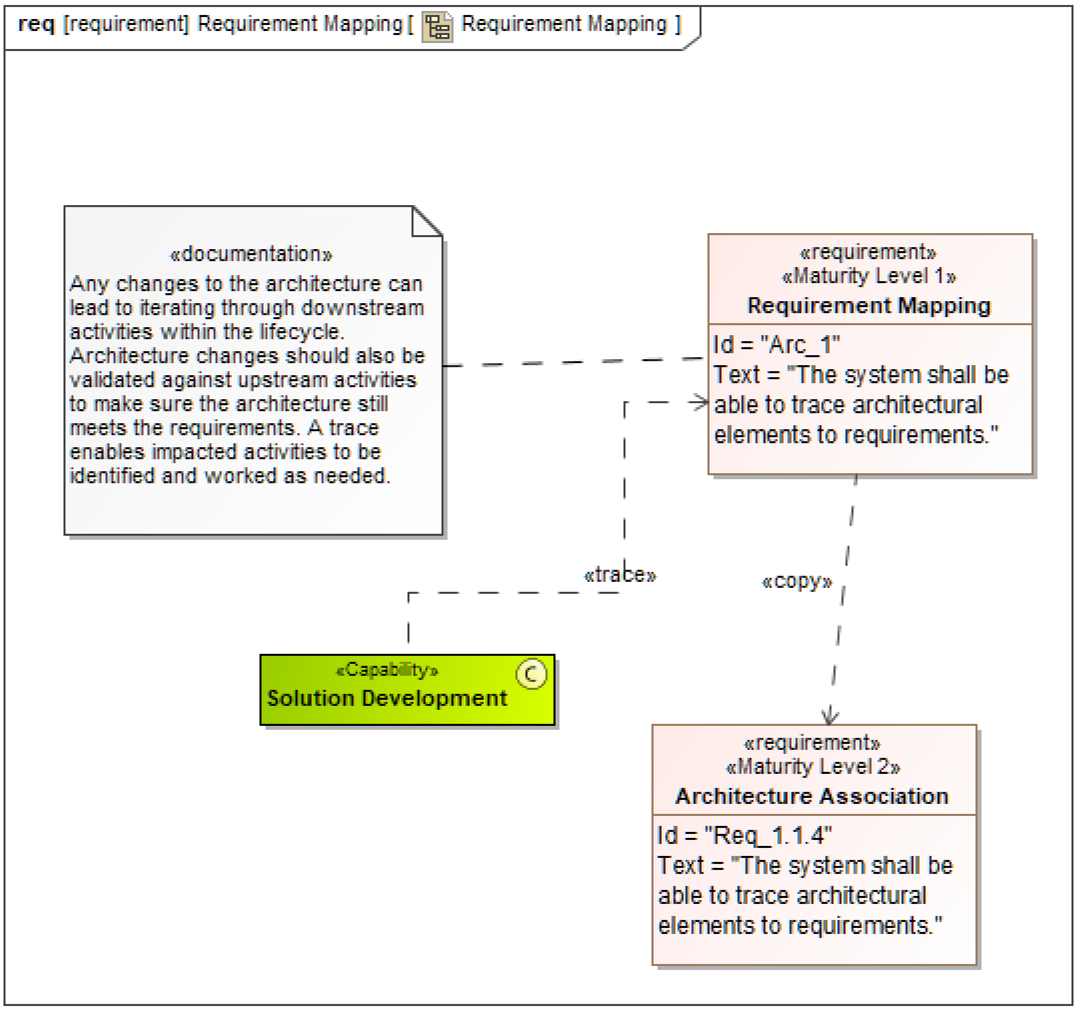

req [requirement] Requirement Mapping [Requirement Mapping]
«documentation»
Any changes to the architecture can lead to iterating through downstream activities within the lifecycle. Architecture changes should also be validated against upstream activities to make sure the architecture still meets the requirements. A trace enables impacted activities to be identified and worked as needed.
«requirement»
«Maturity Level 1»
Requirement Mapping
Id = "Arc_1"
Text = "The system shall be able to trace architectural elements to requirements."
«trace»
«copy»
«Capability»
Solution Development
«requirement»
«Maturity Level 2»
Architecture Association
Id = "Req_1.1.4"
Text = "The system shall be able to trace architectural elements to requirements."

Figure 25: Requirement Mapping

Documentation:

Any changes to the architecture can lead to iterating through downstream activities within the lifecycle. Architecture changes should also be validated against upstream activities to make sure the architecture still meets the requirements. A trace enables impacted activities to be identified and worked as needed.

3.3.2 Arc_2 Implementation Mapping

The system shall be able to trace architectural elements to implementation elements.

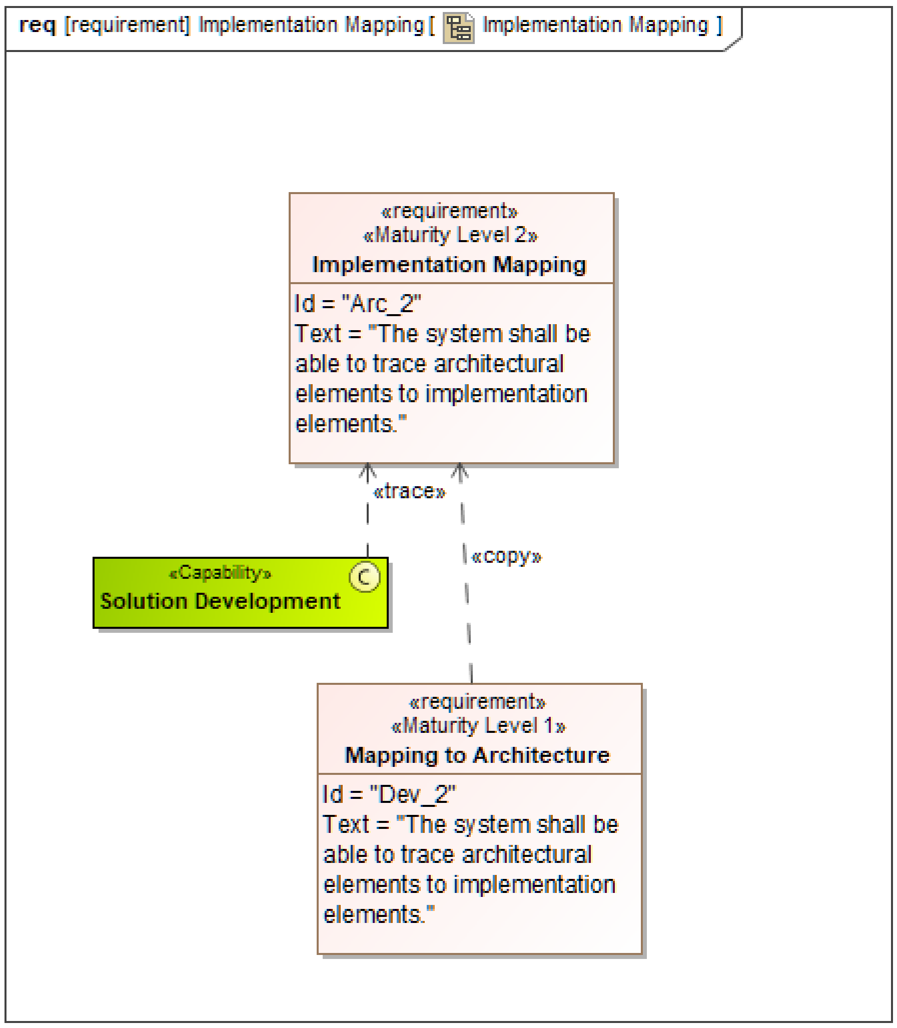

req [requirement] Implementation Mapping [Implementation Mapping]
«requirement»
«Maturity Level 2»
Implementation Mapping
Id = "Arc_2"
Text = "The system shall be able to trace architectural elements to implementation elements."
«trace»
«copy»
«Capability»
Solution Development
«requirement»
«Maturity Level 1»
Mapping to Architecture
Id = "Dev_2"
Text = "The system shall be able to trace architectural elements to implementation elements."

Figure 26: Implementation Mapping

3.3.3 Arc_3 MBSE

The system shall be able to support model-based systems engineering (MBSE).

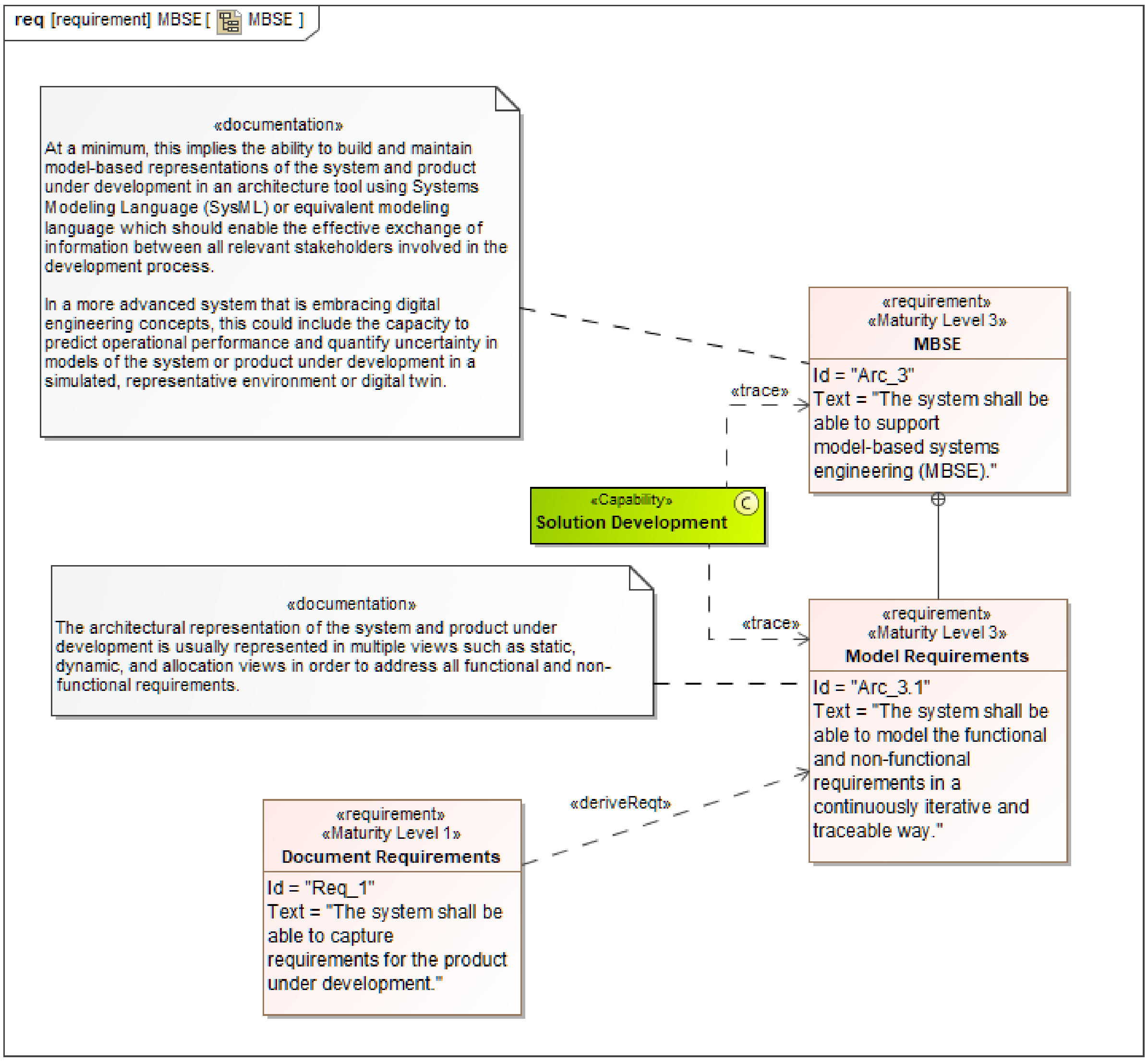

req [requirement] MBSE [MBSE]
«documentation»
At a minimum, this implies the ability to build and maintain model-based representations of the system and product under development in an architecture tool using Systems Modeling Language (SysML) or equivalent modeling language which should enable the effective exchange of information between all relevant stakeholders involved in the development process.
In a more advanced system that is embracing digital engineering concepts, this could include the capacity to predict operational performance and quantify uncertainty in models of the system or product under development in a simulated, representative environment or digital twin.
«requirement» «Maturity Level 3» MBSE
Id = "Arc_3"
Text = "The system shall be able to support model-based systems engineering (MBSE)."
«trace»
«Capability» Solution Development
«documentation»
The architectural representation of the system and product under development is usually represented in multiple views such as static, dynamic, and allocation views in order to address all functional and non-functional requirements.
«trace»
«requirement» «Maturity Level 3» Model Requirements
Id = "Arc_3.1"
Text = "The system shall be able to model the functional and non-functional requirements in a continuously iterative and traceable way."
«deriveReqt»
«requirement» «Maturity Level 1» Document Requirements
Id = "Req_1"
Text = "The system shall be able to capture requirements for the product under development."

Figure 27: MBSE

Documentation:

At a minimum, this implies the ability to build and maintain model-based representations of the system and product under development in an architecture tool using Systems Modeling Language (SysML) or equivalent modeling language which should enable the effective exchange of information between all relevant stakeholders involved in the development process.

In a more advanced system that is embracing digital engineering concepts, this could include the capacity to predict operational performance and quantify uncertainty in models of the system or product under development in a simulated, representative environment or digital twin.

3.3.3.1 Arc_3.1 Model Requirements

The system shall be able to model the functional and non-functional requirements in a continuously iterative and traceable way.

Documentation:

The architectural representation of the system and product under development is usually represented in multiple views such as static, dynamic, and allocation views in order to address all functional and non-functional requirements.

3.3.4 Arc_4 Software Assurance

The system shall be able to support the incorporation of security and privacy engineering principles into the product under development's architecture and design.

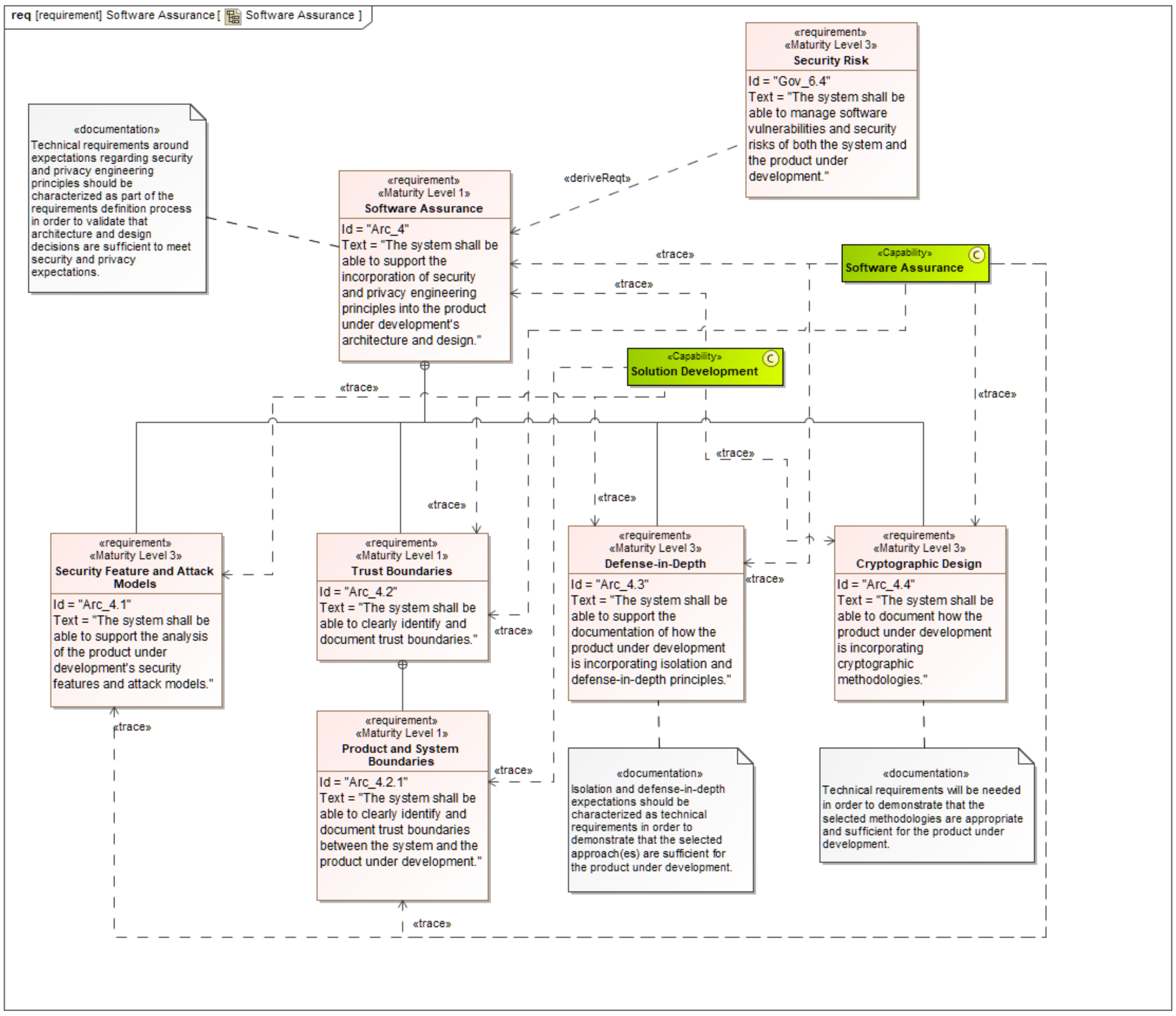

req [requirement] Software Assurance [Software Assurance]
«documentation» Technical requirements around expectations regarding security and privacy engineering principles should be characterized as part of the requirements definition process in order to validate that architecture and design decisions are sufficient to meet security and privacy expectations.
«requirement» «Maturity Level 1» Software Assurance Id = "Arc_4" Text = "The system shall be able to support the incorporation of security and privacy engineering principles into the product under development's architecture and design."
«requirement» «Maturity Level 3» Security Risk Id = "Gov_6.4" Text = "The system shall be able to manage software vulnerabilities and security risks of both the system and the product under development."
«deriveReqt»
«trace»
«Capability» Software Assurance
«Capability» Solution Development
«requirement» «Maturity Level 3» Security Feature and Attack Models Id = "Arc_4.1" Text = "The system shall be able to support the analysis of the product under development's security features and attack models."
«requirement» «Maturity Level 1» Trust Boundaries Id = "Arc_4.2" Text = "The system shall be able to clearly identify and document trust boundaries."
«requirement» «Maturity Level 3» Defense-in-Depth Id = "Arc_4.3" Text = "The system shall be able to support the documentation of how the product under development is incorporating isolation and defense-in-depth principles."
«requirement» «Maturity Level 3» Cryptographic Design Id = "Arc_4.4" Text = "The system shall be able to document how the product under development is incorporating cryptographic methodologies."
«requirement» «Maturity Level 1» Product and System Boundaries Id = "Arc_4.2.1" Text = "The system shall be able to clearly identify and document trust boundaries between the system and the product under development."
«documentation» Isolation and defense-in-depth expectations should be characterized as technical requirements in order to demonstrate that the selected approach(es) are sufficient for the product under development.
«documentation» Technical requirements will be needed in order to demonstrate that the selected methodologies are appropriate and sufficient for the product under development.

Figure 28: Software Assurance

Documentation:

Technical requirements around expectations regarding security and privacy engineering principles should be characterized as part of the requirements definition process in order to validate that architecture and design decisions are sufficient to meet security and privacy expectations.

3.3.4.1 Arc_4.1 Security Feature and Attack Models

The system shall be able to support the analysis of the product under development's security features and attack models.

3.3.4.2 Arc_4.2 Trust Boundaries

The system shall be able to clearly identify and document trust boundaries.

3.3.4.2.1 Arc_4.2.1 Product and System Boundaries

The system shall be able to clearly identify and document trust boundaries between the system and the product under development.

3.3.4.3 Arc_4.3 Defense-in-Depth

The system shall be able to support the documentation of how the product under development is incorporating isolation and defense-in-depth principles.

Documentation:

Isolation and defense-in-depth expectations should be characterized as technical requirements in order to demonstrate that the selected approach(es) are sufficient for the product under development.

3.3.4.4 Arc_4.4 Cryptographic Design

The system shall be able to document how the product under development is incorporating cryptographic methodologies.

Documentation:

Technical requirements will be needed in order to demonstrate that the selected methodologies are appropriate and sufficient for the product under development.

3.4 4 Development

3.4.1 Dev_1 Mapping to Requirements

The system shall be able to trace implementation elements to requirements.

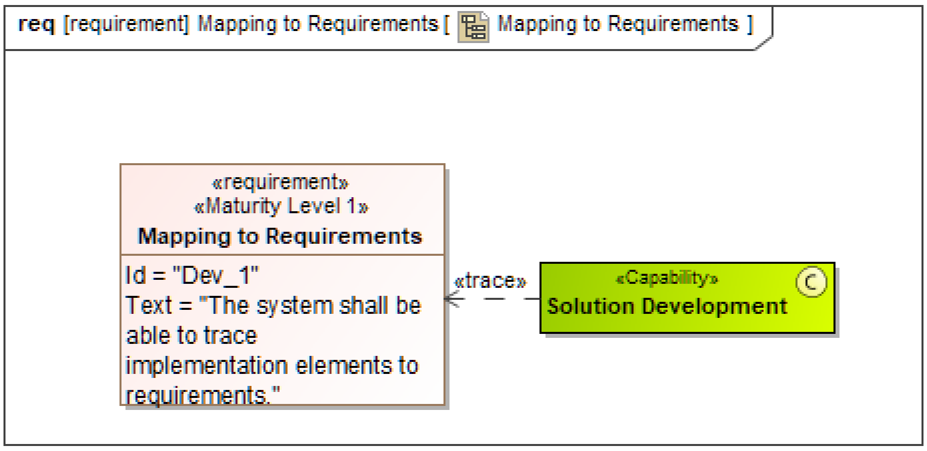

req [requirement] Mapping to Requirements [Mapping to Requirements]
«requirement»
«Maturity Level 1»
Mapping to Requirements
Id = "Dev_1"
Text = "The system shall be able to trace implementation elements to requirements."
«trace»
«Capability»
Solution Development

Figure 29: Mapping to Requirements

3.4.2 Dev_2 Mapping to Architecture

The system shall be able to trace architectural elements to implementation elements.

