Autonomous Vehicle <Something> Development Lifecycle (*AV<X>DL*)

<author>

<title>

Version <version>

Table of Contents

[1. Introduction 6](#_Toc54694591)

[2. Overview 8](#_Toc54694592)

[3. Philosophy 10](#_Toc54694593)

[4. Background Material 11](#_Toc54694594)

[5. Continuous Improvement 13](#_Toc54694595)

[6. Relationship to ISO 21434 14](#_Toc54694596)

[7. Hardware-Software Relationship 15](#_Toc54694597)

[8. Implementation Framework 16](#_Toc54694598)

[9. Process Phases and Requirements 20](#_Toc54694599)

[9.1 Foundation Phase 21](#_Toc54694600)

[9.1.1 Training [AVXDL-Foundation-1] 22](#_Toc54694601)

[9.1.2 Roles and Responsibilities [AVXDL-Foundation-2] 24](#_Toc54694602)

[9.2 Requirements Phase 25](#_Toc54694603)

[9.2.1 Security Requirements Definition [AVXDL-Requirements-1] 26](#_Toc54694604)

[9.2.2 Requirements Gate [AVXDL-Requirements-2] 28](#_Toc54694605)

[9.3 Design Phase 29](#_Toc54694606)

[9.3.1 Apply Security Requirements and Risk Information to Design [AVXDL-Design-1] 30](#_Toc54694607)

[9.3.2 Security Design Review [AVXDL-Design-2] 32](#_Toc54694608)

[9.3.5 Design Gate [AVXDL-Design-5] 34](#_Toc54694609)

[9.4 Implementation Phase 35](#_Toc54694610)

[9.5 Verification Phase 36](#_Toc54694611)

[9.6 Release Phase 37](#_Toc54694612)

[9.7 Operation Phase 38](#_Toc54694613)

[9.8 Decommissioning Phase 39](#_Toc54694614)

[10. Requirement Role Assignments 40](#_Toc54694615)

[11. Groups 41](#_Toc54694616)

[11.1 Groups [devops] 43](#_Toc54694617)

[11.2 Groups [development] 44](#_Toc54694618)

[11.3 Groups [security] 45](#_Toc54694619)

[12. NCWF Roles 46](#_Toc54694620)

[12.1 Cyber Defense Forensics Analyst (IN-FOR-002) 47](#_Toc54694621)

[13. SSDF Background Material 48](#_Toc54694622)

[13.1 Prepare the Organization (PO) Practices 49](#_Toc54694623)

[13.2 Protect Software (PS) Practices 50](#_Toc54694624)

[15. ISO/SAE 21434 Background Material 51](#_Toc54694625)

[15.1 Overall Cybersecurity Management (5) 53](#_Toc54694626)

[15.4 21434 Risk Assessment (8) 55](#_Toc54694627)

[16. Reference Documents 57](#_Toc54694628)

[17. Continuous Improvement Progress Summary Example 60](#_Toc54694629)

[18. AVXDL Product Dependencies 61](#_Toc54694630)

[19. AVCDL Training Path 69](#_Toc54694631)

List of Tables

[Table 1 - relationship among standards 9](#_Toc54694327)

[Table 2 - requirement role assignments 40](#_Toc54694328)

[Table 3 - requirement-group mapping 42](#_Toc54694329)

[Table 4 - devops requirement responsibilities 43](#_Toc54694330)

[Table 5 - development requirement responsibilities 44](#_Toc54694331)

[Table 6 - security requirement responsibilities 45](#_Toc54694332)

[Table 7 - maturity tracking example 60](#_Toc54694333)

List of Figures

[Figure 1 - AVXDL sources 12](#_Toc54694703)

[Figure 2 - AVXDL phases 16](#_Toc54694704)

[Figure 3 - AVXDL phases cyclic 18](#_Toc54694705)

[Figure 4 – AV<X>DL-PDCA Phase Requirement Mapping 19](#_Toc54694706)