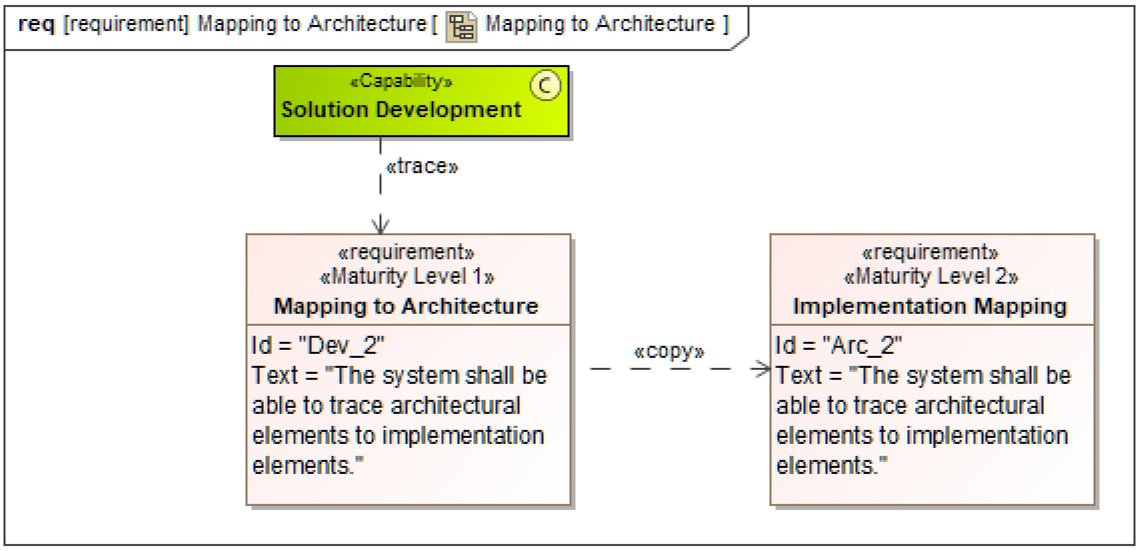

req [requirement] Mapping to Architecture [Mapping to Architecture]
«Capability»
Solution Development
«trace»
«requirement»
«Maturity Level 1»
Mapping to Architecture
Id = "Dev_2"
Text = "The system shall be able to trace architectural elements to implementation elements."
«copy»
«requirement»
«Maturity Level 2»
Implementation Mapping
Id = "Arc_2"
Text = "The system shall be able to trace architectural elements to implementation elements."

Figure 30: Mapping to Architecture

3.4.3 Dev_3 Mapping to Tests

The system shall be able to trace implementation elements to test artifacts.

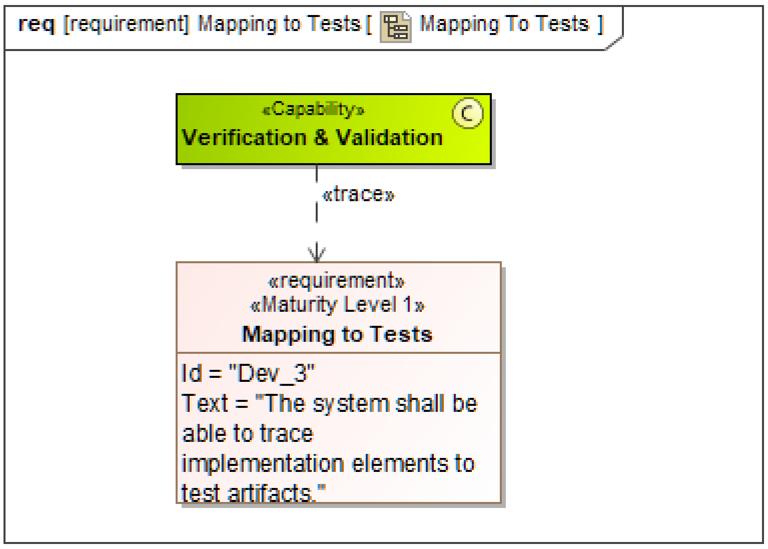

req [requirement] Mapping to Tests [Mapping To Tests]
«Capability»
Verification & Validation
«trace»
«requirement»
«Maturity Level 1»
Mapping to Tests
Id = "Dev_3"
Text = "The system shall be able to trace implementation elements to test artifacts."

Figure 31: Mapping To Tests

3.4.4 Dev_4 Secure Software Development

The system shall be able to support the use of secure coding practices and tooling throughout the development lifecycle.

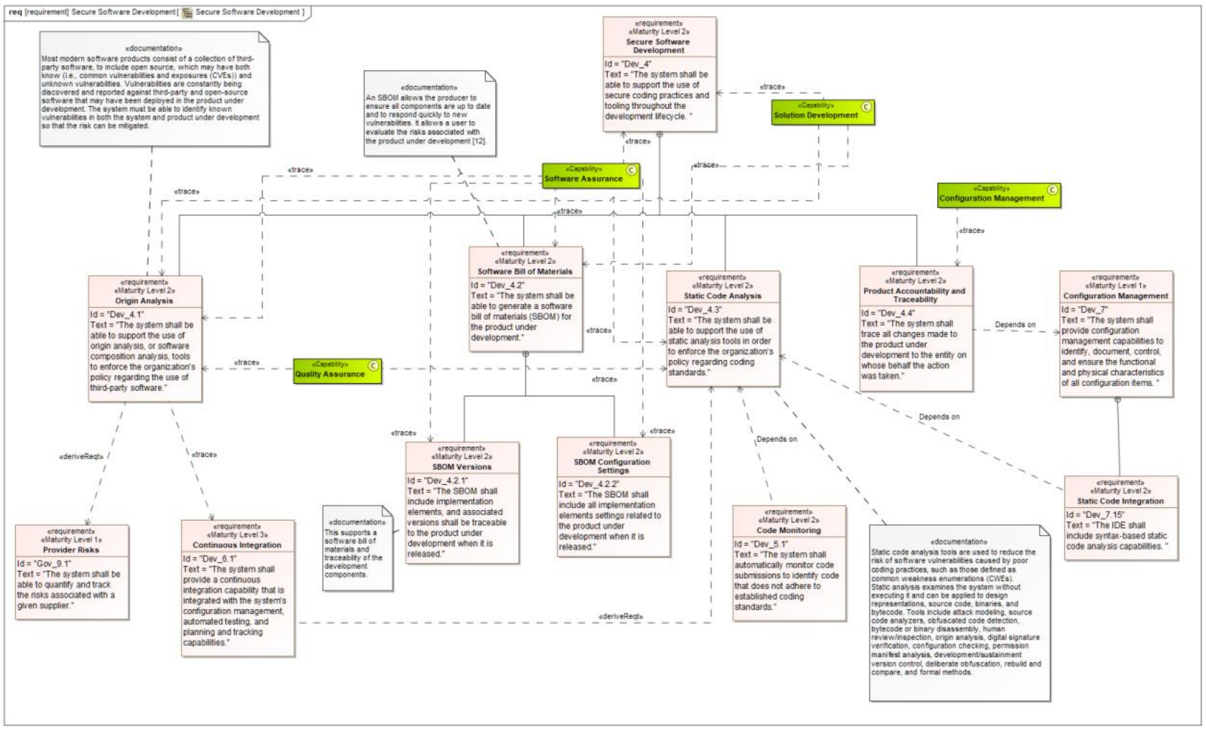

req [requirement] Secure Software Development [Secure Software Development]
«documentation» Most modern software products consist of a collection of third-party software, to include open source, which may have both know (i.e., common vulnerabilities and exposures (CVEs)) and unknown vulnerabilities. Vulnerabilities are constantly being discovered and reported against third-party and open-source software that may have been deployed in the product under development. The system must be able to identify known vulnerabilities in both the system and product under development so that the risk can be mitigated.
«documentation» An SBOM allows the producer to ensure all components are up to date and to respond quickly to new vulnerabilities. It allows a user to evaluate the risks associated with the product under development [12].
«requirement» «Maturity Level 2» Secure Software Development Id = "Dev_4" Text = "The system shall be able to support the use of secure coding practices and tooling throughout the development lifecycle. "
«Capability» Solution Development
«Capability» Software Assurance
«Capability» Configuration Management
«Capability» Quality Assurance
«requirement» «Maturity Level 2» Origin Analysis Id = "Dev_4.1" Text = "The system shall be able to support the use of origin analysis, or software composition analysis, tools to enforce the organization's policy regarding the use of third-party software."
«requirement» «Maturity Level 2» Software Bill of Materials Id = "Dev_4.2" Text = "The system shall be able to generate a software bill of materials (SBOM) for the product under development."
«requirement» «Maturity Level 2» Static Code Analysis Id = "Dev_4.3" Text = "The system shall be able to support the use of static analysis tools in order to enforce the organization's policy regarding coding standards."
«requirement» «Maturity Level 2» Product Accountability and Traceability Id = "Dev_4.4" Text = "The system shall trace all changes made to the product under development to the entity on whose behalf the action was taken."
«requirement» «Maturity Level 1» Configuration Management Id = "Dev_7" Text = "The system shall provide configuration management capabilities to identify, document, control, and ensure the functional and physical characteristics of all configuration items. "
«requirement» «Maturity Level 1» Provider Risks Id = "Gov_9.1" Text = "The system shall be able to quantify and track the risks associated with a given supplier."
«requirement» «Maturity Level 3» Continuous Integration Id = "Dev_6.1" Text = "The system shall provide a continuous integration capability that is integrated with the system's configuration management, automated testing, and planning and tracking capabilities."
«documentation» This supports a software bill of materials and traceability of the development components.
«requirement» «Maturity Level 2» SBOM Versions Id = "Dev_4.2.1" Text = "The SBOM shall include implementation elements, and associated versions shall be traceable to the product under development when it is released."
«requirement» «Maturity Level 2» SBOM Configuration Settings Id = "Dev_4.2.2" Text = "The SBOM shall include all implementation elements settings related to the product under development when it is released."
«requirement» «Maturity Level 2» Code Monitoring Id = "Dev_5.1" Text = "The system shall automatically monitor code submissions to identify code that does not adhere to established coding standards."
«documentation» Static code analysis tools are used to reduce the risk of software vulnerabilities caused by poor coding practices, such as those defined as common weakness enumerations (CWEs). Static analysis examines the system without executing it and can be applied to design representations, source code, binaries, and bytecode. Tools include attack modeling, source code analyzers, obfuscated code detection, bytecode or binary disassembly, human review/inspection, origin analysis, digital signature verification, configuration checking, permission manifest analysis, development/sustainment version control, deliberate obfuscation, rebuild and compare, and formal methods.
«requirement» «Maturity Level 2» Static Code Integration Id = "Dev_7.15" Text = "The IDE shall include syntax-based static code analysis capabilities. "
«trace»
«deriveReqt»
Depends on

Figure 32: Secure Software Development

3.4.4.1 Dev_4.1 Origin Analysis

The system shall be able to support the use of origin analysis, or software composition analysis, tools to enforce the organization's policy regarding the use of third-party software.

Documentation:

Most modern software products consist of a collection of third-party software, to include open source, which may have both know (i.e., common vulnerabilities and exposures (CVEs)) and unknown vulnerabilities. Vulnerabilities are constantly being discovered and reported against third-party and open-source software that may have been deployed in the product under development. The system must be able to identify known vulnerabilities in both the system and product under development so that the risk can be mitigated.

3.4.4.2 Dev_4.2 Software Bill of Materials

The system shall be able to generate a software bill of materials (SBOM) for the product under development.

Documentation:

An SBOM allows the producer to ensure all components are up to date and to respond quickly to new vulnerabilities. It allows a user to evaluate the risks associated with the product under development [Wikipedia 2021c].

3.4.4.2.1 Dev_4.2.1 SBOM Versions

The SBOM shall include implementation elements, and associated versions shall be traceable to the product under development when it is released.

Documentation:

This supports a software bill of materials and traceability of the development components.

3.4.4.2.2 Dev_4.2.2 SBOM Configuration Settings

The SBOM shall include all implementation elements settings related to the product under development when it is released.

3.4.4.3 Dev_4.3 Static Code Analysis

The system shall be able to support the use of static analysis tools in order to enforce the organization's policy regarding coding standards.

Documentation:

Static code analysis tools are used to reduce the risk of software vulnerabilities caused by poor coding practices, such as those defined as common weakness enumerations (CWEs).

Static analysis examines the system without executing it and can be applied to design representations, source code, binaries, and bytecode. Tools include attack modeling, source code analyzers, obfuscated code detection, bytecode or binary disassembly, human review/inspection, origin analysis, digital signature verification, configuration checking, permission manifest analysis, development/sustainment version control, deliberate obfuscation, rebuild and compare, and formal methods.

3.4.4.4 Dev_4.4 Product Accountability and Traceability

The system shall trace all changes made to the product under development to the entity on whose behalf the action was taken.

3.4.5 Dev_5 Code Reviews

The system shall be able to support both informal and formal code reviews.

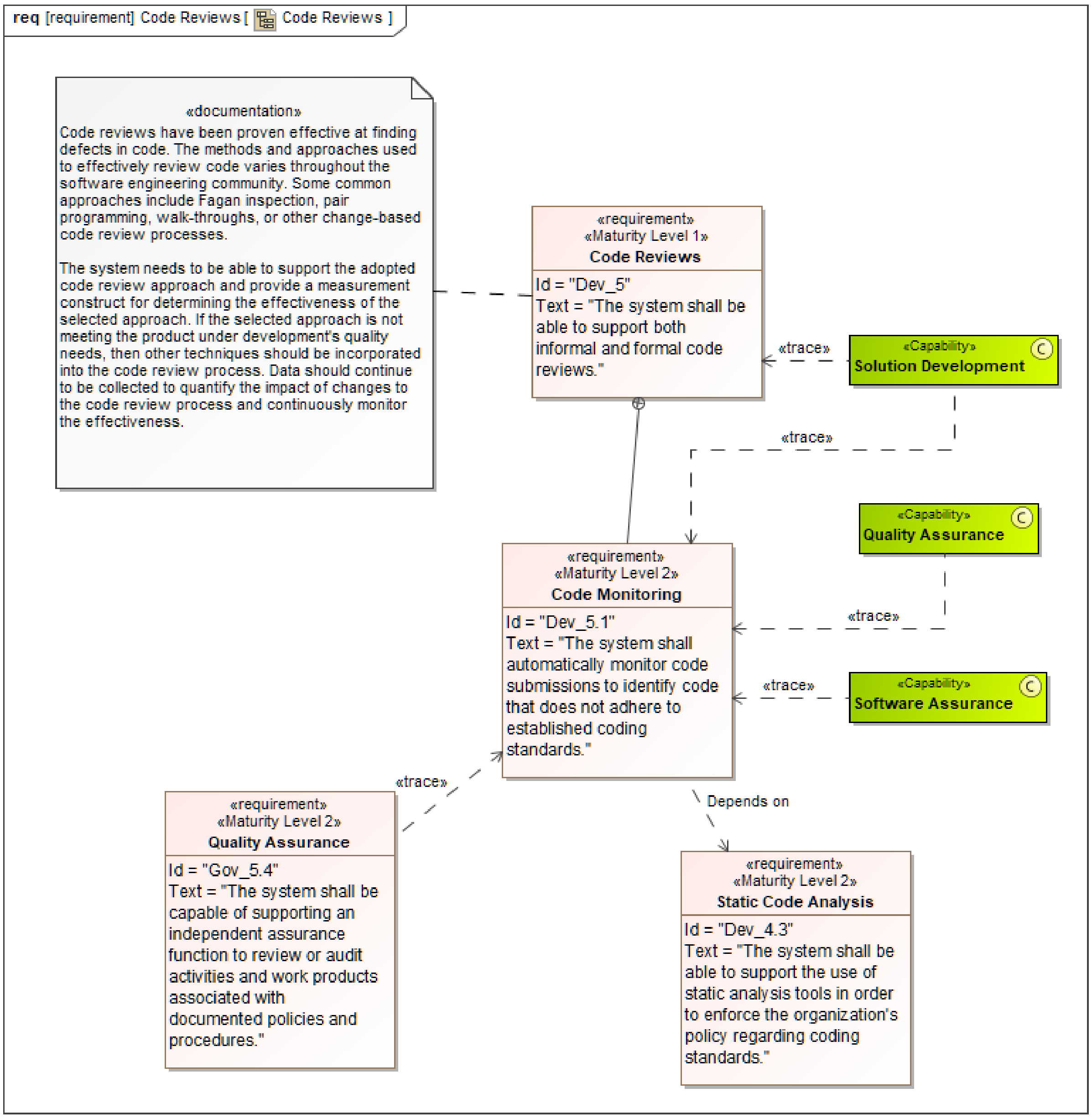

req [requirement] Code Reviews [Code Reviews]
«documentation»
Code reviews have been proven effective at finding defects in code. The methods and approaches used to effectively review code varies throughout the software engineering community. Some common approaches include Fagan inspection, pair programming, walk-throughs, or other change-based code review processes.
The system needs to be able to support the adopted code review approach and provide a measurement construct for determining the effectiveness of the selected approach. If the selected approach is not meeting the product under development's quality needs, then other techniques should be incorporated into the code review process. Data should continue to be collected to quantify the impact of changes to the code review process and continuously monitor the effectiveness.
«requirement» «Maturity Level 1» Code Reviews
Id = "Dev_5"
Text = "The system shall be able to support both informal and formal code reviews."
«trace»
«Capability» Solution Development
«trace»
«Capability» Quality Assurance
«requirement» «Maturity Level 2» Code Monitoring
Id = "Dev_5.1"
Text = "The system shall automatically monitor code submissions to identify code that does not adhere to established coding standards."
«trace»
«trace»
«Capability» Software Assurance
«trace»
«requirement» «Maturity Level 2» Quality Assurance
Id = "Gov_5.4"
Text = "The system shall be capable of supporting an independent assurance function to review or audit activities and work products associated with documented policies and procedures."
Depends on
«requirement» «Maturity Level 2» Static Code Analysis
Id = "Dev_4.3"
Text = "The system shall be able to support the use of static analysis tools in order to enforce the organization's policy regarding coding standards."

Figure 33: Code Reviews

Documentation:

Code reviews have been proven effective at finding defects in code. The methods and approaches used to effectively review code varies throughout the software engineering community. Some common approaches include Fagan inspection, pair programming, walk-throughs, or other change-based code review processes.

The system needs to be able to support the adopted code review approach and provide a measurement construct for determining the effectiveness of the selected approach. If the selected approach is not meeting the product under development's quality needs, then other techniques should be incorporated into the code review process. Data should continue to be collected to quantify the impact of changes to the code review process and continuously monitor the effectiveness.

3.4.5.1 Dev_5.1 Code Monitoring

The system shall automatically monitor code submissions to identify code that does not adhere to established coding standards.

3.4.6 Dev_6 Orchestration

The system shall provide orchestration capabilities to automatically coordinate and execute many tasks together.

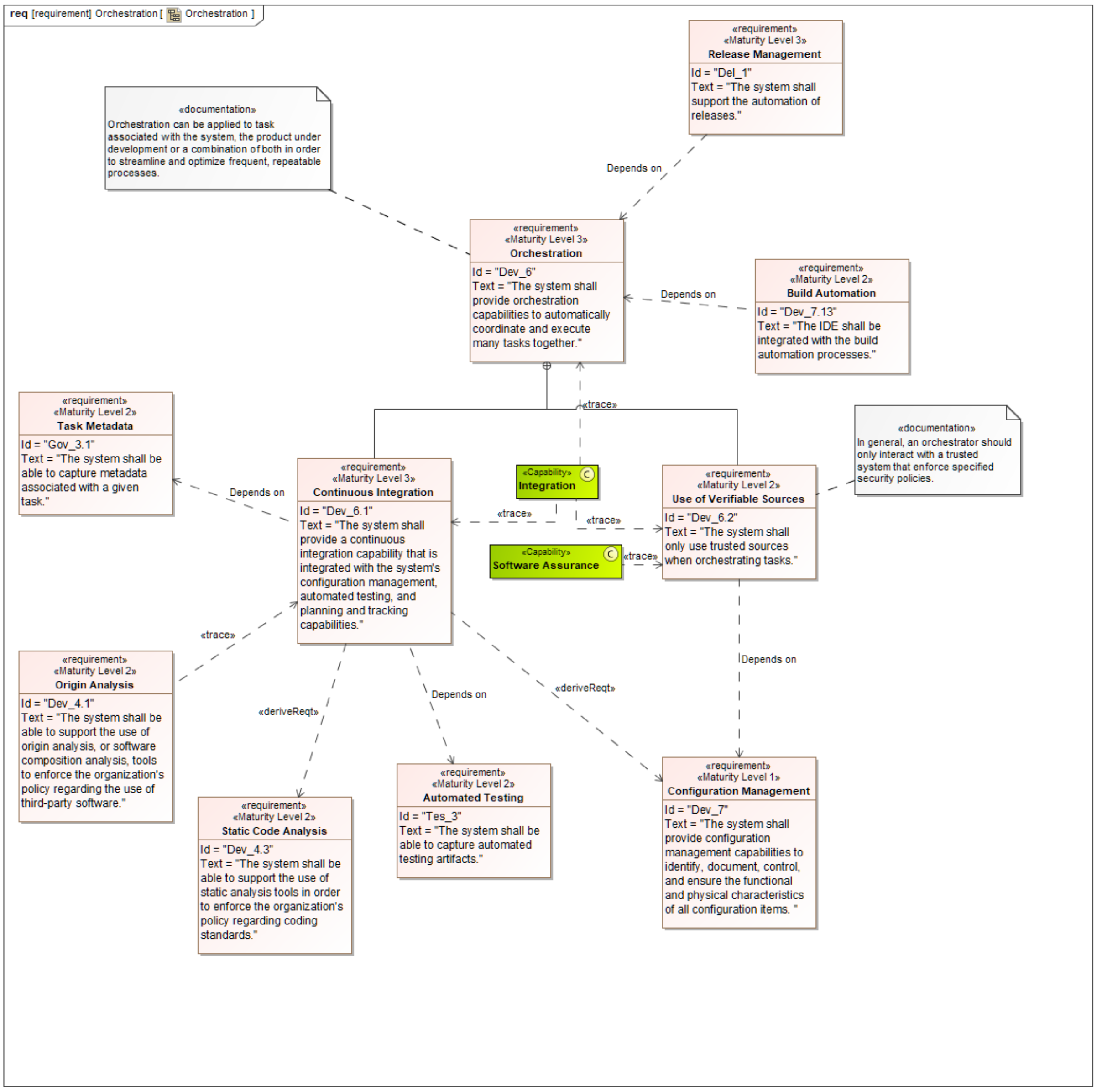

req [requirement] Orchestration [Orchestration]
«requirement» «Maturity Level 3» Release Management Id = "Del_1" Text = "The system shall support the automation of releases."
«documentation» Orchestration can be applied to task associated with the system, the product under development or a combination of both in order to streamline and optimize frequent, repeatable processes.
Depends on
«requirement» «Maturity Level 3» Orchestration Id = "Dev_6" Text = "The system shall provide orchestration capabilities to automatically coordinate and execute many tasks together."
Depends on
«requirement» «Maturity Level 2» Build Automation Id = "Dev_7.13" Text = "The IDE shall be integrated with the build automation processes."
«trace»
«requirement» «Maturity Level 2» Task Metadata Id = "Gov_3.1" Text = "The system shall be able to capture metadata associated with a given task."
Depends on
«requirement» «Maturity Level 3» Continuous Integration Id = "Dev_6.1" Text = "The system shall provide a continuous integration capability that is integrated with the system's configuration management, automated testing, and planning and tracking capabilities."
«Capability» Integration
«trace»
«trace»
«Capability» Software Assurance
«trace»
«requirement» «Maturity Level 2» Use of Verifiable Sources Id = "Dev_6.2" Text = "The system shall only use trusted sources when orchestrating tasks."
«documentation» In general, an orchestrator should only interact with a trusted system that enforce specified security policies.
«trace»
«requirement» «Maturity Level 2» Origin Analysis Id = "Dev_4.1" Text = "The system shall be able to support the use of origin analysis, or software composition analysis, tools to enforce the organization's policy regarding the use of third-party software."
«deriveReqt»
Depends on
«deriveReqt»
Depends on
«requirement» «Maturity Level 2» Static Code Analysis Id = "Dev_4.3" Text = "The system shall be able to support the use of static analysis tools in order to enforce the organization's policy regarding coding standards."
«requirement» «Maturity Level 2» Automated Testing Id = "Tes_3" Text = "The system shall be able to capture automated testing artifacts."
«requirement» «Maturity Level 1» Configuration Management Id = "Dev_7" Text = "The system shall provide configuration management capabilities to identify, document, control, and ensure the functional and physical characteristics of all configuration items."