[Figure 5 - ISO 21434 document overview 52](#_Toc54694707)

[Figure 6 - AVXDL product dependencies – foundation phase 61](#_Toc54694708)

[Figure 7 - AVXDL product dependencies – requirements phase 62](#_Toc54694709)

[Figure 8 - AVXDL product dependencies – design phase 63](#_Toc54694710)

[Figure 9 - AVXDL product dependencies – implementation phase 64](#_Toc54694711)

[Figure 10 - AVXDL product dependencies – verification phase 65](#_Toc54694712)

[Figure 11 - AVXDL product dependencies – release phase 66](#_Toc54694713)

[Figure 12 - AVXDL product dependencies – operation phase 67](#_Toc54694714)

[Figure 13 - AVXDL product dependencies – decommissioning phase 68](#_Toc54694715)

[Figure 14 - AVCDL Training Path 69](#_Toc54694716)

# Introduction

## Abstract

This material documents a ***proposed*** Autonomous Vehicle <Something> Development Lifecycle (***AV<X>DL***).

## Questions, Errors and Other

This documentation is generally maintained by [Charles Wilson](mailto:charles.wilson@aptiv.com?subject=AVCDL%20process%20documentation%20query&body=I%20have%20a%20question%20about%20the%20SDL%20process%20documentation.). Please feel free to contact him should you have any questions.

## Scannable email query QR



**Note:** QR code generated [using](https://www.qr-code-generator.com/).

## Organization

This document is organized as follows:

Diagram

Description automatically generated

# Overview

The ***AVXDL*** is intended as an **overlay** to a pre-existing software development process. This may be an [ISO 26262](https://en.wikipedia.org/wiki/ISO_26262)- or [IEC 62304](https://en.wikipedia.org/wiki/IEC_62304)-based one.

## What it is

The ***AVXDL*** is a set of identified processes, requirements those processes, generated products and mapping to corresponding ISO/SAE 21434 work products; for the purpose of ensuring the creation of secure systems. It is intended to support auditing of the development process in the area of security as specified in ISO/SAE 21434.

## What it isn’t

The ***AVXDL*** does not attempt to specify:

* implementation methodology (waterfall, agile, TDD, BDD, spiral, etc.)
* underlying development tools
* remediation methodology

## Where it fits

The ***AVXDL*** is not a standalone solution. It is intended to implement the non-organizational elements of a larger product development lifecycle framework (***AVPDL***). Moreover, it is designed to overlay both the safety, software and system development lifecycle frameworks as specified in ISO 26262, ISO/IEC 12207 and ISO/IEC 15288 respectively.

We can visualize the relationship between these as follows:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **AVPDL** | **AVXDL** | **15288** | **12207** | **26262** | **21434** |
| organization processes | N/A | technical processes | technical processes | management of functional safety | overall cybersecurity management |
| supporting processes | project dependent cybersecurity management |
| foundation phase | [foundation phase](#_9.1_Foundation_Phase) | N/A | N/A | concept phase | concept phase |
| requirements phase | [requirements phase](#_9.2_Requirements_Phase) | requirements definition | requirements definition | safety requirements | cybersecurity requirements |
| requirements analysis | system requirements analysis | hazard analysis / risk assessment | cybersecurity assessment |
| design phase | [design phase](#_9.3_Design_Phase) | architectural design | system architectural design | architectural design | cybersecurity design |
| implementation phase | [implementation phase](#_9.4_Implementation_Phase) | implementation | implementation | implementation | development |
| integration | system integration | integration and verification | integration and verification |
| verification phase | [verification phase](#_9.5_Verification_Phase) | verification | system qualification testing |
| transition | software installation |
| software acceptance support |
| release phase | [release phase](#_9.6_Release_Phase) | validation | production | production |
| operation phase | [operation phase](#_9.7_Operation_Phase) | operation | software operation | operation, service and decommissioning | continuous cybersecurity activities |
| maintenance | software maintenance | operation and maintenance |
| decommissioning phase | [decommissioning phase](#_9.8_Decommissioning_Phase) | disposal | software disposal | decommissioning |
| supplier processes | N/A | agreement processes | agreement processes | supporting processes | distributed cybersecurity activities |

Table - relationship among standards

The ***AVXDL*** does not attempt to cover either the organization or supplier-related processes because these are managed at the organizational level.

# Philosophy

The creation of secure software is not simply a programming endeavor. It begins with ensuring all team members have an understanding of how software and systems are made secure; requires the additional stages in the design and testing phase; and permeates all typical development practices.

It is impossible to integrate security into a large operational system in a single pass. This is a reality stemming from the lack of security focus within the computing industry as a whole. In order to be effective, security must be seen as emergent property and not an adjunct capability. This stands in contrast to the recent trend toward minimal functional development. For a system to be secure it must be secure by design, not coincidence. Therefore, there are some foundational elements which should be in place prior to implementation.

Given the scope of the problem space, the most appropriate approach entails:

* creation of an [**implementation framework**](#_Implementation_Framework)
* [**continuous improvement**](#_Continuous_Improvement) of the security posture

# Background Material

The ***AVXDL*** is based on methodologies proven in industry as well as standards bodies’ recommendations. These include **MSSDL**, **SSDF** and **ISO 21434**.

## Microsoft SDL (MSSDL)

The archetype for the SDL is the Microsoft SDL (**MSSDL**). It divides the development process into seven phases. These phases form a cycle of ever-improving security posture.

Additional details about MSSDL can be explored [here](#_14._MSSDL_Background).

## NIST SSDF (SSDF)

The NIST Secure Software Development Framework (**SSDF**) provides a more general approach which calls out a number of practices and provides references to the applicable standards. Within each of these are multiple practices and tasks.

The advantage of the SSDF over the MSSDL is that it provides a greater level of specificity and better supports existing international standards. It also calls out practices assumed, but not specified in the MSSDL.

Additional details about SSDF can be explored [here](#_13._SSDF_Background).

## ISO/SAE 21434 (’434)

Road Vehicles - Cybersecurity Engineering (**’434**) is intended to address the cybersecurity aspects of electrical and electronic (E/E) systems within road vehicles. Its goal is to enable organizations to:

* define cybersecurity policies and processes
* manage cybersecurity risk
* foster a cybersecurity culture

Like the MSSDL, the development process is divided into phases.

Unlike MSSDL and SSDF, which are lifecycle-focused, non-domain-specific documents, ’434 is a regulatory-focused, domain-specific (road vehicle E/E systems) work. This can be clearly seen in the ’434 document overview diagram:

Additional details about ISO/SAE 21434 can be explored [here](#_15._ISO/SAE_21434).

## ISO 26262 (’262)

Road vehicles — Functional safety (**’262**) is intended to address the safety aspects of electrical and electronic (E/E) systems within road vehicles. Although not used as a primary source reference for the **AVXDL**, the **AVXDL** can be aligned to it. This allows for easier integration with existing development processes.

## ISO/IEC/IEEE 12207 (’207)

Systems and software engineering – Software life cycle processes (**’207**) provides a set of processes required to systematically implement a development lifecycle. This should be considered the grounding process alignment document.

## Contributions Visualized

These sources come together as follows:

A picture containing diagram

Description automatically generated

Figure - AVXDL sources

**Note:** The red rounded rectangles indicate corresponding ’262 processes.

# Continuous Improvement

The creation of secure software is not simply a programming endeavor. It begins with ensuring all team members have an understanding of how software and systems are made secure; requires the additional stages in the design and testing phase; and permeates all typical development practices.

As a system of systems, an autonomous vehicle will always be subject to variable-rate development. That is to say that each system within the system of systems is developed at a rate which may (and probably does) differ from that of other systems. Being the case, it is highly unlikely that the security profile of the system as a whole will be the same. This motivates the adoption of a process of continual improvement within each of the constituent systems, driving toward an ever-improved overall security profile.