Figure 34: Orchestration

Documentation:

Orchestration can be applied to task associated with the system, the product under development or a combination of both in order to streamline and optimize frequent, repeatable processes.

3.4.6.1 Dev_6.1 Continuous Integration

The system shall provide a continuous integration capability that is integrated with the system's configuration management, automated testing, and planning and tracking capabilities.

3.4.6.2 Dev_6.2 Use of Verifiable Sources

The system shall only use trusted sources when orchestrating tasks.

Documentation:

In general, an orchestrator should only interact with a trusted system that enforce specified security policies.

3.4.7 Dev_7 Configuration Management

The system shall provide configuration management capabilities to identify, document, control, and ensure the functional and physical characteristics of all configuration items.

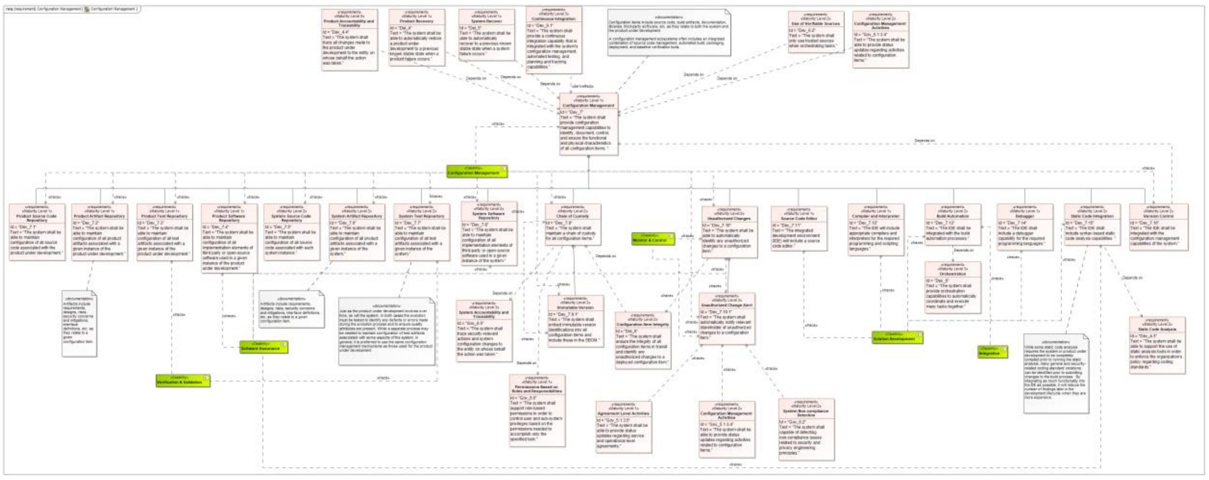


Figure 35: Configuration Management

Documentation:

Configuration items include source code, build artifacts, documentation, libraries, third-party software, etc. as they relate to both the system and the product under development.

A configuration management ecosystems often includes an integrated combination of source code management, automated build, packaging, deployment, and baseline verification tools.

3.4.7.1 Dev_7.1 Product Source Code Repository

The system shall be able to maintain configuration of all source code associated with the product under development.

3.4.7.2 Dev_7.2 Product Artifact Repository

The system shall be able to maintain configuration of all product artifacts associated with a given instance of the product under development.

Documentation:

Artifacts include requirements, designs, risks, security concerns and mitigations, interface definitions, etc. as they relate to a given configuration item.

3.4.7.3 Dev_7.3 Product Test Repository

The system shall be able to maintain configuration of all test artifacts associated with a given instance of the product under development.

3.4.7.4 Dev_7.4 Product Software Repository

The system shall be able to maintain configuration of all implementation elements of third party or open-source software used in a given instance of the product under development.

3.4.7.5 Dev_7.5 System Source Code Repository

The system shall be able to maintain configuration of all source code associated with each system instance.

3.4.7.6 Dev_7.6 System Artifact Repository

The system shall be able to maintain configuration of all product artifacts associated with a given instance of the system.

Documentation:

Artifacts include requirements, designs, risks, security concerns and mitigations, interface definitions, etc. as they relate to a given configuration item.

3.4.7.7 Dev_7.7 System Test Repository

The system shall be able to maintain configuration of all test artifacts associated with a given instance of the system.

Documentation:

Just as the product under development evolves over time, so will the system. In both cases the evolution must be tested to identify any defects or errors made during the evolution process and to ensure quality attributes are present. While a separate process may be needed to maintain configuration of test artifacts associated with some aspects of the system, in general, it is preferred to use the same configuration management mechanisms as those used for the product under development.

3.4.7.8 Dev_7.8 System Software Repository

The system shall be able to maintain configuration of all implementation elements of third-party or open-source software used in a given instance of the system.

3.4.7.9 Dev_7.9 Chain of Custody

The system shall maintain a chain of custody for all configuration items.

3.4.7.9.1 Dev_7.9.1 Immutable Version

The system shall embed immutable version identifications into all configuration items and include these in the SBOM.

3.4.7.10 Dev_7.10 Unauthorized Changes

The system shall be able to automatically identify any unauthorized changes to a configuration item.

3.4.7.10.1 Dev_7.10.1 Unauthorized Change Alert

The system shall automatically notify relevant stakeholder of unauthorized changes to a configuration item.

3.4.7.11 Dev_7.11 Source Code Editor

The integrated development environment (IDE) will include a source code editor.

3.4.7.12 Dev_7.12 Compiler and Interpreter

The IDE will include appropriate compilers and interpreters for the required programming and scripting languages.

3.4.7.13 Dev_7.13 Build Automation

The IDE shall be integrated with the build automation processes.

3.4.7.14 Dev_7.14 Debugger

The IDE shall include a debugger capability for the required programming languages.

3.4.7.15 Dev_7.15 Static Code Integration

The IDE shall include syntax-based static code analysis capabilities.

Documentation:

While some static code analysis requires the system or product under development to be completely compiled prior to running the static analysis, many general and security-related coding standard violations can be identified prior to submitting changes to the build process. By integrating as much functionality into the IDE as possible, it will reduce the number of findings later in the development lifecycle when they are more expensive.

3.4.7.16 Dev_7.16 Version Control

The IDE shall be integrated with the configuration management capabilities of the system.

3.4.8 Dev_8 Integrated Development Environment (IDE)

The system shall provide programmers with the comprehensive facilities needed to develop software for both the product under development's and system's targeted environments.

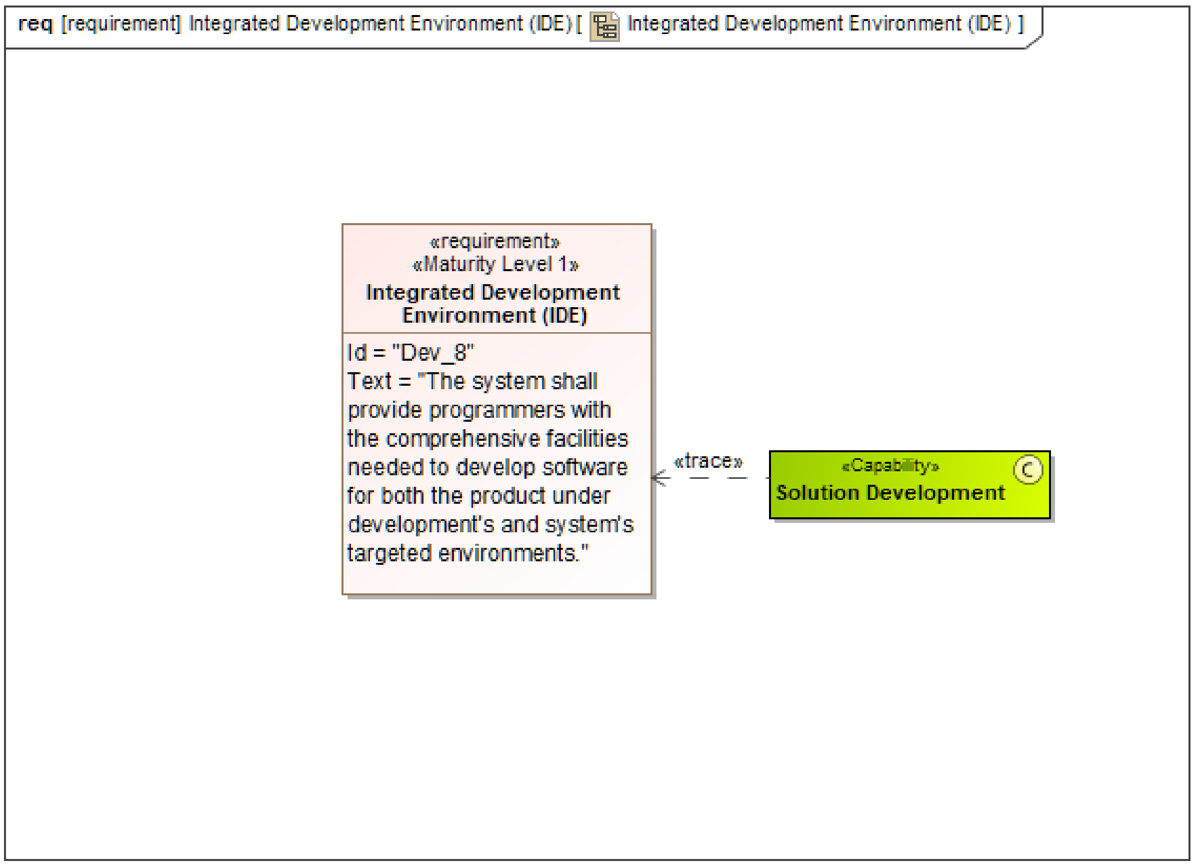

req [requirement] Integrated Development Environment (IDE) [Integrated Development Environment (IDE)]
«requirement»
«Maturity Level 1»
Integrated Development Environment (IDE)
Id = "Dev_8"
Text = "The system shall provide programmers with the comprehensive facilities needed to develop software for both the product under development's and system's targeted environments."
«trace»
«Capability»
Solution Development

Figure 36: Integrated Development Environment (IDE)

3.4.9 Dev_9 Development Information Radiator

The system shall support the near real-time reporting of development status to all relevant stakeholders.

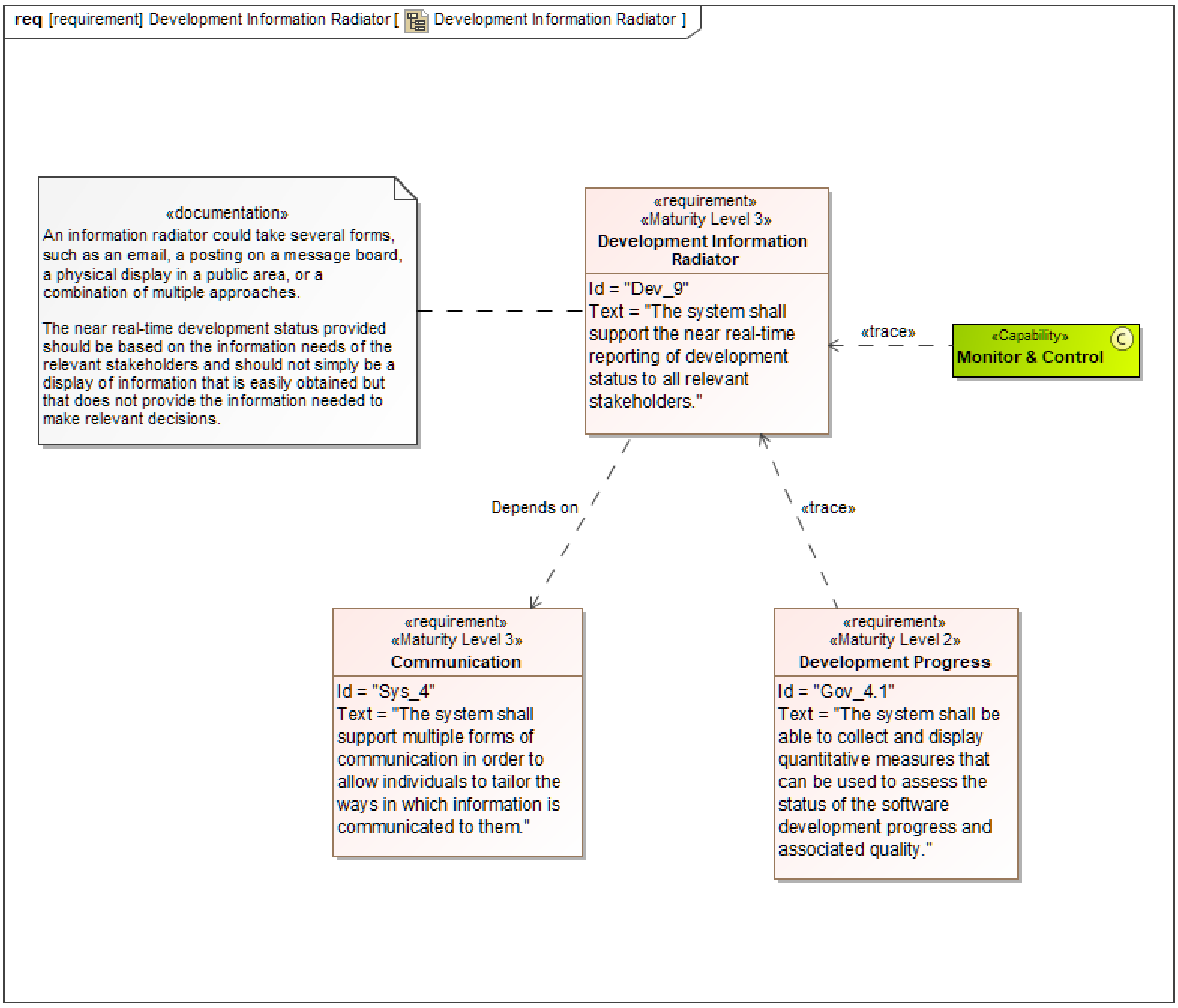

req [requirement] Development Information Radiator [Development Information Radiator]
«documentation»
An information radiator could take several forms, such as an email, a posting on a message board, a physical display in a public area, or a combination of multiple approaches.
The near real-time development status provided should be based on the information needs of the relevant stakeholders and should not simply be a display of information that is easily obtained but that does not provide the information needed to make relevant decisions.
«requirement» «Maturity Level 3» Development Information Radiator
Id = "Dev_9"
Text = "The system shall support the near real-time reporting of development status to all relevant stakeholders."
«trace»
«Capability» Monitor & Control
Depends on
«trace»
«requirement» «Maturity Level 3» Communication
Id = "Sys_4"
Text = "The system shall support multiple forms of communication in order to allow individuals to tailor the ways in which information is communicated to them."
«requirement» «Maturity Level 2» Development Progress
Id = "Gov_4.1"
Text = "The system shall be able to collect and display quantitative measures that can be used to assess the status of the software development progress and associated quality."

Figure 37: Development Information Radiator

Documentation:

An information radiator could take several forms, such as an email, a posting on a message board, a physical display in a public area, or a combination of multiple approaches.

The near real-time development status provided should be based on the information needs of the relevant stakeholders and should not simply be a display of information that is easily obtained but that does not provide the information needed to make relevant decisions.

3.5 5 Test

3.5.1 Tes_1 Manual Testing

The system shall be able to capture manual testing artifacts.

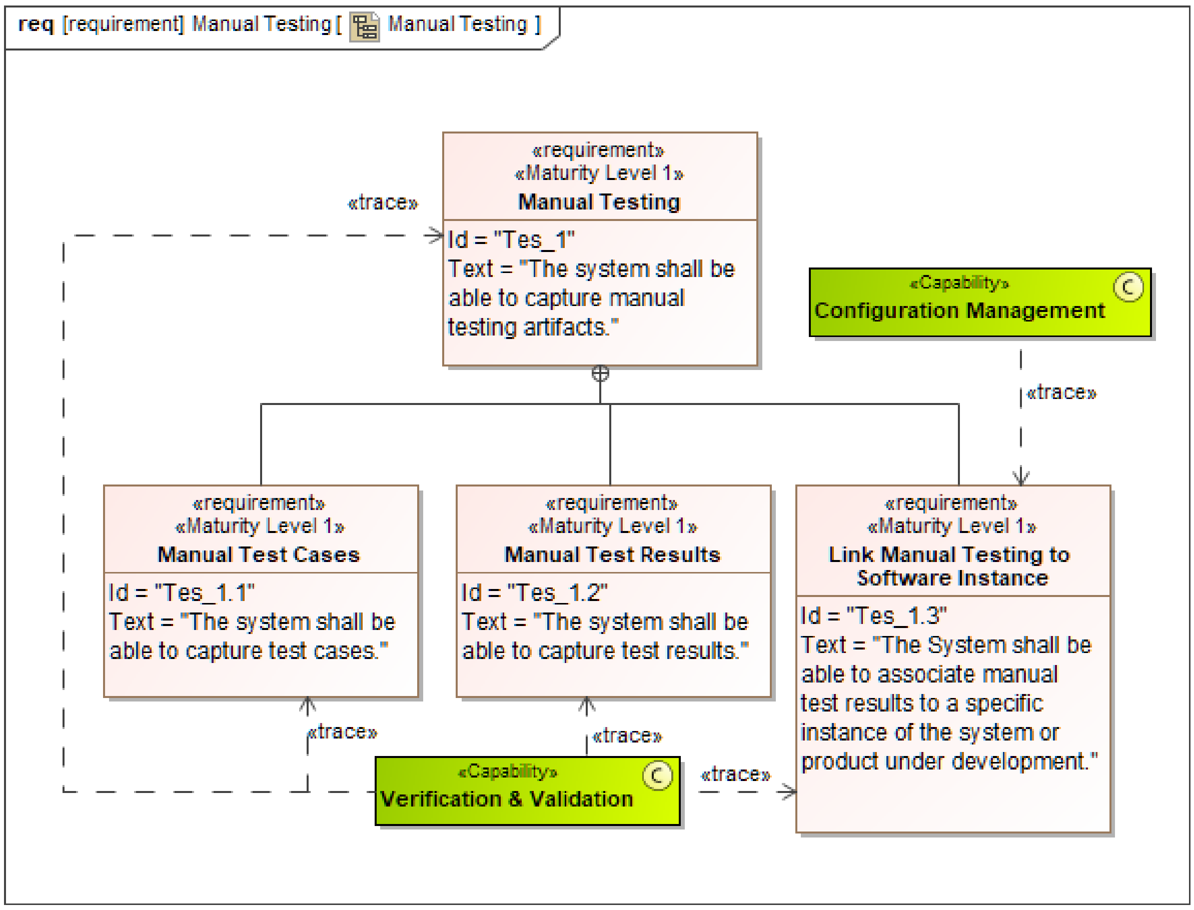

req [requirement] Manual Testing [Manual Testing]
«requirement» «Maturity Level 1» Manual Testing
Id = "Tes_1"
Text = "The system shall be able to capture manual testing artifacts."
«Capability» Configuration Management
«trace»
«trace»
«requirement» «Maturity Level 1» Manual Test Cases
Id = "Tes_1.1"
Text = "The system shall be able to capture test cases."
«requirement» «Maturity Level 1» Manual Test Results
Id = "Tes_1.2"
Text = "The system shall be able to capture test results."
«requirement» «Maturity Level 1» Link Manual Testing to Software Instance
Id = "Tes_1.3"
Text = "The System shall be able to associate manual test results to a specific instance of the system or product under development."
«trace»
«trace»
«trace»
«Capability» Verification & Validation

Figure 38: Manual Testing

3.5.1.1 Tes_1.1 Manual Test Cases

The system shall be able to capture test cases.

3.5.1.2 Tes_1.2 Manual Test Results

The system shall be able to capture test results.

3.5.1.3 Tes_1.3 Link Manual Testing to Software Instance

The System shall be able to associate manual test results to a specific instance of the system or product under development.

3.5.2 Tes_2 Requirement Association

The system shall be able to trace test artifacts to requirements.

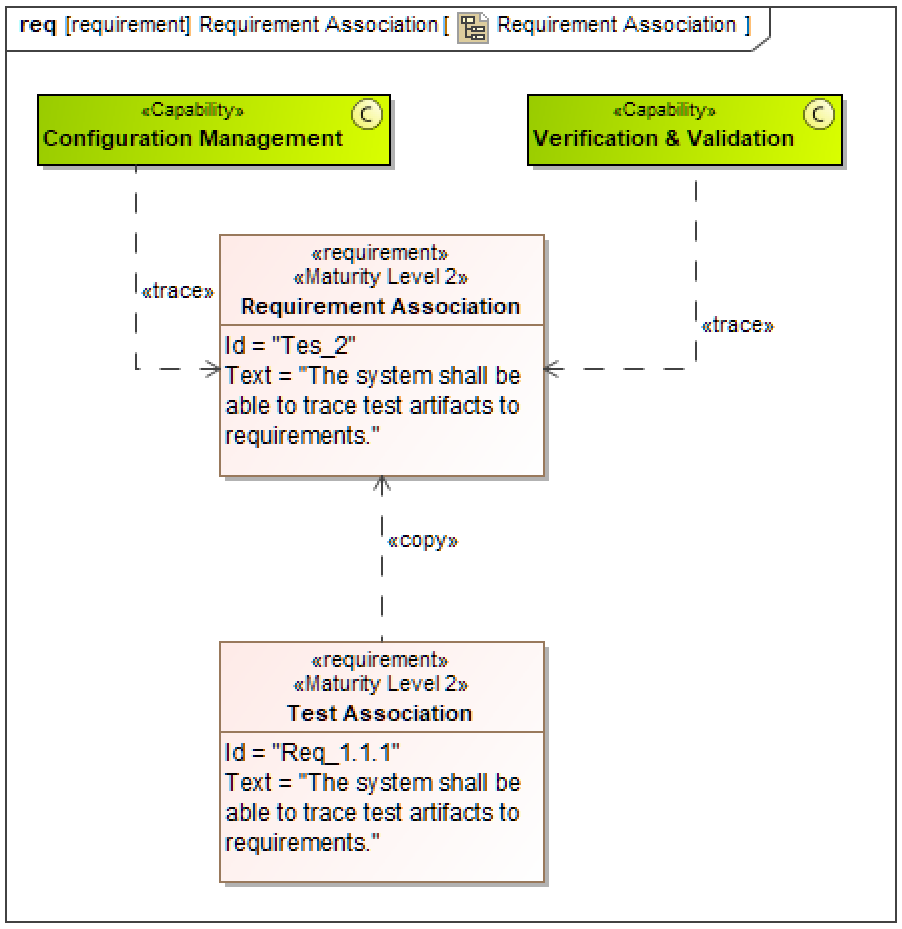

req [requirement] Requirement Association [Requirement Association]
«Capability»
Configuration Management
«Capability»
Verification & Validation
«trace»
«trace»
«requirement»
«Maturity Level 2»
Requirement Association
Id = "Tes_2"
Text = "The system shall be able to trace test artifacts to requirements."
«copy»
«requirement»
«Maturity Level 2»
Test Association
Id = "Req_1.1.1"
Text = "The system shall be able to trace test artifacts to requirements."

Figure 39: Requirement Association

3.5.3 Tes_3 Automated Testing

The system shall be able to capture automated testing artifacts.