Continuous improvement is not merely *pro forma*, but requisite. Whether we are considering the initial implementation of security or ongoing development, there will be the need to develop and refine the security model. This will need to be done at every level of the system (physical, network, protocol, and application) as well as across all the various sub-systems. Given the time-consuming nature of threat modeling, risk assessment, and threat mitigation; the only practical approach is to apply ever-increasing levels of security. This may manifest as an outside-in approach wherein the external attack surfaces are secured first with the interior system following; or by addressing the fundamental mechanisms by which data is managed within the system; or a combination of the two dependent upon the maturity of the security in place in any given sub-system.

The ***AVXDL*** itself will also be subject to continuous improvement. This may manifest in changes to individual phases and their associated requirements. It may include changes in ownership of various phase requirements. There will always be new tools to consider in implementation of the ***AVXDL***.

In order to track progress of implementation of the ***AVXDL*** within the organization, **ISO 21827** Systems Security Engineering - Capability Maturity Model (**SSE-CMM**) will be used as criteria for evaluation. These levels are refined through application of Cybersecurity Maturity Model Certification (**US DoD CMMC**) level assignments of the requirements called out in ***Protecting Controlled Unclassified Information in Nonfederal Systems and Organizations*** (**NIST SP 800-171**).

An example tracking report is shown [here](#_17._Continuous_Improvement_1).

# Relationship to ISO 21434

As stated in the [**overview**](#_Overview), the ***AVXDL*** is a set of identified processes, requirements for those processes, generated products and their mapping to corresponding ISO 21434 work products. This section will provide more detail about the relationship between the two.

## Compliance versus Conformance

The ***AVXDL*** is designed to enable the organization to comply with the requirements of and enable the creation of products capable of satisfying the specifications for work products called for in ISO 21434. It does not conform to ISO 21434 in that it does not assume that any given certification standard’s structure matches the form of any given organization’s product development lifecycle framework, which encompasses development, safety, security and other needs. Any certification methodology with processes not conforming to the organization’s actual development processes will be error-prone and unsuccessful.

## Product Mapping

The ***AVXDL*** is designed to overlay an ISO 15288 / ISO 12207 lifecycle. As such, roughly 50 products are generated during any given product release’s lifetime. The ***AVXDL*** provides mapping of these, which are the natural outcomes from implementation of cybersecurity best practices, to related ISO 21434. In this way, developmental friction is reduced as there is no expectation that individual contributors from any group (PMO, risk, devops, development, …) be familiar with ISO 21434 and its details. This would, in fact, be seen as a negative as ISO 21434 is only the first of what will become many jurisdictional compliance standards needing to be complied with. By having a best practice lifecycle (***AVXDL***), we are more readily able to embrace compliance with regulatory standards as the appear.

# Hardware-Software Relationship

Upon reading the ***AVXDL***, the question often arises as to why there is no specific reference to the hardware aspects of cybersecurity and how this relates to ISO 21434 work products. This section will provide more detail about the relationship between the two, as well as the implications with respect to **’434**.

## *AVXDL* is about Process

The ***AVXDL*** is at the core a framework supporting a collection of processes implementing a set of requirements. As noted in the [**background** **material**](#_Background_Material), it is built assuming the presence of the **’288** and **’207** standards within the organization. It further presumes that a best practices hardware-software development strategy akin the that described in **’262** (V-model). The phases and their requirements are sufficient to cover both hardware and software.

This is not to say that there is no consideration of hardware within the realm of cybersecurity as applied to the product’s lifecycle. The cybersecurity requirements have explicit provision for hardware-specific requirements. Refer to the secondary document [**Security Requirements Taxonomy**](#avcdl-archive-avcdl-release-2) for additional details.

Any hardware-specific requirements are linked to specific product elements during the requirements phase as set out in the secondary document [**Product-level Security Requirements**](#ref_product_level_security_requirements).