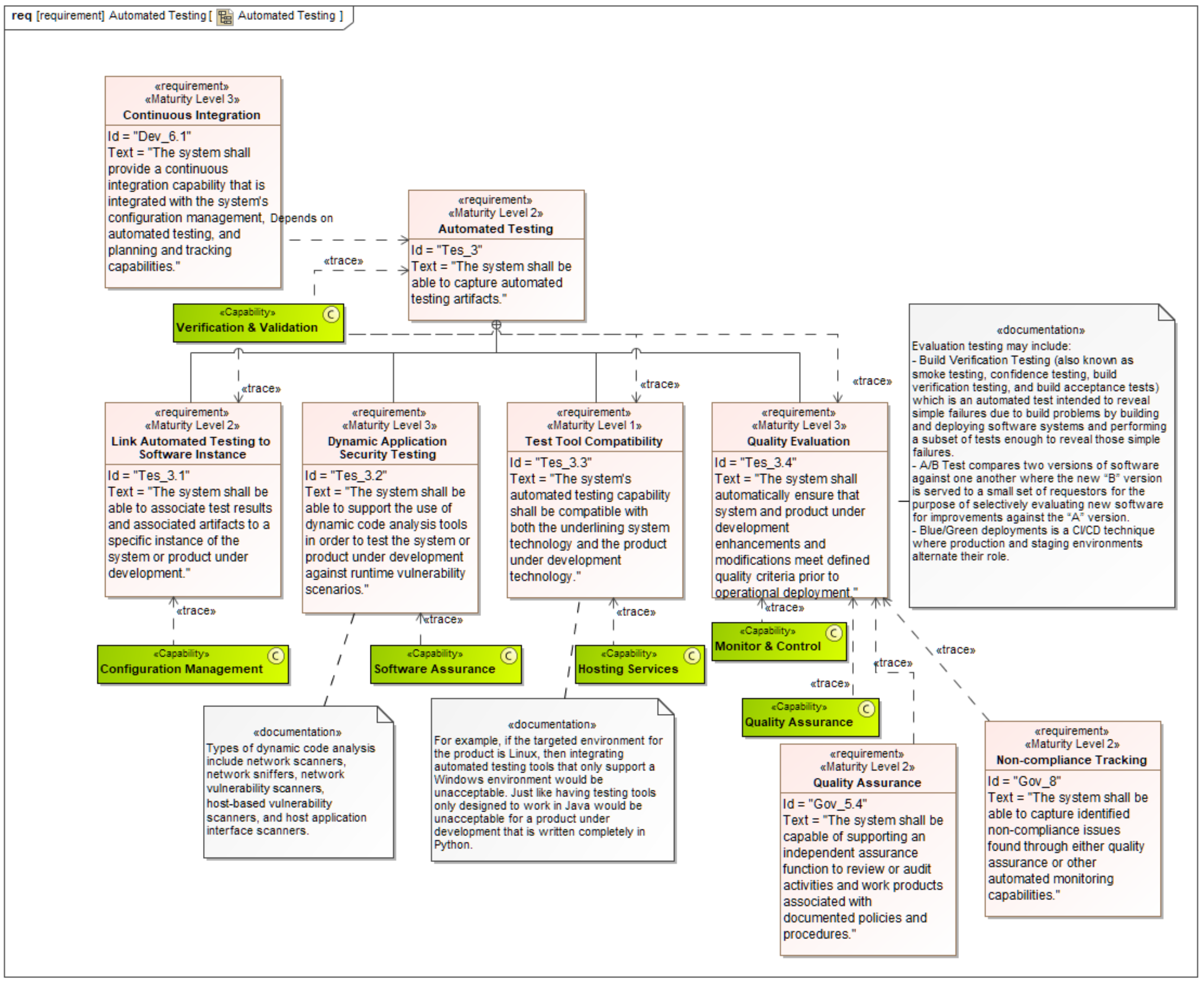

req [requirement] Automated Testing [Automated Testing]
«requirement» «Maturity Level 3» Continuous Integration
Id = "Dev_6.1"
Text = "The system shall provide a continuous integration capability that is integrated with the system's configuration management, automated testing, and planning and tracking capabilities."
Depends on
«trace»
«requirement» «Maturity Level 2» Automated Testing
Id = "Tes_3"
Text = "The system shall be able to capture automated testing artifacts."
«Capability» Verification & Validation
«documentation» Evaluation testing may include:
- Build Verification Testing (also known as smoke testing, confidence testing, build verification testing, and build acceptance tests) which is an automated test intended to reveal simple failures due to build problems by building and deploying software systems and performing a subset of tests enough to reveal those simple failures.
- A/B Test compares two versions of software against one another where the new "B" version is served to a small set of requestors for the purpose of selectively evaluating new software for improvements against the "A" version.
- Blue/Green deployments is a CI/CD technique where production and staging environments alternate their role.
«requirement» «Maturity Level 2» Link Automated Testing to Software Instance
Id = "Tes_3.1"
Text = "The system shall be able to associate test results and associated artifacts to a specific instance of the system or product under development."
«requirement» «Maturity Level 3» Dynamic Application Security Testing
Id = "Tes_3.2"
Text = "The system shall be able to support the use of dynamic code analysis tools in order to test the system or product under development against runtime vulnerability scenarios."
«requirement» «Maturity Level 1» Test Tool Compatibility
Id = "Tes_3.3"
Text = "The system's automated testing capability shall be compatible with both the underlining system technology and the product under development technology."
«requirement» «Maturity Level 3» Quality Evaluation
Id = "Tes_3.4"
Text = "The system shall automatically ensure that system and product under development enhancements and modifications meet defined quality criteria prior to operational deployment."
«Capability» Configuration Management
«Capability» Software Assurance
«Capability» Hosting Services
«Capability» Monitor & Control
«Capability» Quality Assurance
«documentation» Types of dynamic code analysis include network scanners, network sniffers, network vulnerability scanners, host-based vulnerability scanners, and host application interface scanners.
«documentation» For example, if the targeted environment for the product is Linux, then integrating automated testing tools that only support a Windows environment would be unacceptable. Just like having testing tools only designed to work in Java would be unacceptable for a product under development that is written completely in Python.
«requirement» «Maturity Level 2» Quality Assurance
Id = "Gov_5.4"
Text = "The system shall be capable of supporting an independent quality assurance function to review or audit activities and work products associated with documented policies and procedures."
«requirement» «Maturity Level 2» Non-compliance Tracking
Id = "Gov_8"
Text = "The system shall be able to capture identified non-compliance issues found through either quality assurance or other automated monitoring capabilities."

Figure 40: Automated Testing

3.5.3.1 Tes_3.1 Link Automated Testing to Software Instance

The system shall be able to associate test results and associated artifacts to a specific instance of the system or product under development.

3.5.3.2 Tes_3.2 Dynamic Application Security Testing

The system shall be able to support the use of dynamic code analysis tools in order to test the system or product under development against runtime vulnerability scenarios.

Documentation:

Types of dynamic code analysis include network scanners, network sniffers, network vulnerability scanners, host-based vulnerability scanners, and host application interface scanners.

3.5.3.3 Tes_3.3 Test Tool Compatibility

The system's automated testing capability shall be compatible with both the underlining system technology and the product under development technology.

Documentation:

For example, if the targeted environment for the product is Linux, then integrating automated testing tools that only support a Windows environment would be unacceptable. Just like having testing tools only designed to work in Java would be unacceptable for a product under development that is written completely in Python.

3.5.3.4 Tes_3.4 Quality Evaluation

The system shall automatically ensure that system and product under development enhancements and modifications meet defined quality criteria prior to operational deployment.

Documentation:

Evaluation testing may include:

- Build Verification Testing (also known as smoke testing, confidence testing, build verification testing, and build acceptance tests) which is an automated test intended to reveal simple failures due to build problems by building and deploying software systems and performing a subset of tests enough to reveal those simple failures.

- A/B Test compares two versions of software against one another where the new “B” version is served to a small set of requestors for the purpose of selectively evaluating new software for improvements against the “A” version.

- Blue/Green deployments is a CI/CD technique where production and staging environments alternate their role.

3.5.4 Tes_4 Code Coverage

The system shall be able to provide code coverage data associated with each test performed against a specific instance or version of the system or product under development.

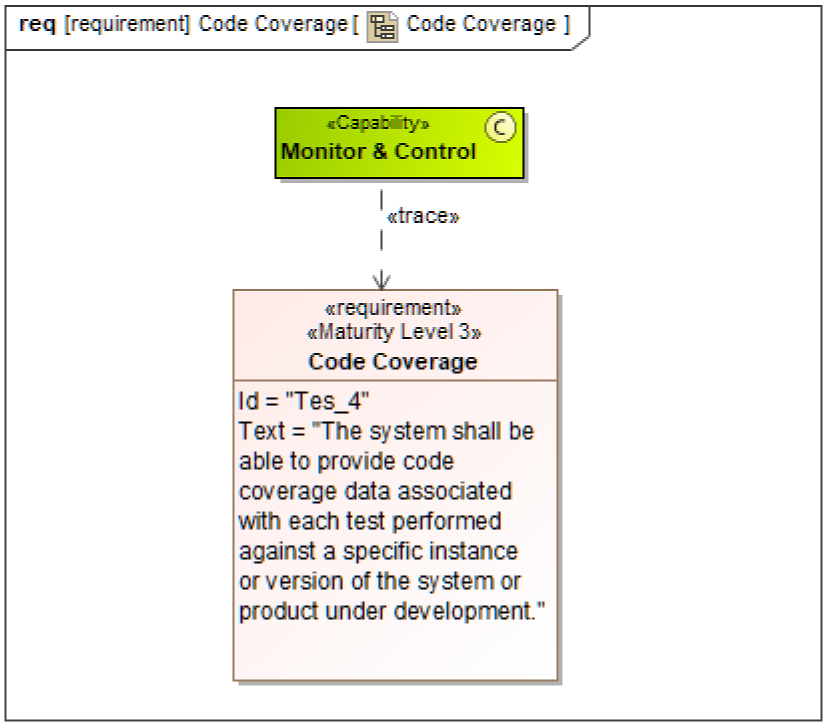

req [requirement] Code Coverage [Code Coverage]
«Capability»
Monitor & Control
«trace»
«requirement»
«Maturity Level 3»
Code Coverage
Id = "Tes_4"
Text = "The system shall be able to provide code coverage data associated with each test performed against a specific instance or version of the system or product under development."

Figure 41: Code Coverage

3.5.5 Tes_5 Penetration and Fuzz Testing

The system shall be able to provide a controlled environment in which penetration and fuzz testing can be performed against both the system and the product under development without adversely impacting the operational or development environments.

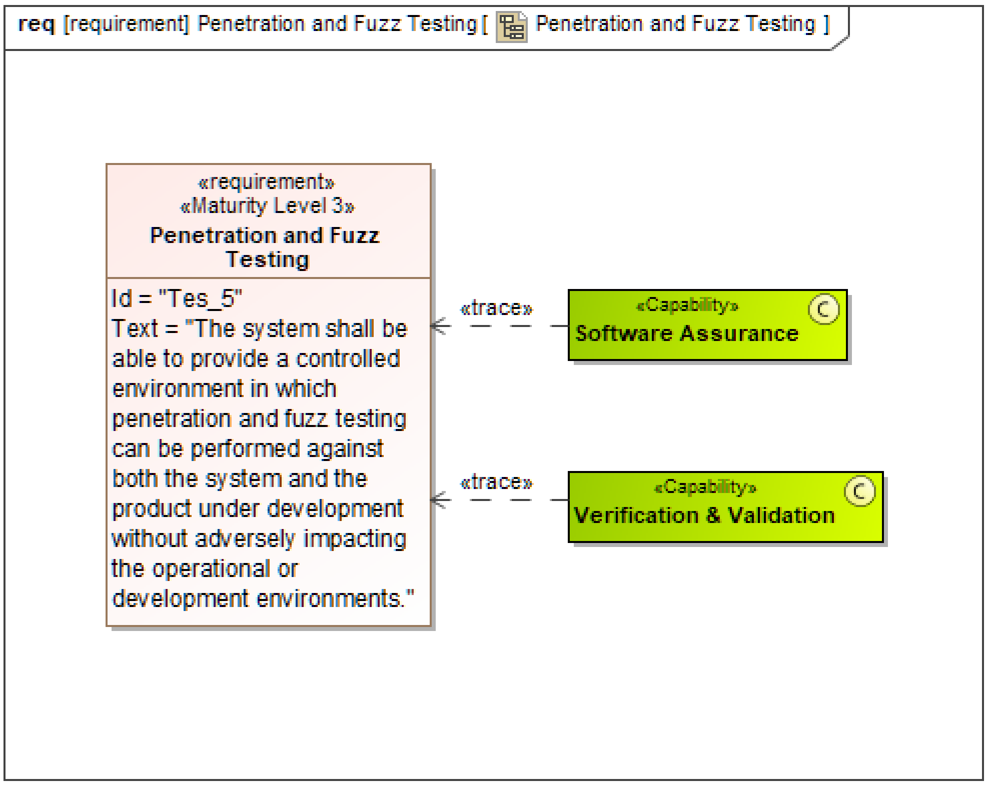

req [requirement] Penetration and Fuzz Testing [Penetration and Fuzz Testing]
«requirement»
«Maturity Level 3»
Penetration and Fuzz Testing
Id = "Tes_5"
Text = "The system shall be able to provide a controlled environment in which penetration and fuzz testing can be performed against both the system and the product under development without adversely impacting the operational or development environments."
«trace»
«Capability»
Software Assurance
«trace»
«Capability»
Verification & Validation

Figure 42: Penetration and Fuzz Testing

3.5.6 Tes_6 Testing Information Radiator

The system shall support the near real-time reporting of test results to all relevant stakeholders.

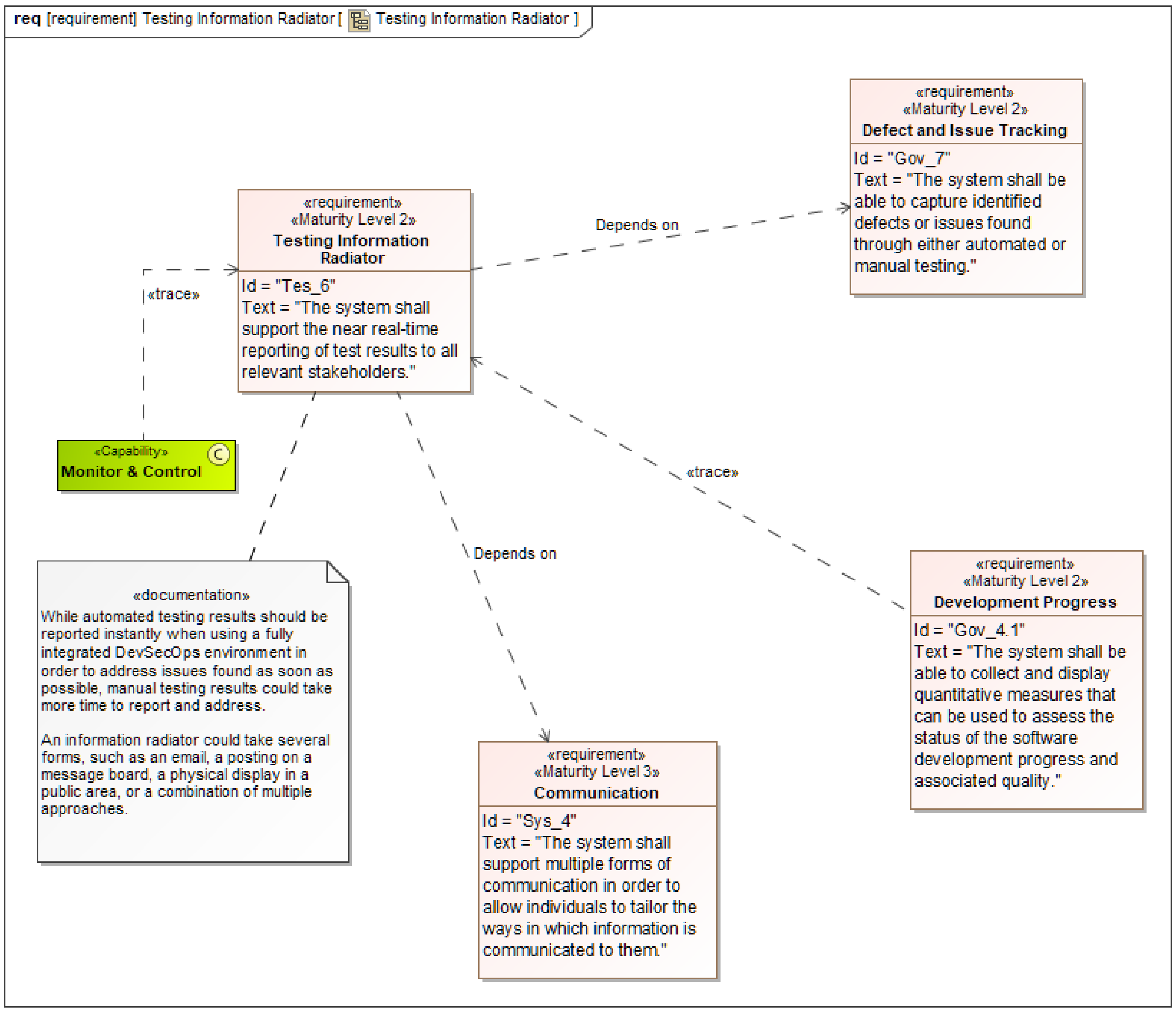

req [requirement] Testing Information Radiator [Testing Information Radiator]
«requirement» «Maturity Level 2» Defect and Issue Tracking
Id = "Gov_7"
Text = "The system shall be able to capture identified defects or issues found through either automated or manual testing."
«requirement» «Maturity Level 2» Testing Information Radiator
Id = "Tes_6"
Text = "The system shall support the near real-time reporting of test results to all relevant stakeholders."
Depends on
«trace»
«Capability» Monitor & Control
«trace»
Depends on
«documentation»
While automated testing results should be reported instantly when using a fully integrated DevSecOps environment in order to address issues found as soon as possible, manual testing results could take more time to report and address.
An information radiator could take several forms, such as an email, a posting on a message board, a physical display in a public area, or a combination of multiple approaches.
«requirement» «Maturity Level 2» Development Progress
Id = "Gov_4.1"
Text = "The system shall be able to collect and display quantitative measures that can be used to assess the status of the software development progress and associated quality."
«requirement» «Maturity Level 3» Communication
Id = "Sys_4"
Text = "The system shall support multiple forms of communication in order to allow individuals to tailor the ways in which information is communicated to them."

Figure 43: Testing Information Radiator

Documentation:

While automated testing results should be reported instantly when using a fully integrated DevSecOps environment in order to address issues found as soon as possible, manual testing results could take more time to report and address.

An information radiator could take several forms, such as an email, a posting on a message board, a physical display in a public area, or a combination of multiple approaches.

3.5.7 Tes_7 Multi-phase Testing

The system shall support both manual and automated testing throughout the development, maintenance and operational life cycles.

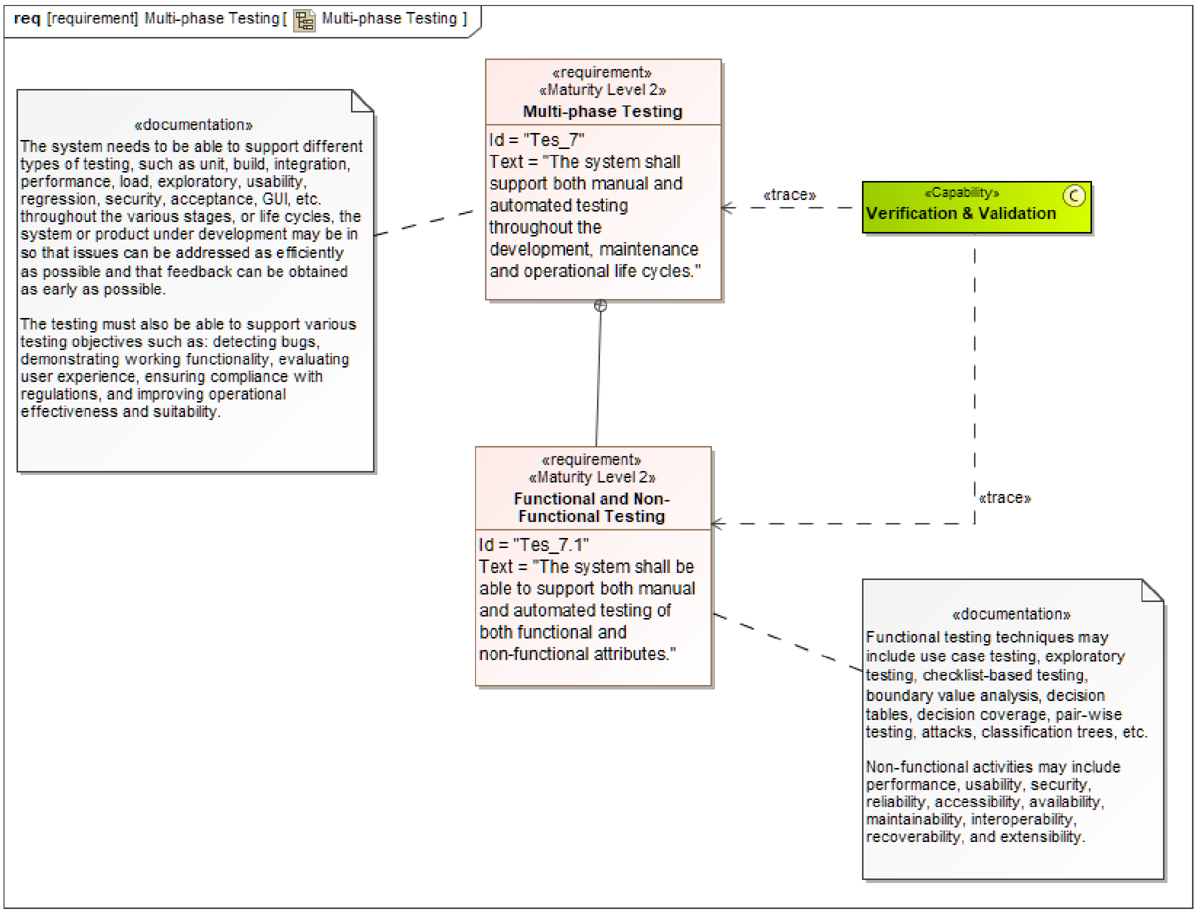

req [requirement] Multi-phase Testing [Multi-phase Testing]
«documentation»
The system needs to be able to support different types of testing, such as unit, build, integration, performance, load, exploratory, usability, regression, security, acceptance, GUI, etc. throughout the various stages, or life cycles, the system or product under development may be in so that issues can be addressed as efficiently as possible and that feedback can be obtained as early as possible.
The testing must also be able to support various testing objectives such as: detecting bugs, demonstrating working functionality, evaluating user experience, ensuring compliance with regulations, and improving operational effectiveness and suitability.
«requirement» «Maturity Level 2» Multi-phase Testing
Id = "Tes_7"
Text = "The system shall support both manual and automated testing throughout the development, maintenance and operational life cycles."
«trace»
«Capability» Verification & Validation
«requirement» «Maturity Level 2» Functional and Non-Functional Testing
Id = "Tes_7.1"
Text = "The system shall be able to support both manual and automated testing of both functional and non-functional attributes."
«trace»
«documentation»
Functional testing techniques may include use case testing, exploratory testing, checklist-based testing, boundary value analysis, decision tables, decision coverage, pair-wise testing, attacks, classification trees, etc.
Non-functional activities may include performance, usability, security, reliability, accessibility, availability, maintainability, interoperability, recoverability, and extensibility.