## ISO 21434 Compliance

The ***AVXDL*** is designed enable the production of a supporting case for certification under **’434**. **’434** has no hardware-specific work products or requirements. There is the desire that we be able to show that we apply cybersecurity to both hardware and software. This will be evidenced through application of cybersecurity concepts, goals and requirements using the processes and requirements set out in the ***AVXDL***.

# Implementation Framework

As indicated in the background section, the structure of the **AVXDL** is a fusion of the **MSSDL**, **SSDF**, and **ISO 21434**.

It visualizes as follows:

Timeline

Description automatically generated

Figure - AVXDL phases

The process elements fall into three broad categories:

* **Foundational** process elements (shown in blue) form a foundation for secure development and take place outside the normal development cycle. These may be done in parallel and are subject to refresh as the security landscape changes. The basis for this phase comes from **MSSDL** and **’434**, and the practices from **SSDF**.
  + [foundation](#_Foundation_Phase)
* **Intra-developmental** process elements (shown in green) serve to augment the existing development processes. The phases come from **MSSDL** and **’434**, and the practices from **SSDF**. To a large extent, the framework on which these phase augmentations hang already exists as part of the typical non-security-aware development process.
  + [requirements](#_Requirements_Phase)
  + [design](#_Design_Phase)
  + [implementation](#_Implementation_Phase)
  + [verification](#_Verification_Phase)
  + [release](#_Release_Phase)

**Note:** In addition to its ISO/SAE 21434-supporting requirements, each phase also has an exit gate requirement. There are no ’434 work products in the gate requirements as these are intended to verify the completion of the other requirements within the phase.

* **Post-developmental** phases (shown in yellow) take place once the product has been released.
  + [operation](#_Operation_Phase)
  + [decommissioning](#_Decommissioning_Phase)

It can also be visualized as a cyclic system:

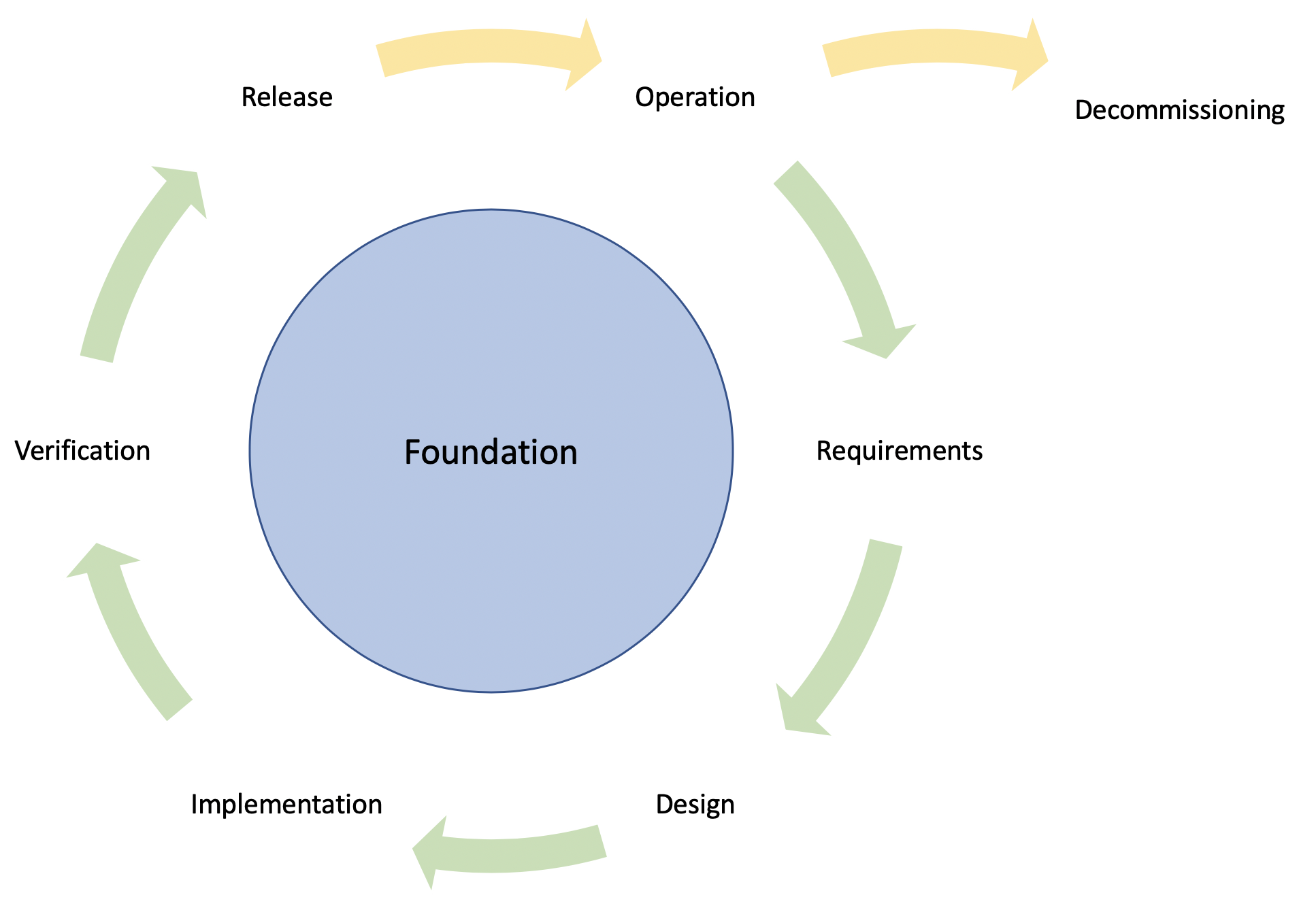


Figure - AVXDL phases cyclic

It is important to note that the AV<X>DL does not mandate an implementation methodology (waterfall, XP, BDD, TDD, scrum, Kanban, spiral, etc.). The phases are dictated by the work product dependencies which are more fully explored in the [AVCDL Product Dependencies](#_18._AVCDL_Product) section.

If a scrum-based agile development methodology is employed, the intra-developmental phases overlay as follows:

Diagram

Description automatically generated

Figure – AV<X>DL-PDCA Phase Requirement Mapping

Since cyclic implementation methodologies use short activity windows (sprints), there is a need to offset the security activities. The following diagram illustrates how the activity cycle unfolds into linear time.

Timeline

Description automatically generated

Figure - Sprint-level Alignment

**Note:** The detailed requirements activity is performed on sprint in advance of design, and design one sprint in advance of implementation. A sprint SP0 is assumed to bootstrap the process.

# Process Phases and Requirements

This following is a summary of the AVXDL phases and their associated requirements.

* [Foundation](#_Foundation_Phase_1)
  + [Foundation-1](#_Training_[AVCDL-Foundation-1]) Training
  + [Foundation-2](#_Roles_and_Responsibilities) Roles and Responsibilities
* [Requirements](#_Requirements_Phase)
  + [Requirements-1](#_Security_Requirements_Definition) Definition of Security Requirements
  + [Requirements-2](#_Requirements_Gate_[AVCDL-Requiremen) Requirements Gate
* [Design](#_Design_Phase)
  + [Design-1](#_Design_Phase) Take Security Requirements and Risk Information into Account During Software Design
  + [Design-5](#_Design_Gate_[AVCDL-Design-5]) Design Gate
* [Implementation](#_Implementation_Phase)
  + [Implementation-9](#_Security_Code_Review) Security Code Review
  + [Implementation-10](#_Implementation_Gate_[AVCDL-Implemen) Implementation Gate
* [Verification](#_Verification_Phase)
  + [Verification-1](#_Fuzz_Testing_[AVCDL-Verification-1]) Fuzz Testing
  + [Verification-5](#_Verification_Gate_[AVCDL-Verificati) Verification Gate
* [Release](#_Release_Phase)
  + [Release-1](#_Final_Security_Review) Final Security Review
  + [Release-3](#_Release_Gate_[AVCDL-Release-3]) Release Gate
* [Operation](#_Operation_Phase)
  + [Operation-1](#_Identify_and_Confirm) Identify and Confirm Vulnerabilities on an Ongoing Basis
  + [Operation-4](#_Secure_Deployment_[AVCDL-Operation-) Secure Deployment
* [Decommissioning](#_Decommissioning_Phase)
  + [Decommissioning-1](#_Apply_Decommissioning_Protocol) Decommissioning Protocol