Figure 44: Multi-phase Testing

Documentation:

The system needs to be able to support different types of testing, such as unit, build, integration, performance, load, exploratory, usability, regression, security, acceptance, GUI, etc. throughout the various stages, or life cycles, the system or product under development may be in so that issues can be addressed as efficiently as possible and that feedback can be obtained as early as possible.

The testing must also be able to support various testing objectives such as: detecting bugs, demonstrating working functionality, evaluating user experience, ensuring compliance with regulations, and improving operational effectiveness and suitability.

3.5.7.1 Tes_7.1 Functional and Non-Functional Testing

The system shall be able to support both manual and automated testing of both functional and non-functional attributes.

Documentation:

Functional testing techniques may include use case testing, exploratory testing, checklist-based testing, boundary value analysis, decision tables, decision coverage, pair-wise testing, attacks, classification trees, etc.

Non-functional activities may include performance, usability, security, reliability, accessibility, availability, maintainability, interoperability, recoverability, and extensibility.

3.6 6 Delivery

3.6.1 Del_1 Release Management

The system shall support the automation of releases.

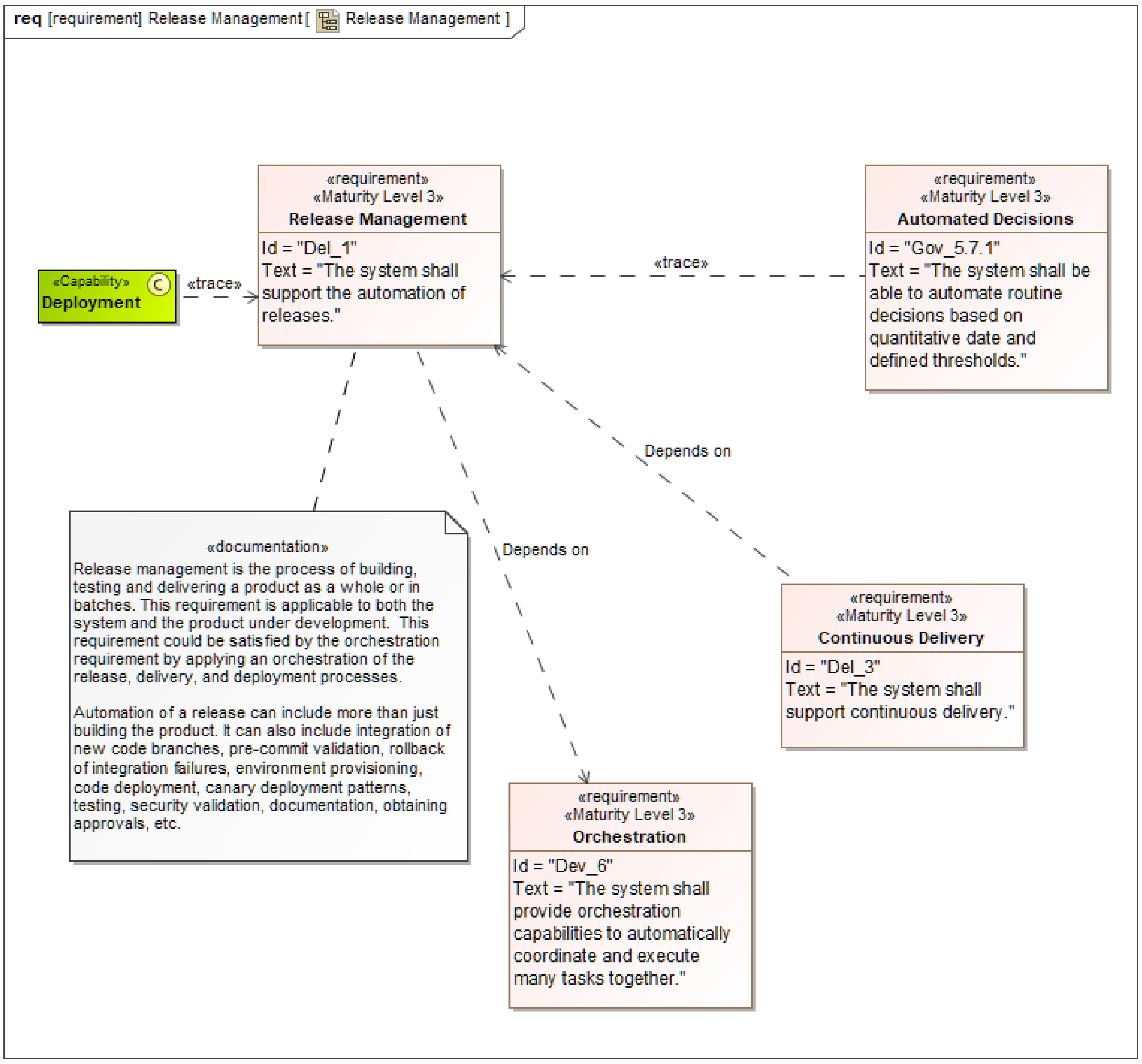

req [requirement] Release Management [Release Management]
«Capability» Deployment
«trace»
«requirement» «Maturity Level 3» Release Management
Id = "Del_1"
Text = "The system shall support the automation of releases."
«trace»
«requirement» «Maturity Level 3» Automated Decisions
Id = "Gov_5.7.1"
Text = "The system shall be able to automate routine decisions based on quantitative date and defined thresholds."
Depends on
Depends on
«requirement» «Maturity Level 3» Continuous Delivery
Id = "Del_3"
Text = "The system shall support continuous delivery."
«requirement» «Maturity Level 3» Orchestration
Id = "Dev_6"
Text = "The system shall provide orchestration capabilities to automatically coordinate and execute many tasks together."
«documentation»
Release management is the process of building, testing and delivering a product as a whole or in batches. This requirement is applicable to both the system and the product under development. This requirement could be satisfied by the orchestration requirement by applying an orchestration of the release, delivery, and deployment processes.
Automation of a release can include more than just building the product. It can also include integration of new code branches, pre-commit validation, rollback of integration failures, environment provisioning, code deployment, canary deployment patterns, testing, security validation, documentation, obtaining approvals, etc.

Figure 45: Release Management

Documentation:

Release management is the process of building, testing and delivering a product as a whole or in batches. This requirement is applicable to both the system and the product under development. This requirement could be satisfied by the orchestration requirement by applying an orchestration of the release, delivery, and deployment processes.

Automation of a release can include more than just building the product. It can also include integration of new code branches, pre-commit validation, rollback of integration failures, environment provisioning, code deployment, canary deployment patterns, testing, security validation, documentation, obtaining approvals, etc.

3.6.2 Del_2 ITSM Service Desk

The system shall provide an integrated ITSM service desk capability to manage the end-to-end delivery of services to users.

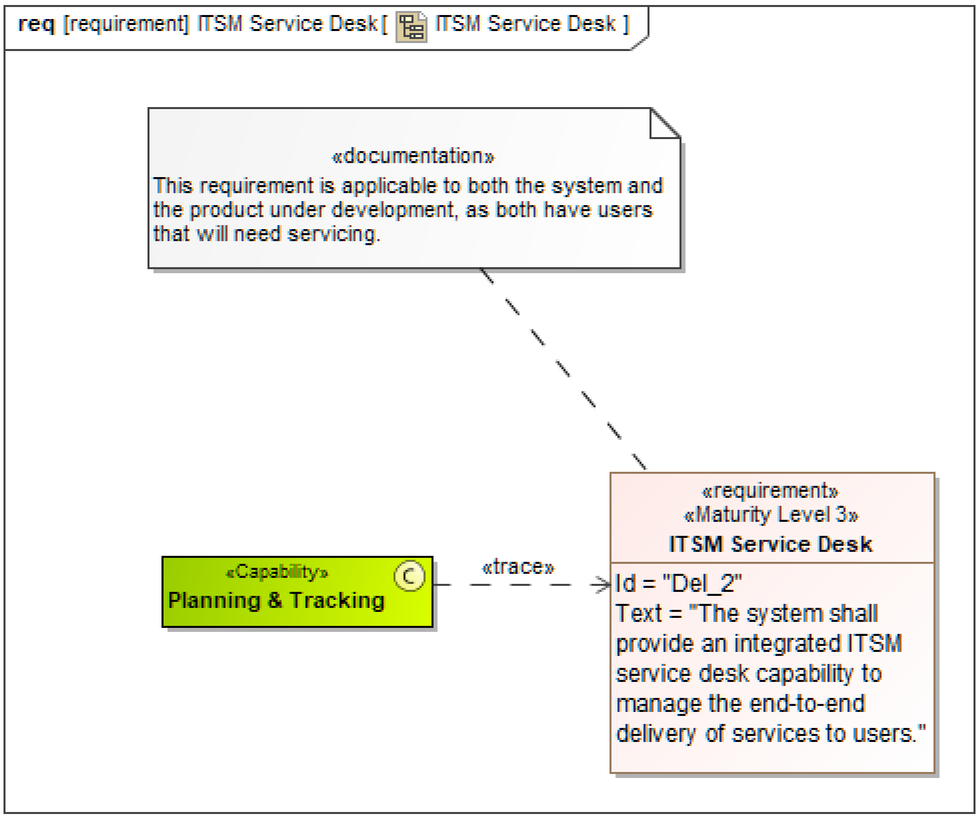

req [requirement] ITSM Service Desk [ITSM Service Desk]
«documentation»
This requirement is applicable to both the system and the product under development, as both have users that will need servicing.
«Capability»
Planning & Tracking
«trace»
«requirement»
«Maturity Level 3»
ITSM Service Desk
Id = "Del_2"
Text = "The system shall provide an integrated ITSM service desk capability to manage the end-to-end delivery of services to users."

Figure 46: ITSM Service Desk

Documentation:

This requirement is applicable to both the system and the product under development, as both have users that will need servicing.

3.6.3 Del_3 Continuous Delivery

The system shall support continuous delivery.

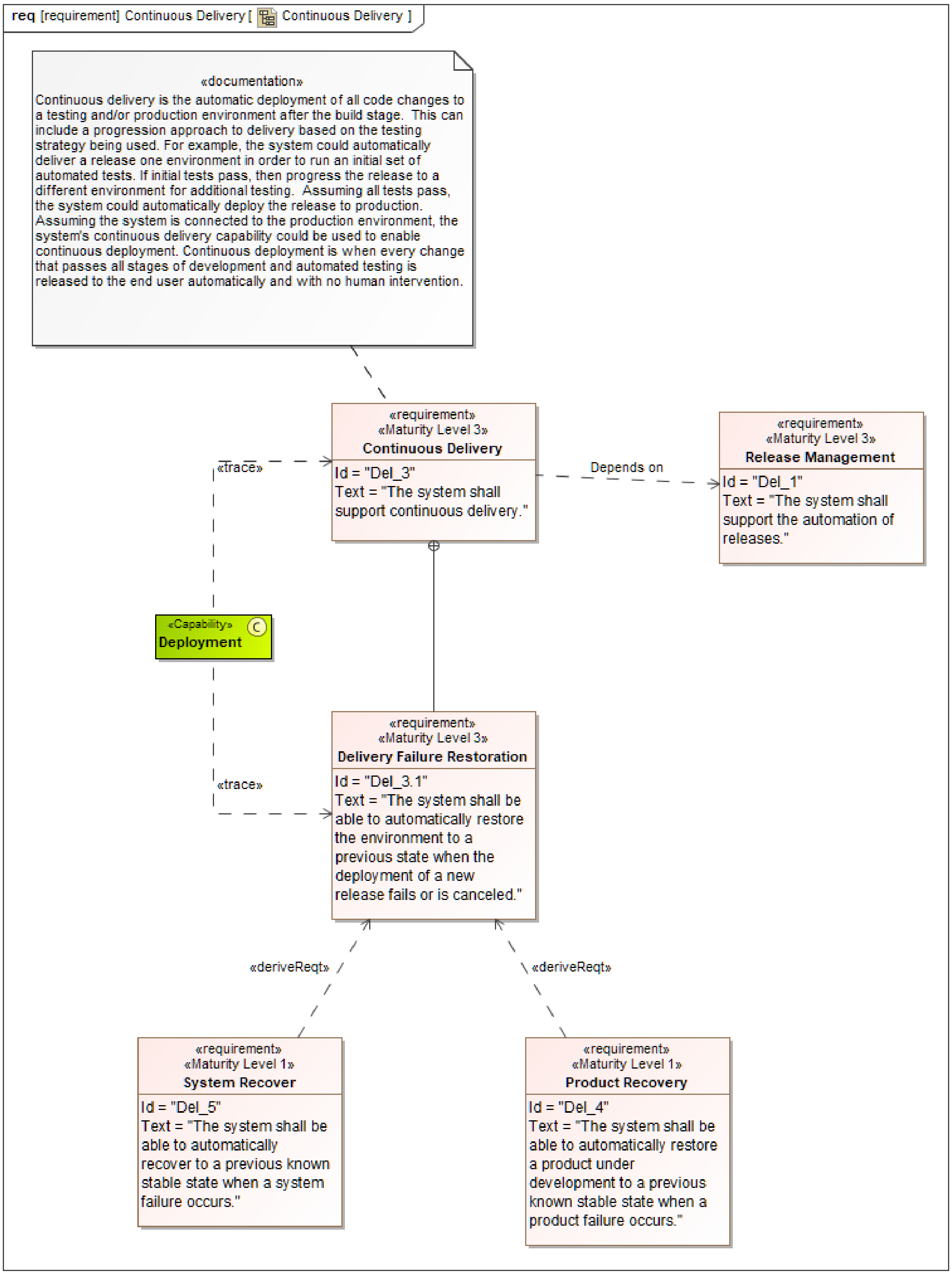

req [requirement] Continuous Delivery [Continuous Delivery]
«documentation»
Continuous delivery is the automatic deployment of all code changes to a testing and/or production environment after the build stage. This can include a progression approach to delivery based on the testing strategy being used. For example, the system could automatically deliver a release one environment in order to run an initial set of automated tests. If initial tests pass, then progress the release to a different environment for additional testing. Assuming all tests pass, the system could automatically deploy the release to production. Assuming the system is connected to the production environment, the system's continuous delivery capability could be used to enable continuous deployment. Continuous deployment is when every change that passes all stages of development and automated testing is released to the end user automatically and with no human intervention.
«requirement» «Maturity Level 3» Continuous Delivery
Id = "Del_3"
Text = "The system shall support continuous delivery."
Depends on
«requirement» «Maturity Level 3» Release Management
Id = "Del_1"
Text = "The system shall support the automation of releases."
«trace»
«Capability» Deployment
«trace»
«requirement» «Maturity Level 3» Delivery Failure Restoration
Id = "Del_3.1"
Text = "The system shall be able to automatically restore the environment to a previous state when the deployment of a new release fails or is canceled."
«deriveReqt»
«deriveReqt»
«requirement» «Maturity Level 1» System Recover
Id = "Del_5"
Text = "The system shall be able to automatically recover to a previous known stable state when a system failure occurs."
«requirement» «Maturity Level 1» Product Recovery
Id = "Del_4"
Text = "The system shall be able to automatically restore a product under development to a previous known stable state when a product failure occurs."

Figure 47: Continuous Delivery

Documentation:

Continuous delivery is the automatic deployment of all code changes to a testing and/or production environment after the build stage. This can include a progression approach to delivery based on the testing strategy being used. For example, the system could automatically deliver a release one environment in order to run an initial set of automated tests. If initial tests pass, then progress the release to a different environment for additional testing. Assuming all tests pass, the system could automatically deploy the release to production. Assuming the system is connected to the production environment, the system's continuous delivery capability could be used to enable continuous deployment. Continuous deployment is when every change that passes all stages of development and automated testing is released to the end user automatically and with no human intervention.

3.6.3.1 Del_3.1 Delivery Failure Restoration

The system shall be able to automatically restore the environment to a previous state when the deployment of a new release fails or is canceled.

3.6.4 Del_4 Product Recovery

The system shall be able to automatically restore a product under development to a previous known stable state when a product failure occurs.

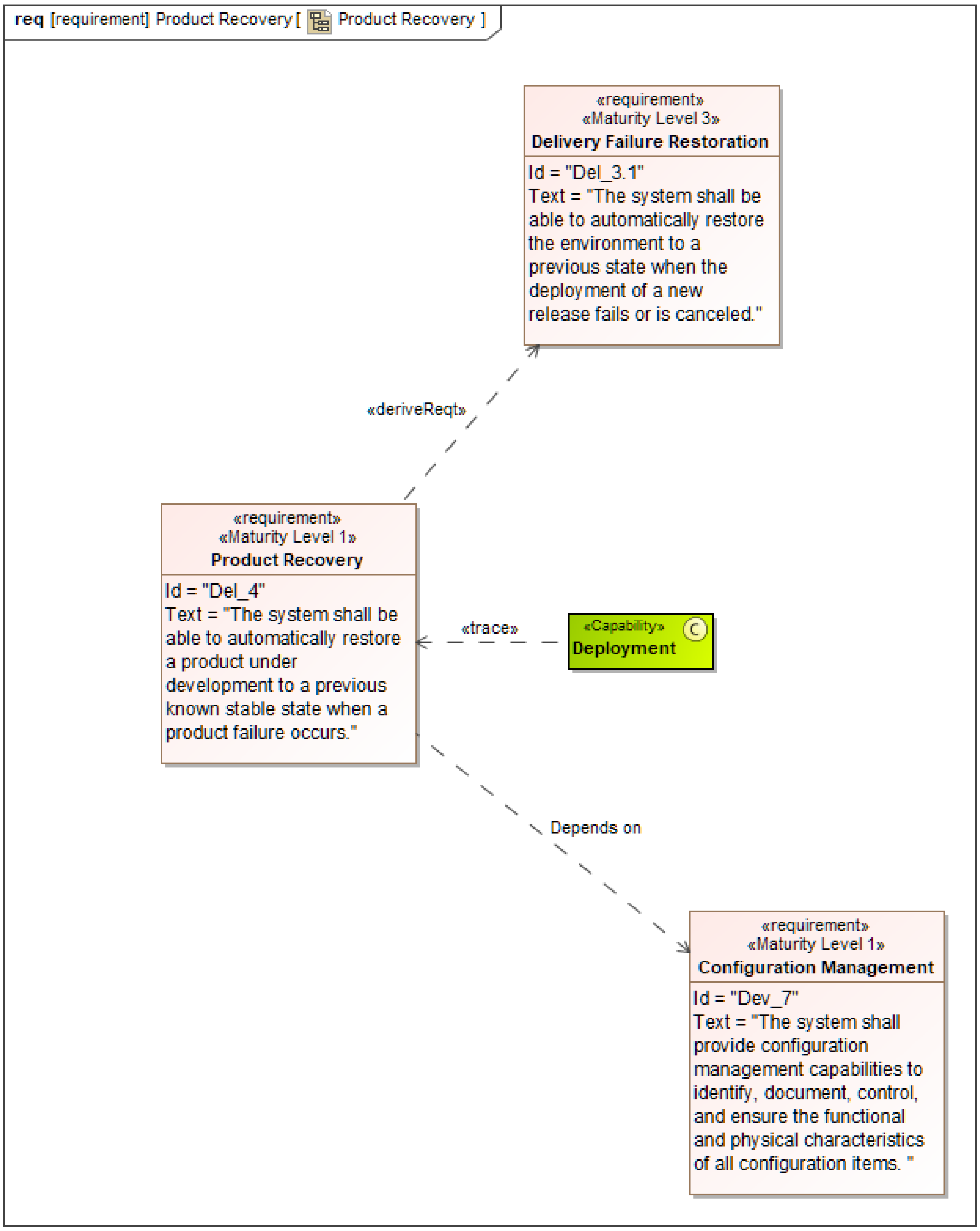

req [requirement] Product Recovery [Product Recovery]
«requirement»
«Maturity Level 3»
Delivery Failure Restoration
Id = "Del_3.1"
Text = "The system shall be able to automatically restore the environment to a previous state when the deployment of a new release fails or is canceled."
«deriveReqt»
«requirement»
«Maturity Level 1»
Product Recovery
Id = "Del_4"
Text = "The system shall be able to automatically restore a product under development to a previous known stable state when a product failure occurs."
«trace»
«Capability»
Deployment
Depends on
«requirement»
«Maturity Level 1»
Configuration Management
Id = "Dev_7"
Text = "The system shall provide configuration management capabilities to identify, document, control, and ensure the functional and physical characteristics of all configuration items. "

Figure 48: Product Recovery

3.6.5 Del_5 System Recover

The system shall be able to automatically recover to a previous known stable state when a system failure occurs.

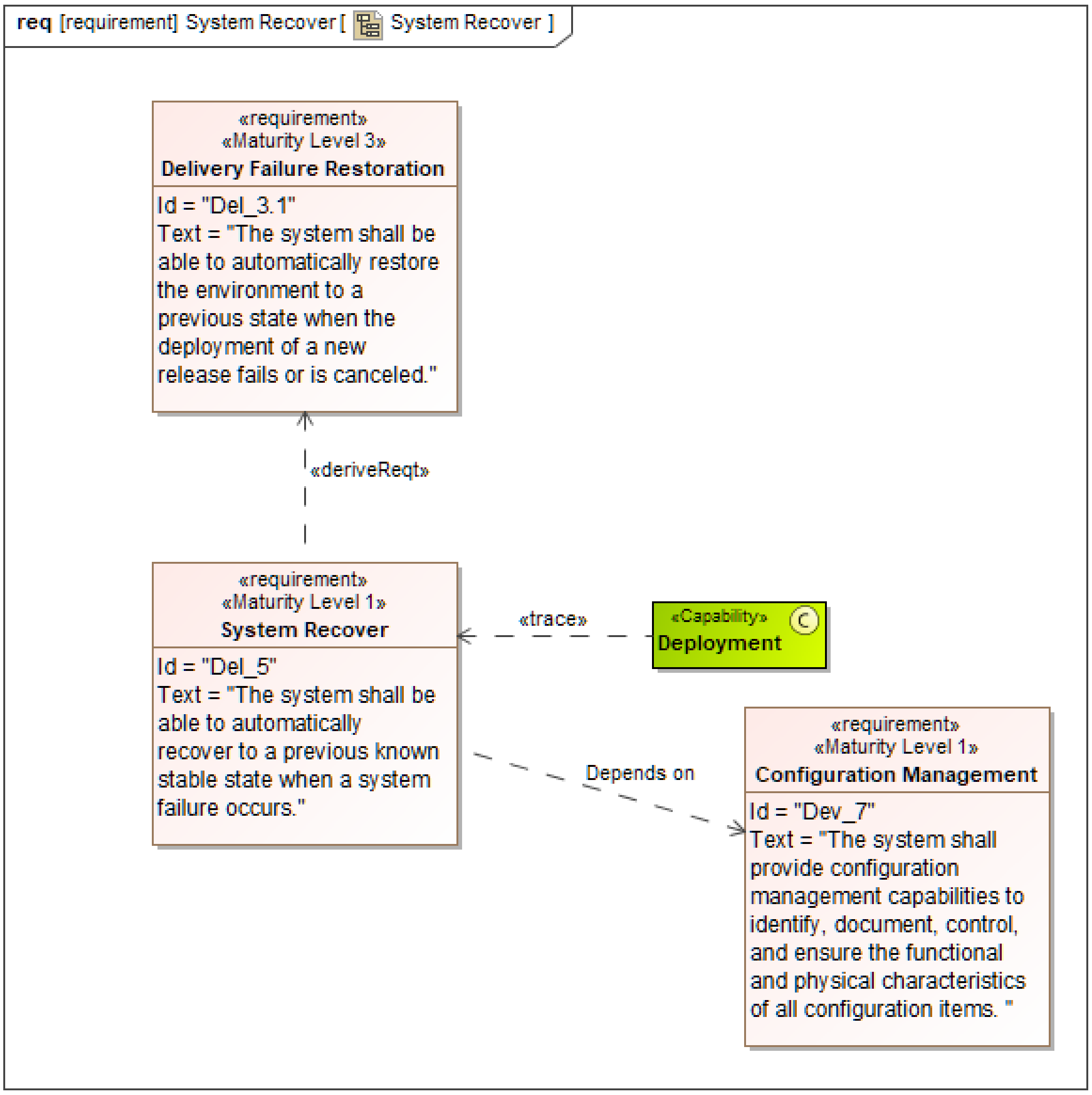

req [requirement] System Recover [System Recover]
«requirement»
«Maturity Level 3»
Delivery Failure Restoration
Id = "Del_3.1"
Text = "The system shall be able to automatically restore the environment to a previous state when the deployment of a new release fails or is canceled."
«deriveReqt»
«requirement»
«Maturity Level 1»
System Recover
Id = "Del_5"
Text = "The system shall be able to automatically recover to a previous known stable state when a system failure occurs."
«trace»
«Capability»
Deployment
Depends on
«requirement»
«Maturity Level 1»
Configuration Management
Id = "Dev_7"
Text = "The system shall provide configuration management capabilities to identify, document, control, and ensure the functional and physical characteristics of all configuration items. "

Figure 49: System Recover

3.6.6 Del_6 Configuration Item Integrity

The system shall ensure the integrity of all configuration items in transit and identify any unauthorized changes to a deployed configuration item.

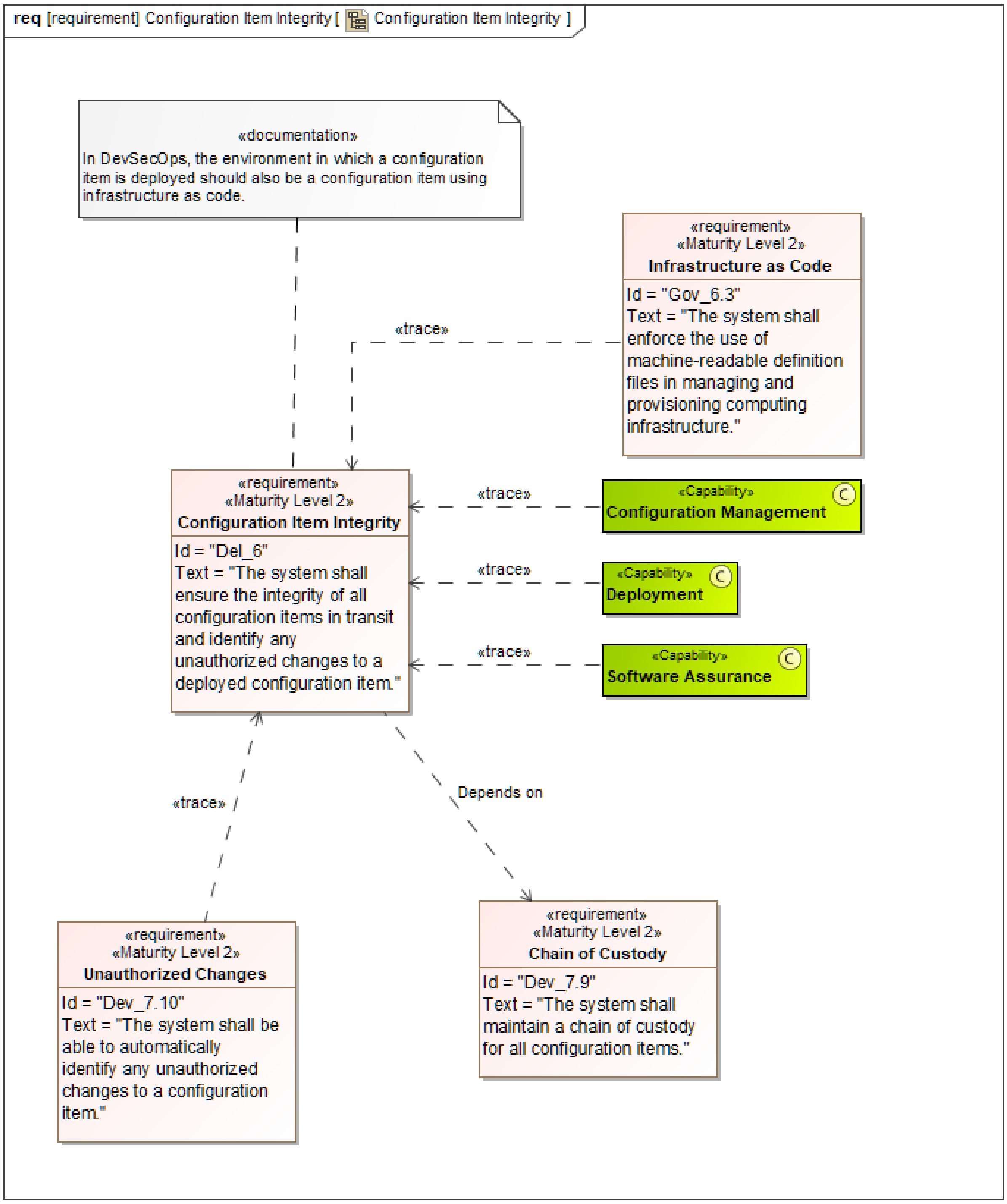

req [requirement] Configuration Item Integrity [Configuration Item Integrity]
«documentation»
In DevSecOps, the environment in which a configuration item is deployed should also be a configuration item using infrastructure as code.
«requirement» «Maturity Level 2» Infrastructure as Code
Id = "Gov_6.3"
Text = "The system shall enforce the use of machine-readable definition files in managing and provisioning computing infrastructure."
«trace»
«requirement» «Maturity Level 2» Configuration Item Integrity
Id = "Del_6"
Text = "The system shall ensure the integrity of all configuration items in transit and identify any unauthorized changes to a deployed configuration item."
«trace»
«Capability» Configuration Management
«trace»
«Capability» Deployment
«trace»
«Capability» Software Assurance
«trace»
Depends on
«requirement» «Maturity Level 2» Unauthorized Changes
Id = "Dev_7.10"
Text = "The system shall be able to automatically identify any unauthorized changes to a configuration item."
«requirement» «Maturity Level 2» Chain of Custody
Id = "Dev_7.9"
Text = "The system shall maintain a chain of custody for all configuration items."

Figure 50: Configuration Item Integrity

Documentation:

In DevSecOps, the environment in which a configuration item is deployed should also be a configuration item using infrastructure as code.

3.7 7 System Infrastructure

3.7.1 Sys_1 System's Non-functional Requirements

The system's infrastructure shall be designed and implemented to meet the organizations or specific product under development's reliability, security, dependability, scalability, and regulatory needs.

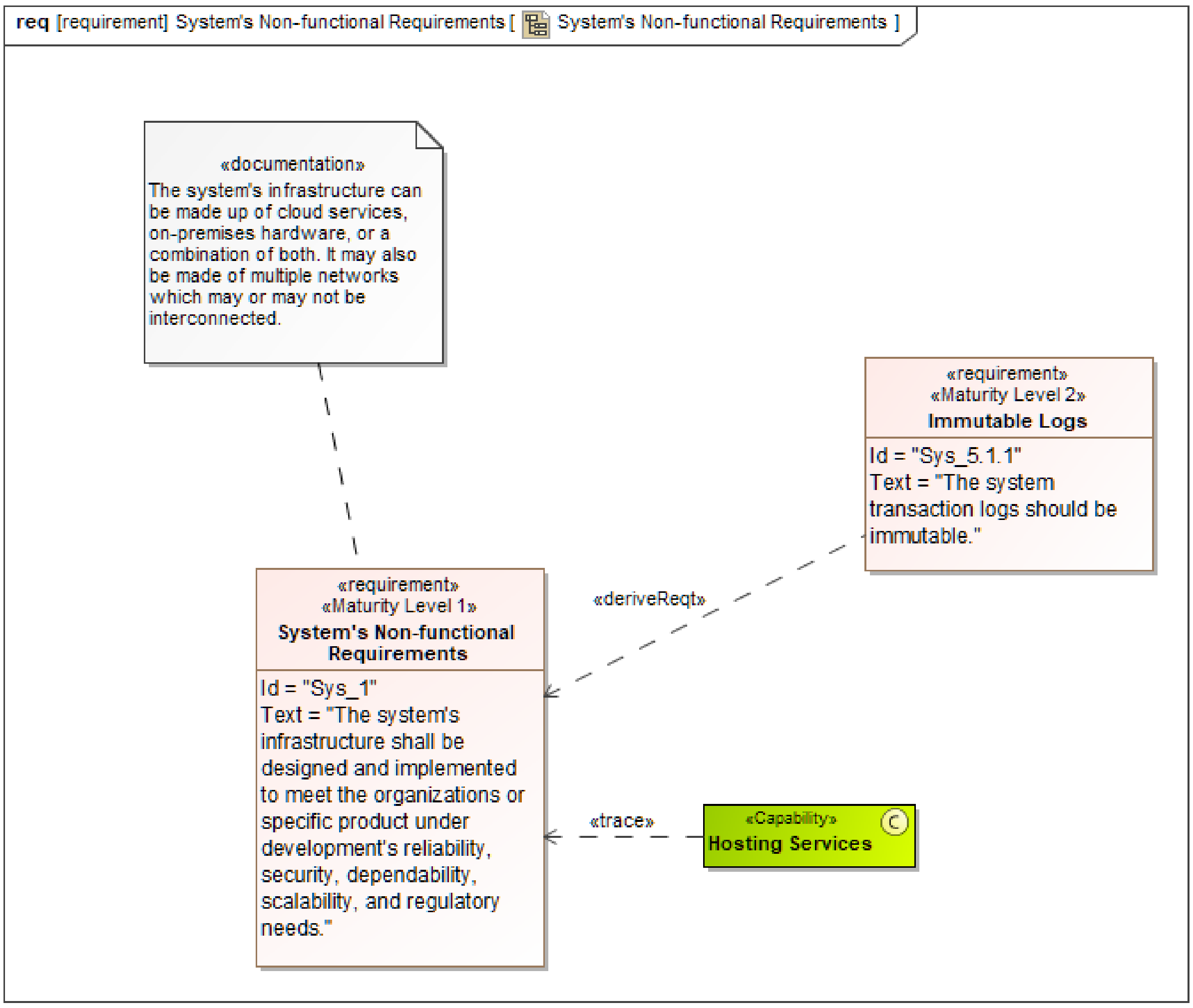

req [requirement] System's Non-functional Requirements [System's Non-functional Requirements]
«documentation»
The system's infrastructure can be made up of cloud services, on-premises hardware, or a combination of both. It may also be made of multiple networks which may or may not be interconnected.
«requirement» «Maturity Level 2» Immutable Logs
Id = "Sys_5.1.1"
Text = "The system transaction logs should be immutable."
«deriveReqt»
«requirement» «Maturity Level 1» System's Non-functional Requirements
Id = "Sys_1"
Text = "The system's infrastructure shall be designed and implemented to meet the organizations or specific product under development's reliability, security, dependability, scalability, and regulatory needs."
«trace»
«Capability» Hosting Services

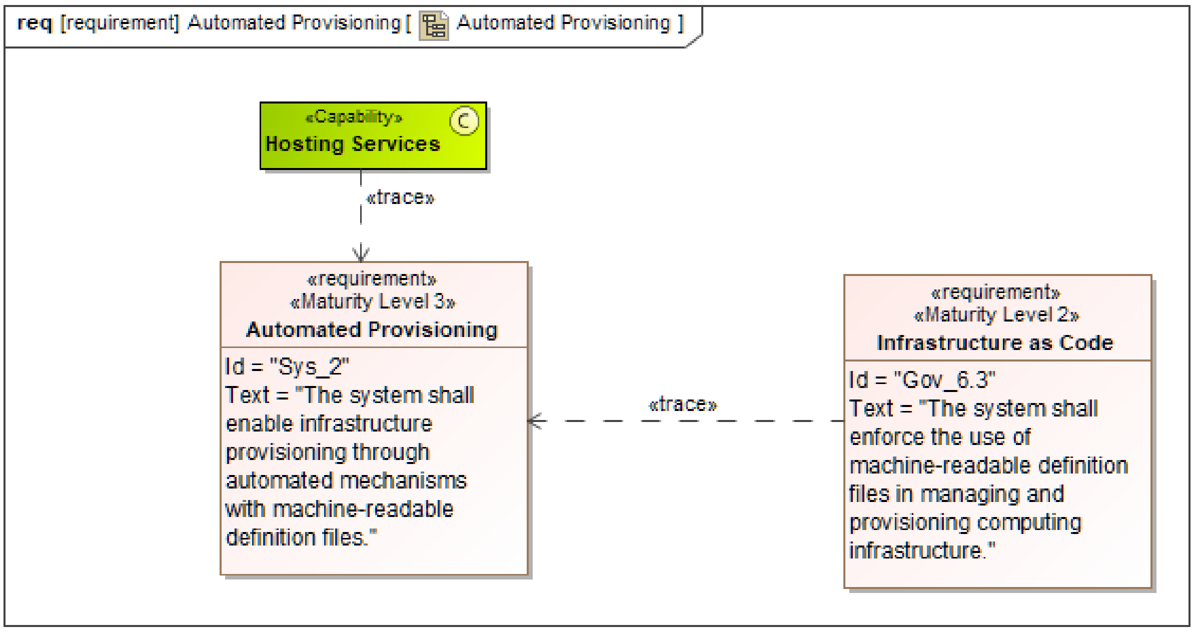
Figure 51: System's Non-functional Requirements

Documentation:

The system's infrastructure can be made up of cloud services, on-premises hardware, or a combination of both. It may also be made of multiple networks which may or may not be interconnected.

3.7.2 Sys_2 Automated Provisioning

The system shall enable infrastructure provisioning through automated mechanisms with machine-readable definition files.


req [requirement] Automated Provisioning [Automated Provisioning]
«Capability»
Hosting Services
«trace»
«trace»

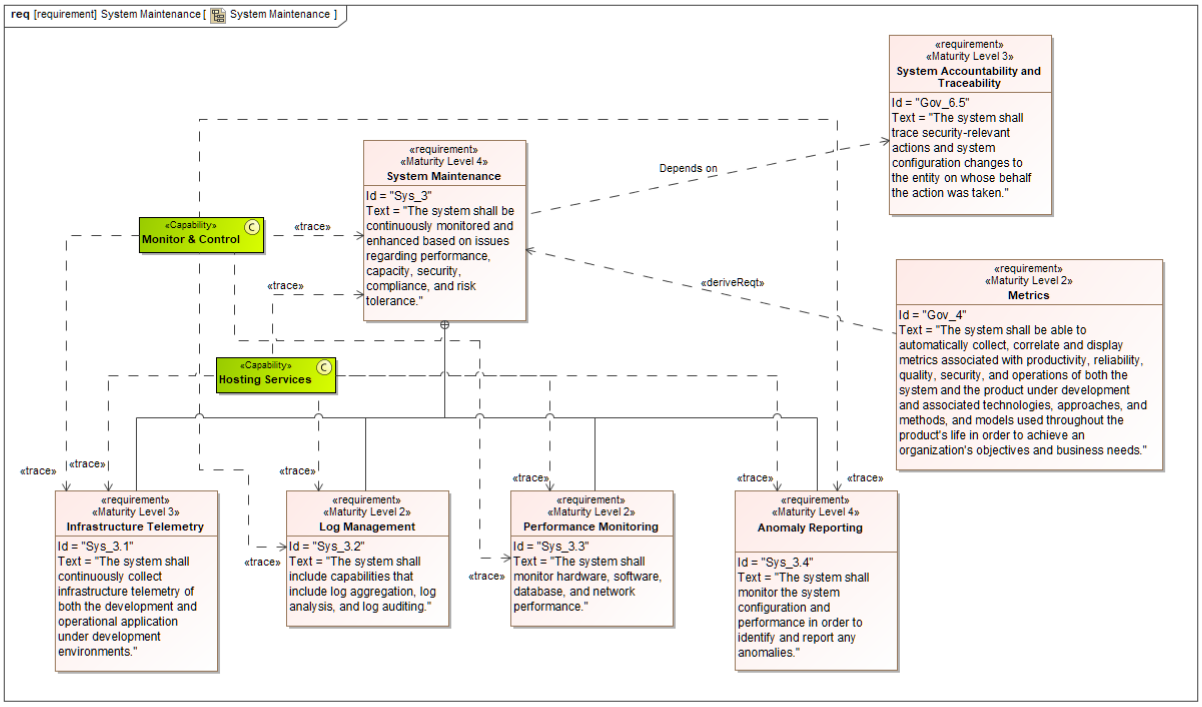
«requirement»
«Maturity Level 3»
Automated Provisioning
Id = "Sys_2"
Text = "The system shall enable infrastructure provisioning through automated mechanisms with machine-readable definition files."

«requirement»
«Maturity Level 2»
Infrastructure as Code
Id = "Gov_6.3"
Text = "The system shall enforce the use of machine-readable definition files in managing and provisioning computing infrastructure."

Figure 52: Automated Provisioning

3.7.3 Sys_3 System Maintenance

The system shall be continuously monitored and enhanced based on issues regarding performance, capacity, security, compliance, and risk tolerance.



«requirement»
«Maturity Level 4»
System Maintenance
Id = "Sys_3"
Text = "The system shall be continuously monitored and enhanced based on issues regarding performance, capacity, security, compliance, and risk tolerance."

«requirement»
«Maturity Level 3»
System Accountability and Traceability
Id = "Gov_6.5"
Text = "The system shall trace security-relevant actions and system configuration changes to the entity on whose behalf the action was taken."

«requirement»
«Maturity Level 2»
Metrics
Id = "Gov_4"
Text = "The system shall be able to automatically collect, correlate and display metrics associated with productivity, reliability, quality, security, and operations of both the system and the product under development and associated technologies, approaches, and methods, and models used throughout the product's life in order to achieve an organization's objectives and business needs."

«requirement»
«Maturity Level 3»
Infrastructure Telemetry
Id = "Sys_3.1"
Text = "The system shall continuously collect infrastructure telemetry of both the development and operational application under development environments."

«requirement»
«Maturity Level 2»
Log Management
Id = "Sys_3.2"
Text = "The system shall include capabilities that include log aggregation, log analysis, and log auditing."

«requirement»
«Maturity Level 2»
Performance Monitoring
Id = "Sys_3.3"
Text = "The system shall monitor hardware, software, database, and network performance."

«requirement»
«Maturity Level 4»
Anomaly Reporting
Id = "Sys_3.4"
Text = "The system shall monitor the system configuration and performance in order to identify and report any anomalies."

Figure 53: System Maintenance

3.7.3.1 Sys_3.1 Infrastructure Telemetry

The system shall continuously collect infrastructure telemetry of both the development and operational application under development environments.

3.7.3.2 Sys_3.2 Log Management

The system shall include capabilities that include log aggregation, log analysis, and log auditing.

3.7.3.3 Sys_3.3 Performance Monitoring

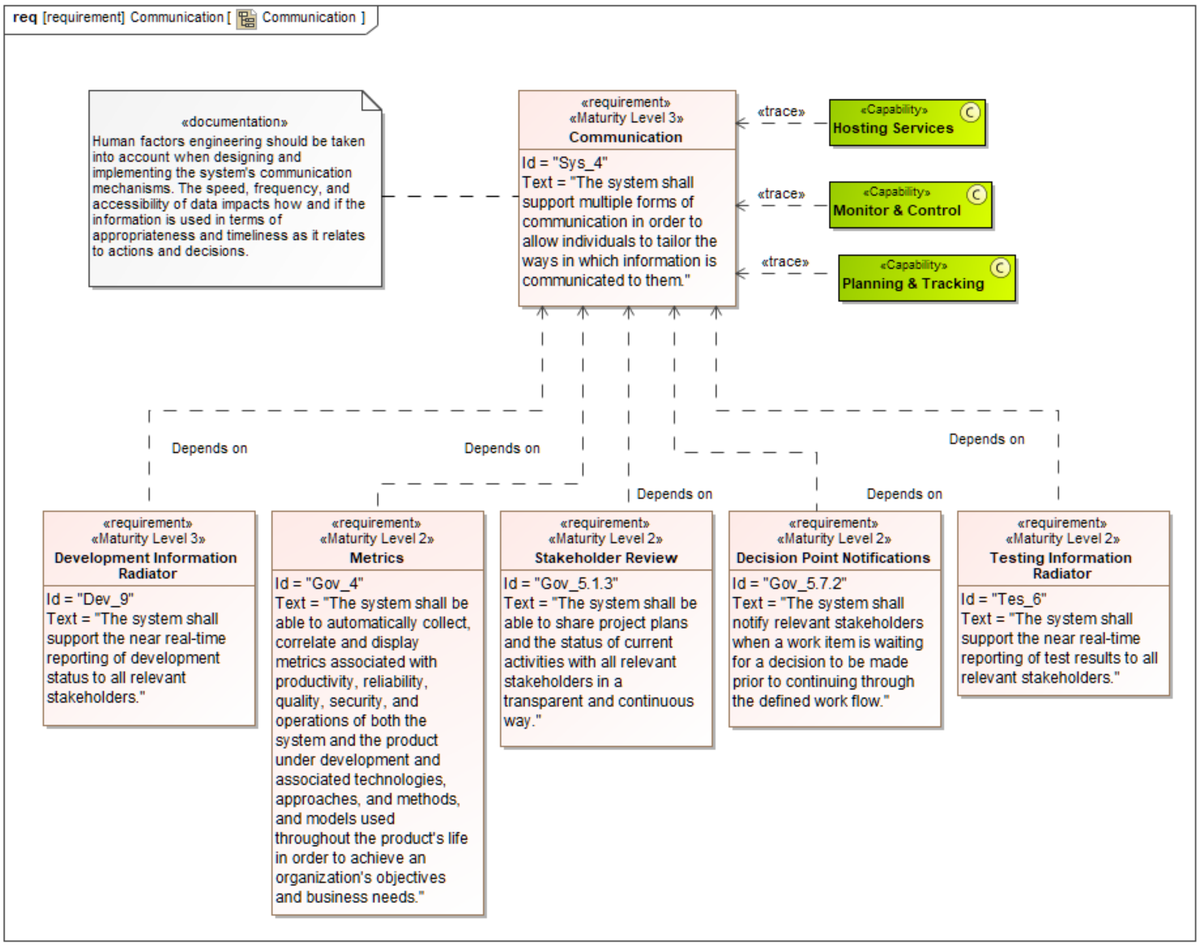
The system shall monitor hardware, software, database, and network performance.

3.7.3.4 Sys_3.4 Anomaly Reporting

The system shall monitor the system configuration and performance in order to identify and report any anomalies.

3.7.4 Sys_4 Communication

The system shall support multiple forms of communication in order to allow individuals to tailor the ways in which information is communicated to them.


req [requirement] Communication [Communication]
«trace»
«trace»
«trace»
«Capability» Hosting Services
«Capability» Monitor & Control
«Capability» Planning & Tracking
Depends on
Depends on
Depends on
Depends on
Depends on

«documentation»
Human factors engineering should be taken into account when designing and implementing the system's communication mechanisms. The speed, frequency, and accessibility of data impacts how and if the information is used in terms of appropriateness and timeliness as it relates to actions and decisions.