# 9.1 Foundation Phase

Predecessor: N/A

Successor: [Requirements Phase](#_Requirements_Phase)

These process elements form a foundation for secure development and take place outside the normal development cycle. These may be done in parallel and are subject to refresh as the security landscape changes.

## [[AVXDL-Foundation-1] Training](#_Training_[AVCDL-Foundation-1]) ([SSDF PO.1](#PO_1) / [MSSDL P1](#MS_P1))

This training ensures that the SDL process and its requirements are understood by those interacting with it.

## [[AVXDL-Foundation-2] Roles and Responsibilities](#_Roles_and_Responsibilities) ([SSDF PO.2](#PO_2))

It is critical to the success of any SDL-based project that the roles and responsibilities be defined and assigned prior to the phase to which they apply. These individuals serve as gatekeepers of security issues at the various phase gates.

Foundation phase product dependencies are visualized in Figure 6.

## ISO 21434 Required Work Products

[WP-05-05] Evidence of tool management

[WP-07-01] List of sources for cybersecurity monitoring

# 9.1.1 Training [AVXDL-Foundation-1]

## Owner

**group:** [security](#_11.3_Groups_[security])

**NCWF role:** [Cyber Instructor](#_12.3_Cyber_Instructor)

## Administration

|  |  |  |  |
| --- | --- | --- | --- |
| security | devops | development | risk |
| R | I | I | I |

There should be a general security awareness training covering the motivation for cybersecurity and its relationship to safety.

There are five distinct areas of training (as specified in [MSSDL P1](#MS_P1)):

* Secure design
* Threat modeling
* Secure coding
* Security testing
* Privacy

Aside from the awareness training, the other training classes have different target audiences. They may be presented concurrently. Ideally, they should be presented prior to the phase to which they apply. There should be an annual limited-scope refresher for each.

The overall training sequence is covered in the [**Training Path**](#_19._AVCDL_Training) section.

## Training Provided

**none**

## Phase Requirement Dependencies

**none**

## External Group Product Dependencies

|  |  |
| --- | --- |
| **Group** | **Inputs** |
| Devops | **none** |
| Development | List of programming languages / compilers |
| Risk | **none** |

## AVXDL Products

* [training catalog](#ref_training_catalog)
* [system to track training participation](#ref_system_to_track_training_participati)

## ISO 21434 Required Work Products

**none**

# 9.1.2 Roles and Responsibilities [AVXDL-Foundation-2]

## Owner

**group:** [security](#_11.3_Groups_[security])

**NCWF role:** [Systems Requirements Planner](#_12.9_Systems_Requirements)

## Administration

|  |  |  |  |
| --- | --- | --- | --- |
| security | devops | development | risk |
| R | C | C | - |

**NIST SP 800-181** [***National Initiative for Cybersecurity Education (NICE) Cybersecurity Workforce Framework (NCWF)***] provides an exhaustive breakdown of cybersecurity roles and responsibilities. It provides a common, consistent lexicon that categorizes and describes cybersecurity work. We will draw upon these to establish those needed in support of the ***AVXDL***.

**Note:** Additional information on ***NCWF*** can be found on their [site](https://niccs.us-cert.gov/workforce-development/cyber-security-workforce-framework).

**Note:** There will be tasks and abilities called out for roles in NCWF which are not leveraged. Additionally, there will be areas where there is not a 1-to-1 mapping.