«requirement» «Maturity Level 3» **Communication**
Id = "Sys_4"
Text = "The system shall support multiple forms of communication in order to allow individuals to tailor the ways in which information is communicated to them."

«requirement» «Maturity Level 3» **Development Information Radiator**
Id = "Dev_9"
Text = "The system shall support the near real-time reporting of development status to all relevant stakeholders."

«requirement» «Maturity Level 2» **Metrics**
Id = "Gov_4"
Text = "The system shall be able to automatically collect, correlate and display metrics associated with productivity, reliability, quality, security, and operations of both the system and the product under development and associated technologies, approaches, and methods, and models used throughout the product's life in order to achieve an organization's objectives and business needs."

«requirement» «Maturity Level 2» **Stakeholder Review**
Id = "Gov_5.1.3"
Text = "The system shall be able to share project plans and the status of current activities with all relevant stakeholders in a transparent and continuous way."

«requirement» «Maturity Level 2» **Decision Point Notifications**
Id = "Gov_5.7.2"
Text = "The system shall notify relevant stakeholders when a work item is waiting for a decision to be made prior to continuing through the defined work flow."

«requirement» «Maturity Level 2» **Testing Information Radiator**
Id = "Tes_6"
Text = "The system shall support the near real-time reporting of test results to all relevant stakeholders."

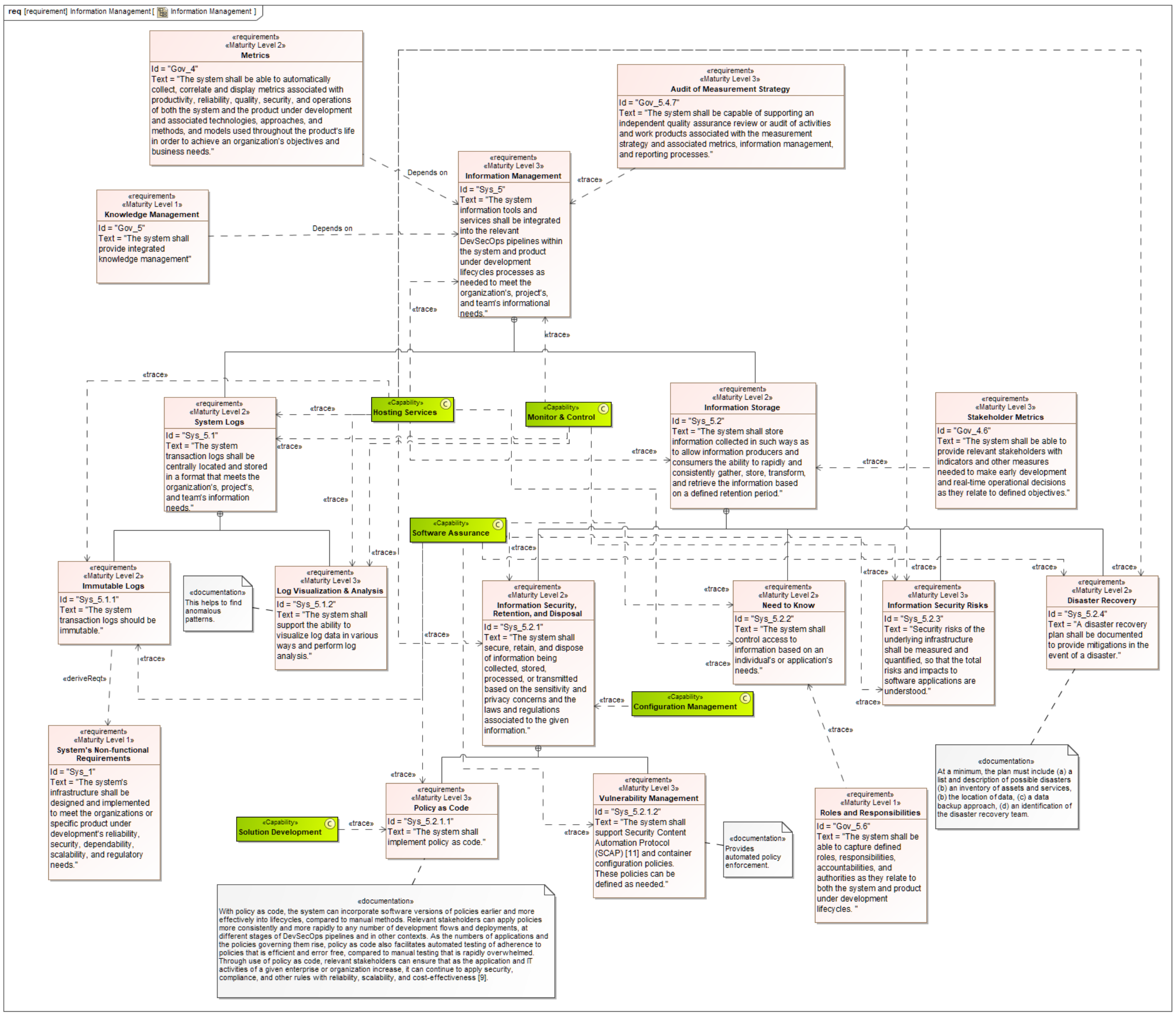
Figure 54: Communication

Documentation:

Human factors engineering should be taken into account when designing and implementing the system's communication mechanisms. The speed, frequency, and accessibility of data impacts how and if the information is used in terms of appropriateness and timeliness as it relates to actions and decisions.

3.7.5 Sys_5 Information Management

The system information tools and services shall be integrated into the relevant DevSecOps pipelines within the system and product under development lifecycles processes as needed to meet the organization's, project's, and team's informational needs.


req [requirement] Information Management [Information Management]
Depends on
Depends on
«trace»
«deriveReqt»
«Capability» Hosting Services
«Capability» Monitor & Control
«Capability» Software Assurance
«Capability» Configuration Management
«Capability» Solution Development

«requirement» «Maturity Level 2» **Metrics**
Id = "Gov_4"
Text = "The system shall be able to automatically collect, correlate and display metrics associated with productivity, reliability, quality, security, and operations of both the system and the product under development and associated technologies, approaches, and methods, and models used throughout the product's life in order to achieve an organization's objectives and business needs."

«requirement» «Maturity Level 1» **Knowledge Management**
Id = "Gov_5"
Text = "The system shall provide integrated knowledge management"

«requirement» «Maturity Level 3» **Information Management**
Id = "Sys_5"
Text = "The system information tools and services shall be integrated into the relevant DevSecOps pipelines within the system and product under development lifecycles processes as needed to meet the organization's, project's, and team's informational needs."

«requirement» «Maturity Level 3» **Audit of Measurement Strategy**
Id = "Gov_5.4.7"
Text = "The system shall be capable of supporting an independent quality assurance review or audit of activities and work products associated with the measurement strategy and associated metrics, information management, and reporting processes."

«requirement» «Maturity Level 2» **System Logs**
Id = "Sys_5.1"
Text = "The system transaction logs shall be centrally located and stored in a format that meets the organization's, project's, and team's information needs."

«requirement» «Maturity Level 2» **Information Storage**
Id = "Sys_5.2"
Text = "The system shall store information collected in such ways as to allow information producers and consumers the ability to rapidly and consistently gather, store, transform, and retrieve the information based on a defined retention period."

«requirement» «Maturity Level 3» **Stakeholder Metrics**
Id = "Gov_4.6"
Text = "The system shall be able to provide relevant stakeholders with indicators and other measures needed to make early development and real-time operational decisions as they relate to defined objectives."

«requirement» «Maturity Level 2» **Immutable Logs**
Id = "Sys_5.1.1"
Text = "The system transaction logs should be immutable."

«documentation»
This helps to find anomalous patterns.

«requirement» «Maturity Level 3» **Log Visualization & Analysis**
Id = "Sys_5.1.2"
Text = "The system shall support the ability to visualize log data in various ways and perform log analysis."

«requirement» «Maturity Level 2» **Information Security, Retention, and Disposal**
Id = "Sys_5.2.1"
Text = "The system shall secure, retain, and dispose of information being collected, stored, processed, or transmitted based on the sensitivity and privacy concerns and the laws and regulations associated to the given information."

«requirement» «Maturity Level 2» **Need to Know**
Id = "Sys_5.2.2"
Text = "The system shall control access to information based on an individual's or application's needs."

«requirement» «Maturity Level 3» **Information Security Risks**
Id = "Sys_5.2.3"
Text = "Security risks of the underlying infrastructure shall be measured and quantified, so that the total risks and impacts to software applications are understood."

«requirement» «Maturity Level 2» **Disaster Recovery**
Id = "Sys_5.2.4"
Text = "A disaster recovery plan shall be documented to provide mitigations in the event of a disaster."

«documentation»
At a minimum, the plan must include (a) a list and description of possible disasters (b) an inventory of assets and services, (b) the location of data, (c) a data backup approach, (d) an identification of the disaster recovery team.

«requirement» «Maturity Level 1» **System's Non-functional Requirements**
Id = "Sys_1"
Text = "The system's infrastructure shall be designed and implemented to meet the organizations or specific product under development's reliability, security, dependability, scalability, and regulatory needs."

«requirement» «Maturity Level 3» **Policy as Code**
Id = "Sys_5.2.1.1"
Text = "The system shall implement policy as code."

«requirement» «Maturity Level 3» **Vulnerability Management**
Id = "Sys_5.2.1.2"
Text = "The system shall support Security Content Automation Protocol (SCAP) [11] and container configuration policies. These policies can be defined as needed."

«documentation»
Provides automated policy enforcement.

«requirement» «Maturity Level 1» **Roles and Responsibilities**
Id = "Gov_5.6"
Text = "The system shall be able to capture defined roles, responsibilities, accountabilities, and authorities as they relate to both the system and product under development lifecycles. "

«documentation»
With policy as code, the system can incorporate software versions of policies earlier and more effectively into lifecycles, compared to manual methods. Relevant stakeholders can apply policies more consistently and more rapidly to any number of development flows and deployments, at different stages of DevSecOps pipelines and in other contexts. As the numbers of applications and the policies governing them rise, policy as code also facilitates automated testing of adherence to policies that is efficient and error free, compared to manual testing that is rapidly overwhelmed. Through use of policy as code, relevant stakeholders can ensure that as the application and IT activities of a given enterprise or organization increase, it can continue to apply security, compliance, and other rules with reliability, scalability, and cost-effectiveness [9].

Figure 55: Information Management

3.7.5.1 Sys_5.1 System Logs

The system transaction logs shall be centrally located and stored in a format that meets the organization's, project's, and team's information needs.

3.7.5.1.1 Sys_5.1.1 Immutable Logs

The system transaction logs should be immutable.

3.7.5.1.2 Sys_5.1.2 Log Visualization & Analysis

The system shall support the ability to visualize log data in various ways and perform log analysis.

Documentation:

This helps to find anomalous patterns.

3.7.5.2 Sys_5.2 Information Storage

The system shall store information collected in such ways as to allow information producers and consumers the ability to rapidly and consistently gather, store, transform, and retrieve the information based on a defined retention period.

3.7.5.2.1 Sys_5.2.1 Information Security, Retention, and Disposal

The system shall secure, retain, and dispose of information being collected, stored, processed, or transmitted based on the sensitivity and privacy concerns and the laws and regulations associated to the given information.

3.7.5.2.1.1 Sys_5.2.1.1 Policy as Code

The system shall implement policy as code.

Documentation:

With policy as code, the system can incorporate software versions of policies earlier and more effectively into lifecycles, compared to manual methods. Relevant stakeholders can apply policies more consistently and more rapidly to any number of development flows and deployments, at different stages of DevSecOps pipelines and in other contexts. As the numbers of applications and the policies governing them rise, policy as code also facilitates automated testing of adherence to policies that is efficient and error free, compared to manual testing that is rapidly overwhelmed. Through use of policy as code, relevant stakeholders can ensure that as the application and IT activities of a given enterprise or organization increase, it can continue to apply security, compliance, and other rules with reliability, scalability, and cost-effectiveness [Milgram 2020].

3.7.5.2.1.2 Sys_5.2.1.2 Vulnerability Management

The system shall support Security Content Automation Protocol (SCAP) [Wikipedia 2021b] and container configuration policies. These policies can be defined as needed.

Documentation:

Provides automated policy enforcement.

3.7.5.2.2 Sys_5.2.2 Need to Know

The system shall control access to information based on an individual's or application's needs.

3.7.5.2.3 Sys_5.2.3 Information Security Risks

Security risks of the underlying infrastructure shall be measured and quantified, so that the total risks and impacts to software applications are understood.

3.7.5.2.4 Sys_5.2.4 Disaster Recovery

A disaster recovery plan shall be documented to provide mitigations in the event of a disaster.

Documentation:

At a minimum, the plan must include (a) a list and description of possible disasters (b) an inventory of assets and services, (b) the location of data, (c) a data backup approach, (d) an identification of the disaster recovery team.

3.7.6 Sys_6 Infrastructure Configuration Management

An infrastructure configuration plan shall be developed that accounts for all infrastructure configuration items.

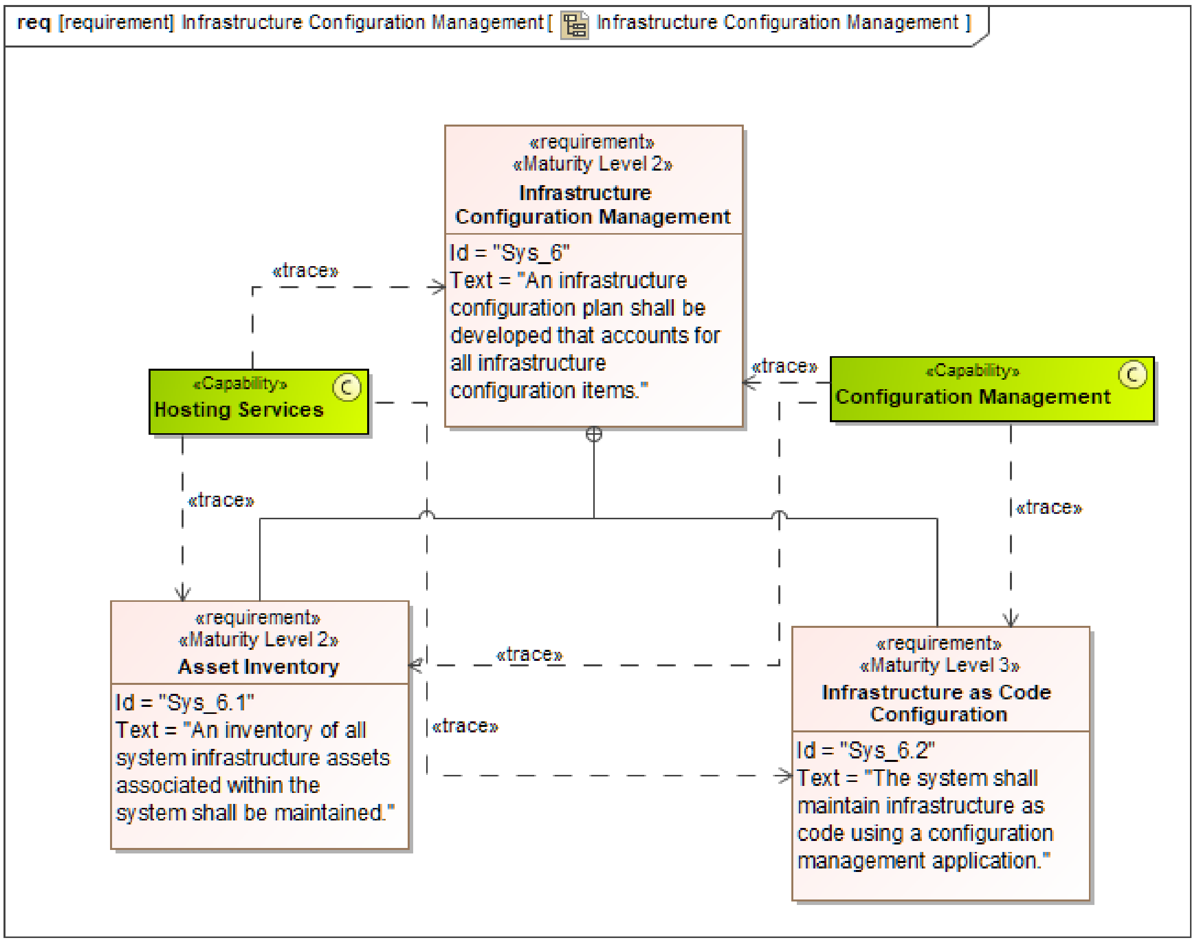

req [requirement] Infrastructure Configuration Management [Infrastructure Configuration Management]
«requirement» «Maturity Level 2» Infrastructure Configuration Management
Id = "Sys_6"
Text = "An infrastructure configuration plan shall be developed that accounts for all infrastructure configuration items."
«Capability» Hosting Services
«Capability» Configuration Management
«trace»
«trace»
«trace»
«trace»
«trace»
«trace»
«requirement» «Maturity Level 2» Asset Inventory
Id = "Sys_6.1"
Text = "An inventory of all system infrastructure assets associated within the system shall be maintained."
«requirement» «Maturity Level 3» Infrastructure as Code Configuration
Id = "Sys_6.2"
Text = "The system shall maintain infrastructure as code using a configuration management application."

Figure 56: Infrastructure Configuration Management

3.7.6.1 Sys_6.1 Asset Inventory

An inventory of all system infrastructure assets associated within the system shall be maintained.

3.7.6.2 Sys_6.2 Infrastructure as Code Configuration

The system shall maintain infrastructure as code using a configuration management application.

3.7.7 Sys_7 Automated Patch Management

The system shall enable a process by which the operating system software and supporting services are upgraded automatically.

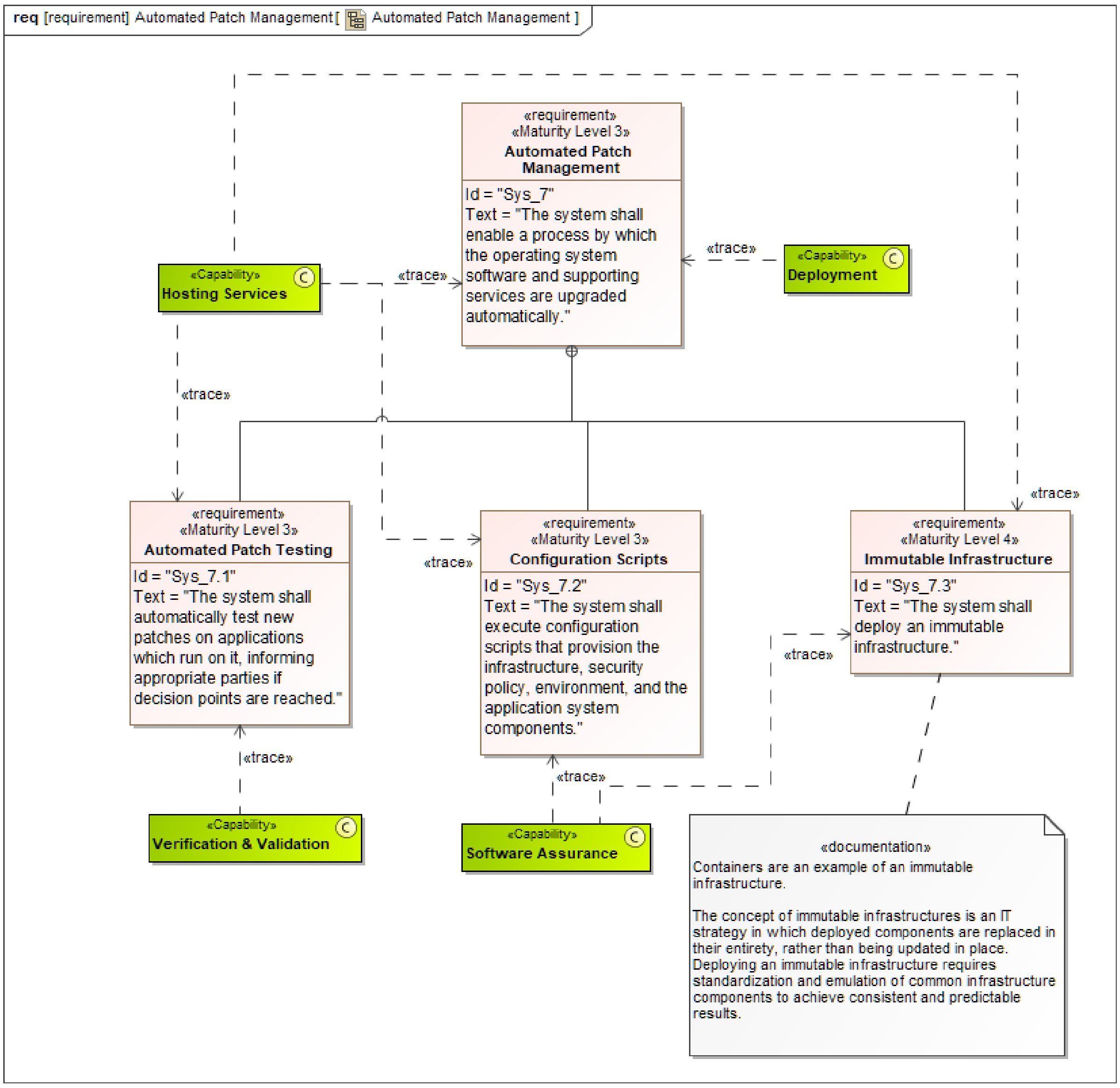

req [requirement] Automated Patch Management [Automated Patch Management]
«requirement» «Maturity Level 3» Automated Patch Management
Id = "Sys_7"
Text = "The system shall enable a process by which the operating system software and supporting services are upgraded automatically."
«Capability» Hosting Services
«Capability» Deployment
«trace»
«requirement» «Maturity Level 3» Automated Patch Testing
Id = "Sys_7.1"
Text = "The system shall automatically test new patches on applications which run on it, informing appropriate parties if decision points are reached."
«requirement» «Maturity Level 3» Configuration Scripts
Id = "Sys_7.2"
Text = "The system shall execute configuration scripts that provision the infrastructure, security policy, environment, and the application system components."
«requirement» «Maturity Level 4» Immutable Infrastructure
Id = "Sys_7.3"
Text = "The system shall deploy an immutable infrastructure."
«Capability» Verification & Validation
«Capability» Software Assurance
«documentation»
Containers are an example of an immutable infrastructure.
The concept of immutable infrastructures is an IT strategy in which deployed components are replaced in their entirety, rather than being updated in place. Deploying an immutable infrastructure requires standardization and emulation of common infrastructure components to achieve consistent and predictable results.

Figure 57: Automated Patch Management

3.7.7.1 Sys_7.1 Automated Patch Testing

The system shall automatically test new patches on applications which run on it, informing appropriate parties if decision points are reached.

3.7.7.2 Sys_7.2 Configuration Scripts

The system shall execute configuration scripts that provision the infrastructure, security policy, environment, and the application system components.

3.7.7.3 Sys_7.3 Immutable Infrastructure

The system shall deploy an immutable infrastructure.

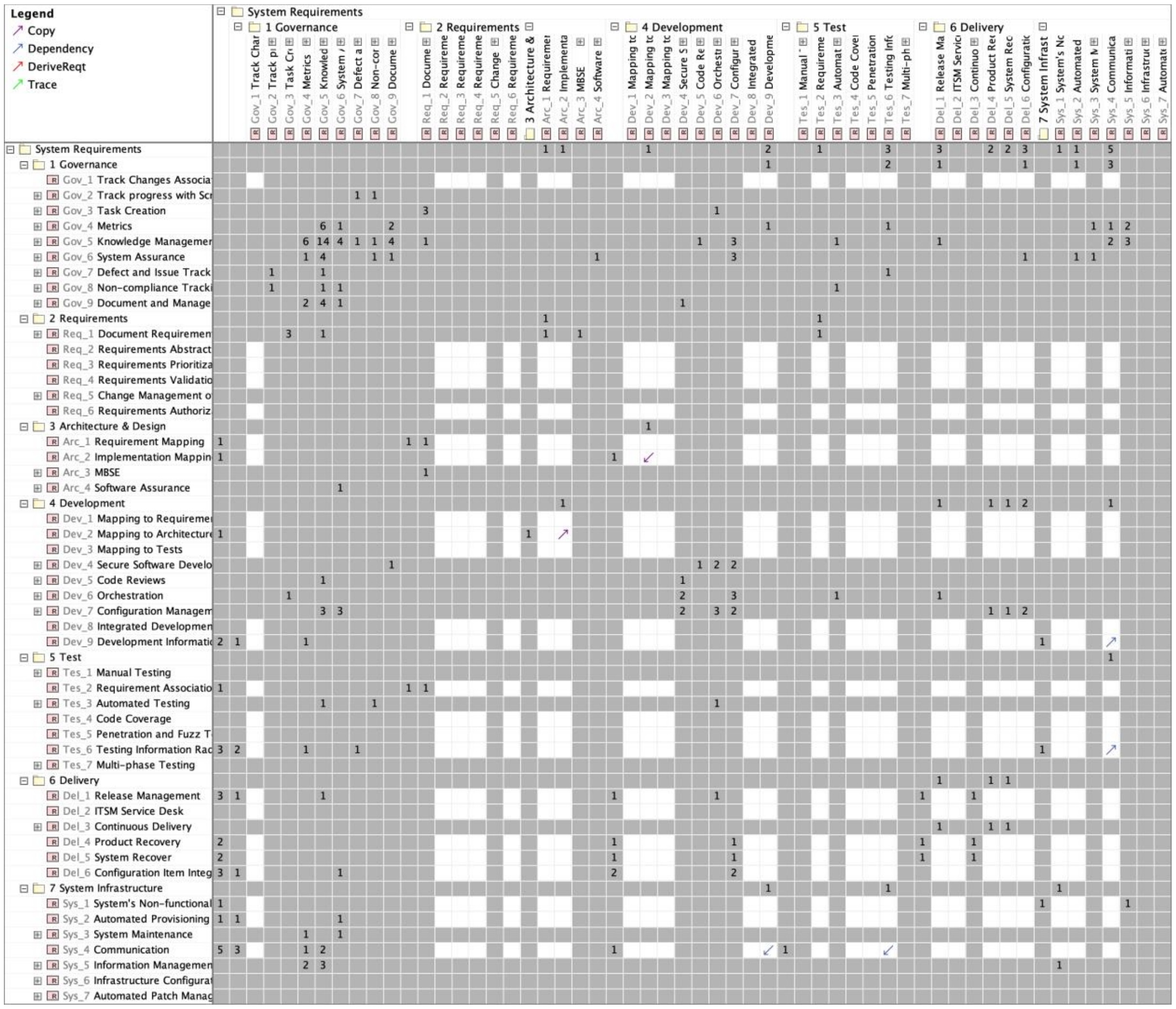
Documentation:

Containers are an example of an immutable infrastructure.

The concept of immutable infrastructures is an IT strategy in which deployed components are replaced in their entirety, rather than being updated in place. Deploying an immutable infrastructure requires standardization and emulation of common infrastructure components to achieve consistent and predictable results.

Appendix

Requirements To Requirements Relationships Matrix

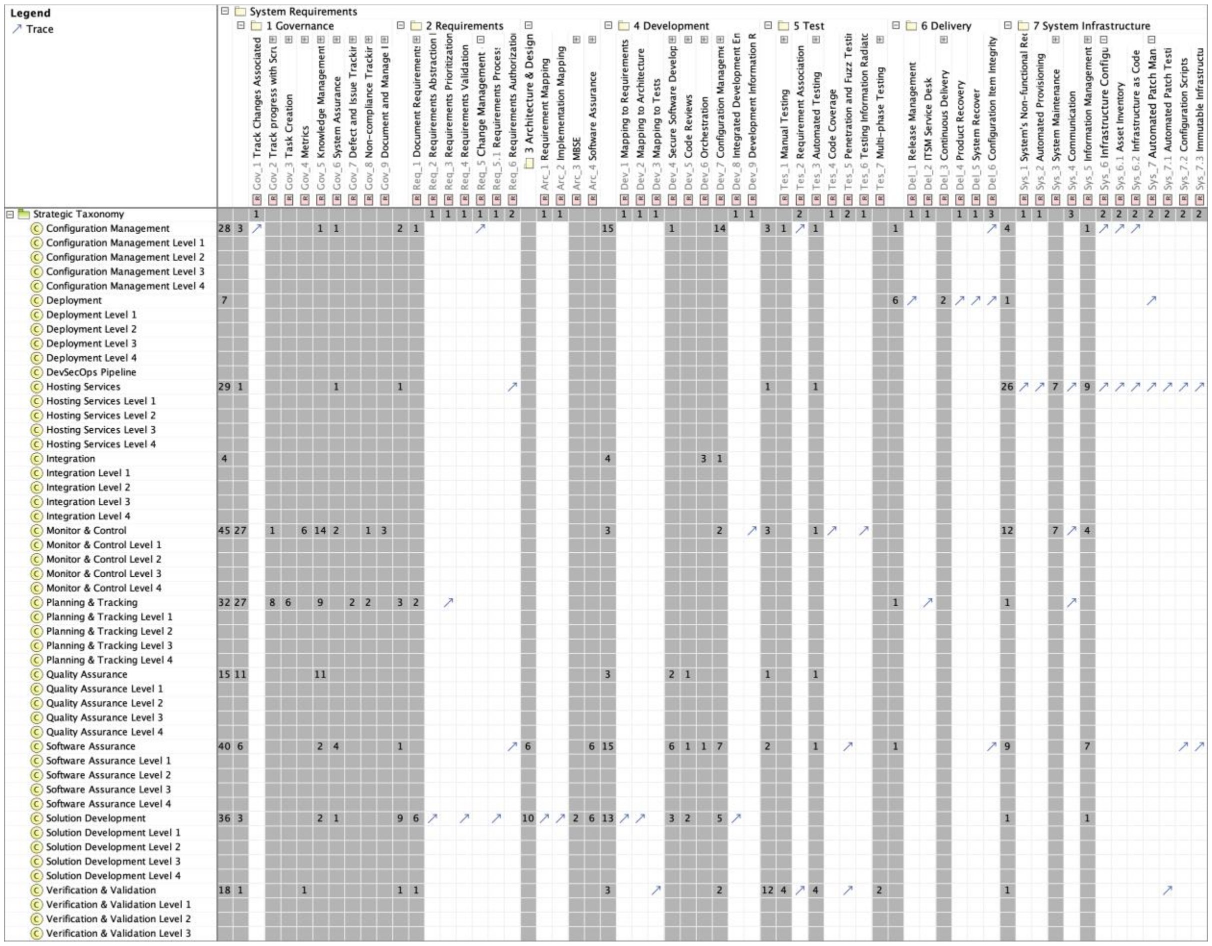


Legend
↗ Copy
↗ Dependency
↗ DeriveReqt
↗ Trace

	System Requirements	1 Governance	Gov_1 Track Char	Gov_2 Track pı	Gov_3 Task Cr	Gov_4 Metrics	Gov_5 Knowled	Gov_6 System /	Gov_7 Defect a	Gov_8 Non-cor	Gov_9 Docume	2 Requirements	Req_1 Docume	Req_2 Requireme	Req_3 Requireme	Req_4 Requireme	Req_5 Change	Req_6 Requireme	3 Architecture &	Arc_1 Requiremer	Arc_2 Implementa	Arc_3 MBSE	Arc_4 Software	4 Development	Dev_1 Mapping tc	Dev_2 Mapping tc	Dev_3 Mapping tc	Dev_4 Secure S	Dev_5 Code Re	Dev_6 Orchestr	Dev_7 Configur	Dev_8 Integrated	Dev_9 Developme	5 Test	Tes_1 Manual	Tes_2 Requireme	Tes_3 Automat	Tes_4 Code Cover	Tes_5 Penetration	Tes_6 Testing Infc	Tes_7 Multi-ph	6 Delivery	Del_1 Release Ma	Del_2 ITSM Servic	Del_3 Continuo	Del_4 Product Rec	Del_5 System Rec	Del_6 Configuratic	7 System Infrast	Sys_1 System's Nc	Sys_2 Automated	Sys_3 System M	Sys_4 Communica	Sys_5 Informati	Sys_6 Infrastruc	Sys_7 Automate
System Requirements																				1	1					1							2			1				3			3			2	2	3		1	1		5			
1 Governance																																	1							2			1					1			1		3			
Gov_1 Track Changes Associa																																																								
Gov_2 Track progress with Scr									1	1																																														
Gov_3 Task Creation													3																	1																										
Gov_4 Metrics							6	1			2																						1							1												1	1	2		
Gov_5 Knowledge Managemer						6	14	4	1	1	4		1																1		3						1						1										2	3		
Gov_6 System Assurance						1	4			1	1												1								3																	1			1	1				
Gov_7 Defect and Issue Track				1			1																																	1																
Gov_8 Non-compliance Tracki				1			1	1																													1																			
Gov_9 Document and Manage						2	4	1																				1																												
2 Requirements																				1																1																				
Req_1 Document Requiremen					3		1													1		1														1																				
Req_2 Requirements Abstract																																																								
Req_3 Requirements Prioritiza																																																								
Req_4 Requirements Validatio																																																								
Req_5 Change Management o																																																								
Req_6 Requirements Authoriz																																																								
3 Architecture & Design																										1																														
Arc_1 Requirement Mapping	1											1	1																																											
Arc_2 Implementation Mappin	1																							1		↙																														
Arc_3 MBSE													1																																											
Arc_4 Software Assurance								1																																																
4 Development																					1																						1			1	1	2					1			
Dev_1 Mapping to Requireme																																																								
Dev_2 Mapping to Architectur	1																		1		↗																																			
Dev_3 Mapping to Tests																																																								
Dev_4 Secure Software Develo											1																		1	2	2																									
Dev_5 Code Reviews							1																					1																												
Dev_6 Orchestration					1																							2			3						1						1													
Dev_7 Configuration Managem							3	3																				2		3	2															1	1	2								
Dev_8 Integrated Developmen																																																								
Dev_9 Development Informatio	2	1				1																																											1				↗			
5 Test																																																					1			
Tes_1 Manual Testing																																																								
Tes_2 Requirement Associatio	1											1	1																																											
Tes_3 Automated Testing							1			1																				1																										
Tes_4 Code Coverage																																																								
Tes_5 Penetration and Fuzz T																																																								
Tes_6 Testing Information Rac	3	2				1			1																																								1				↗			
Tes_7 Multi-phase Testing																																																								
6 Delivery																																											1			1	1									
Del_1 Release Management	3	1					1																	1						1												1			1											
Del_2 ITSM Service Desk																																																								
Del_3 Continuous Delivery																																											1			1	1									
Del_4 Product Recovery	2																							1							1											1			1											
Del_5 System Recover	2																							1							1											1			1											
Del_6 Configuration Item Integ	3	1						1																2							2																									
7 System Infrastructure																																	1							1										1						
Sys_1 System's Non-functional	1																																																1					1		
Sys_2 Automated Provisioning	1	1						1																																																
Sys_3 System Maintenance						1		1																																																
Sys_4 Communication	5	3				1	2																	1									↙	1						↙																
Sys_5 Information Managemen						2	3																																											1						
Sys_6 Infrastructure Configurat																																																								
Sys_7 Automated Patch Manag																																																								

Figure 58: Requirements to Requirements Relationship Matrix

Capabilities To Requirements Relationships Matrix



Legend

↗ Trace

	System Requirements	1 Governance	Gov_1 Track Changes Associated	Gov_2 Track progress with Scr	Gov_3 Task Creation	Gov_4 Metrics	Gov_5 Knowledge Management	Gov_6 System Assurance	Gov_7 Defect and Issue Trackir	Gov_8 Non-compliance Trackir	Gov_9 Document and Manage I	2 Requirements	Req_1 Document Requirements	Req_2 Requirements Abstraction I	Req_3 Requirements Prioritization	Req_4 Requirements Validation	Req_5 Change Management	Req_5.1 Requirements Process	Req_6 Requirements Authorizatio	3 Architecture & Design	Arc_1 Requirement Mapping	Arc_2 Implementation Mapping	Arc_3 MBSE	Arc_4 Software Assurance	4 Development	Dev_1 Mapping to Requirements	Dev_2 Mapping to Architecture	Dev_3 Mapping to Tests	Dev_4 Secure Software Develop	Dev_5 Code Reviews	Dev_6 Orchestration	Dev_7 Configuration Manageme	Dev_8 Integrated Development En	Dev_9 Development Information R	5 Test	Tes_1 Manual Testing	Tes_2 Requirement Association	Tes_3 Automated Testing	Tes_4 Code Coverage	Tes_5 Penetration and Fuzz Testi	Tes_6 Testing Information Radiatc	Tes_7 Multi-phase Testing	6 Delivery	Del_1 Release Management	Del_2 ITSM Service Desk	Del_3 Continuous Delivery	Del_4 Product Recovery	Del_5 System Recover	Del_6 Configuration Item Integrity	7 System Infrastructure	Sys_1 System's Non-functional Re	Sys_2 Automated Provisioning	Sys_3 System Maintenance	Sys_4 Communication	Sys_5 Information Management	Sys_6 Infrastructure Configu	Sys_6.1 Asset Inventory	Sys_6.2 Infrastructure as Code	Sys_7 Automated Patch Man	Sys_7.1 Automated Patch Testi	Sys_7.2 Configuration Scripts	Sys_7.3 Immutable Infrastructu
Strategic Taxonomy			1											1	1	1	1	1	2		1	1				1	1	1					1	1			2		1	2	1			1	1		1	1	3		1	1		3		2	2	2	2	2	2	2
Configuration Management	28	3	↗				1	1				2	1				↗								15				1			14			3	1	↗	1					1						↗	4					1	↗	↗	↗				
Configuration Management Level 1																																																														
Configuration Management Level 2																																																														
Configuration Management Level 3																																																														
Configuration Management Level 4																																																														
Deployment	7																																										6	↗		2	↗	↗	↗	1									↗			
Deployment Level 1																																																														
Deployment Level 2																																																														
Deployment Level 3																																																														
Deployment Level 4																																																														
DevSecOps Pipeline																																																														
Hosting Services	29	1						1				1							↗																1			1												26	↗	↗	7	↗	9	↗	↗	↗	↗	↗	↗	↗
Hosting Services Level 1																																																														
Hosting Services Level 2																																																														
Hosting Services Level 3																																																														
Hosting Services Level 4																																																														
Integration	4																								4						3	1																														
Integration Level 1																																																														
Integration Level 2																																																														
Integration Level 3																																																														
Integration Level 4																																																														
Monitor & Control	45	27		1		6	14	2		1	3														3							2		↗	3			1	↗		↗									12			7	↗	4							
Monitor & Control Level 1																																																													
Monitor & Control Level 2																																																														
Monitor & Control Level 3																																																														
Monitor & Control Level 4																																																														
Planning & Tracking	32	27		8	6		9		2	2		3	2		↗																												1		↗					1				↗								
Planning & Tracking Level 1																																																														
Planning & Tracking Level 2																																																														
Planning & Tracking Level 3																																																														
Planning & Tracking Level 4																																																														
Quality Assurance	15	11					11																		3				2	1					1			1																								
Quality Assurance Level 1																																																														
Quality Assurance Level 2																																																														
Quality Assurance Level 3																																																														
Quality Assurance Level 4																																																														
Software Assurance	40	6					2	4				1							↗	6				6	15				6	1	1	7			2			1		↗			1						↗	9					7						↗	↗
Software Assurance Level 1																																																														
Software Assurance Level 2																																																														
Software Assurance Level 3																																																														
Software Assurance Level 4																																																														
Solution Development	36	3					2	1				9	6	↗		↗		↗		10	↗	↗	2	6	13	↗	↗		3	2		5	↗																	1					1							
Solution Development Level 1																																																														
Solution Development Level 2																																																														
Solution Development Level 3																																																														
Solution Development Level 4																																																														
Verification & Validation	18	1				1						1	1												3			↗				2			12	4	↗	4		↗		2								1										↗		
Verification & Validation Level 1																																																														
Verification & Validation Level 2																																																														
Verification & Validation Level 3																																																														

Figure 59: Capabilities to Requirements Relationship Matrix

References/Bibliography

URLs are valid as of the publication date of this document.

[Chrissis 2011]
Chrissis, Mary Beth; Konrad, Mike; & Sandy Shrum. *CMMI for Development: Guidance for Process Integration and Product Improvement - Third Edition*. Pearson Education, Inc. 2011.

[U.S. General Services Administration 2021]
U.S. General Services Administration. DevSecOps Guide: Standard DevSecOps Platform Framework. *U.S. General Services Administration Website*. May 17, 2021 [accessed]. https://tech.gsa.gov/guides/dev_sec_ops_guide.

[U.S. Department of Defense 2020]
Department of Defense. DoD Instruction 5000.87*, Operation of the Software Acquisition Pathway*. October 2, 2020.

[Wikipedia 2021a]
Wikipedia. Code coverage. *Wikipedia Website*. February, 1 2021 [accessed]. https://en.wikipedia.org/wiki/Code_coverage.

[The MITRE Corporation 2021a]
The MITRE Corporation. CVE. *CVE Website*. July 13, 2021 [accessed]. https://cve.mitre.org

[The MITRE Corporation 2021b]
The MITRE Corporation. CWE – Common Weakness Enumeration. *CWE Website*. July 13, 2021 [accessed]. https://cwe.mitre.org

[IEEE 2021]
IEEE. *IEEE Standard for DevOps: Building Reliable and Secure Systems Including Application Build, Package, and Deployment*. IEEE Std 2675-2021. Software & Systems Engineering Standards Committee of the IEEE Computer Society, IEEE SA Standards Board, Approved 9 February 2021.

[Girard 2015]
Girard, John P. & Girard, JoAnn L. (2015). Defining knowledge management: Toward an applied compendium. *Online Journal of Applied Knowledge Management*. Volume 3. Issue 1. 2015. Page 14.

[Milgram 2020]
Milgram, Jason. Policy as Code: how we got there and why you benefit. *LinkedIn*. Published October 8, 2020. https://www.linkedin.com/pulse/policy-code-how-we-got-why-you-benefit-jason-milgram

[Study.com 2020]
Study.com. Scrum Board: Definition & Examples. *Study.com Website*. January 5, 2020 [accessed]. https://study.com/academy/lesson/scrum-board-definition-examples.html.

[Wikipedia 2021b]
Wikipedia. Security Content Automation Protocol. *Wikipedia Website*. May 17, 2021 [accessed]. https://en.wikipedia.org/wiki/Security_Content_Automation_Protocol.

[Wikipedia 2021c]
Wikipedia. Software Bill of Materials. *Wikipedia Website*. February 1, 2021 [accessed]. https://en.wikipedia.org/wiki/Software_bill_of_materials.

[U.S. Army 2021d]
Wikipedia. Kanban board. *Wikipedia Website*. February 1, 2021 [accessed]. https://en.wikipedia.org/wiki/Kanban_board.

[Shevchenko 2020]
Shevchenko, Nataliya. An Introduction to Model-Based Systems Engineering (MBSE) [blog post]. *SEI Blog*. January 5, 2020 [accessed]. https://insights.sei.cmu.edu/sei_blog/2020/12/an-introduction-to-model-based-systems-engineering-mbse.html.

[Scaled Agile, Inc. 2021]
Scaled Agile, Inc. Value Steams. *Scaled Agile Framework (SAFe)*. April, 27 2021 [accessed]. https://www.scaledagileframework.com/value-streams/.

[Joint Task Force Transformation Initiative Interagency Working Group 2020]
Joint Task Force Transformation Initiative Interagency Working Group. Security and Privacy Controls for Information Systems and Organizations. *NIST Special Publication*. (SP) 800-53, Revision. 5. https://doi.org/10.6028/NIST.SP.800-53r5

[Woody 2020]
Woody, Carol; Chick, Timothy; Reffett, Aaron; Pavetti, Scott; Laughlin, Richard; Frye, Brent; & Bandor, Michael. 2020. *DevSecOps Pipeline for Complex Software-Intensive Systems: Addressing Cybersecurity Challenges*. The Journal on Systemics, Cybernetics and Informatics: JSCI, Volume 18. Number 5. 2020. Pages 31-36. ISSN: 1690-4.

[U.S. Department of Defense 2018]
U.S. Department of Defense. DoD Reference Architecture Description, https://dodcio.defense.gov/Portals/0/Documents/DIEA/Ref_Archi_Description_Final_v1_18Jun10.pdf .

REPORT DOCUMENTATION PAGE			*Form Approved OMB No. 0704-0188*
Public reporting burden for this collection of information is estimated to average 1 hour per response, including the time for reviewing instructions, searching existing data sources, gathering and maintaining the data needed, and completing and reviewing the collection of information. Send comments regarding this burden estimate or any other aspect of this collection of information, including suggestions for reducing this burden, to Washington Headquarters Services, Directorate for information Operations and Reports, 1215 Jefferson Davis Highway, Suite 1204, Arlington, VA 22202-4302, and to the Office of Management and Budget, Paperwork Reduction Project (0704-0188), Washington, DC 20503.			
1. **AGENCY USE ONLY** (Leave Blank)	2. **REPORT DATE** August 2021		3. **REPORT TYPE AND DATES COVERED** Final
4. **TITLE AND SUBTITLE** DevSecOps Platform-Independent Model: Requirements and Capabilities			5. **FUNDING NUMBERS** FA8702-15-D-0002
6. **AUTHOR(S)**			
7. **PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)** Software Engineering Institute Carnegie Mellon University Pittsburgh, PA 15213			8. **PERFORMING ORGANIZATION REPORT NUMBER** CMU/SEI-2021-TR-010
9. **SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)** SEI Administrative Agent AFLCMC/AZS 5 Eglin Street Hanscom AFB, MA 01731-2100			10. **SPONSORING/MONITORING AGENCY REPORT NUMBER** n/a
11. **SUPPLEMENTARY NOTES**			
12A **DISTRIBUTION/AVAILABILITY STATEMENT** Unclassified/Unlimited, DTIC, NTIS			12B **DISTRIBUTION CODE**
13. **ABSTRACT (MAXIMUM 200 WORDS)**			
14. **SUBJECT TERMS**			15. **NUMBER OF PAGES** 98
16. **PRICE CODE**			
17. **SECURITY CLASSIFICATION OF REPORT** Unclassified	18. **SECURITY CLASSIFICATION OF THIS PAGE** Unclassified	19. **SECURITY CLASSIFICATION OF ABSTRACT** Unclassified	20. **LIMITATION OF ABSTRACT** UL

NSN 7540-01-280-5500

Standard Form 298 (Rev. 2-89) Prescribed by ANSI Std. Z39-18
298-102