**Note:** The role assignments are shown at the top of each requirement page and collected [here](#_Requirement_Role_Assignments).

## Training Provided

**none**

## Phase Requirement Dependencies

**none**

## External Group Product Dependencies

## none

## AVXDL Products

* [roles and responsibilities document](#_Philosophy)

## ISO 21434 Required Work Products

**none**

# 9.2 Requirements Phase

Predecessor: [Foundation Phase](#_Foundation_Phase_1) or [Operation Phase](#_Operation_Phase)

Successor: [Design Phase](#_Design_Phase)

The requirements phase of development is a reiteration of **AVXDL-EX-4 Definition of Security Requirements** (SSDF PO.4 / MSSDL P2) but with higher resolution. In an Agile-based development process, this is to be expected.

## [[AVXDL-Requirements-1] Definition of Security Requirements](#_Security_Requirements_Definition) ([SSDF PO.4](#PO_4) / [MSSDL P2](#MS_P2))

The requirements are created with consideration of the global security requirements. They provide constraints specific to the work under consideration.

## [[AVXDL-Requirements-2] Requirements Gate](#_Requirements_Gate_[AVCDL-Requiremen) ([MSSDL P3](#MS_P3))

Requirements phase exit is conditional (formally gated) on completion of the SDL process for this phase.

Requirements phase product dependencies are visualized in Figure 7.

## ISO 21434 Required Work Products

[WP-09-01] Item definition

[WP-09-07] Cybersecurity concept

[WP-10-02] Cybersecurity requirements for post-development

# 9.2.1 Security Requirements Definition [AVXDL-Requirements-1]

## Owner

**group:** [security](#_11.3_Groups_[security])

**NCWF role:** [Security Architect](#_12.7_Security_Architect)

## Administration

|  |  |  |  |
| --- | --- | --- | --- |
| security | devops | development | risk |
| R | - | I | - |

Requirements need to both consider the global security requirements and add constraints necessary to the specifics of the work under consideration. As with the global-level requirements called out in [AVXDL-Foundation-4](#_Definition_of_Security), these requirements should be derived using the [**security requirements taxonomy**](#avcdl-archive-avcdl-release-2) in order to expose gaps up-front (prior to threat modeling, attack surface analysis, etc.).

Requirements should be traceable through product operation phase to allow for improvement should deficiencies be discovered.

## Training Provided

**yes**

## Phase Requirement Dependencies

[[AVCDL-Foundation-4]](#_Definition_of_Security) Definition of Security Requirements

## External Group Product Dependencies

|  |  |
| --- | --- |
| **Group** | **Inputs** |
| Devops | **none** |
| Development | High-level design |
| Risk | **none** |

## AVXDL Products

* [product-level security goals](#ref_product_level_security_goals)
* [product-level security requirements](#ref_product_level_security_requirements)

## ISO 21434 Required Work Products

[WP-09-01] Item definition

[WP-09-07] Cybersecurity concept

# 9.2.2 Requirements Gate [AVXDL-Requirements-2]

## Owner

**group:** [security](#_11.3_Groups_[security])

**NCWF role:** [Secure Software Assessor](#_12.6_Secure_Software)

## Administration

|  |  |  |  |
| --- | --- | --- | --- |
| security | devops | development | risk |
| R | - | A | - |

Requirements phase exit is conditional (formally gated) on completion of the SDL process for this phase. The security advisor assigned to the release must certify that the project team has satisfied security requirements for this phase.

## Training Provided

**none**

## Phase Requirement Dependencies

[[AVCDL-Requirements-1]](#_Security_Requirements_Definition) Security Requirements Definition

## External Group Product Dependencies

**none**

## AVXDL Products

* [formal gate signoff attesting to completion of all phase products](#ref_requirements_phase_gate)

## ISO 21434 Required Work Products

**none**

# 9.3 Design Phase

Predecessor: [Requirements Phase](#_Requirements_Phase)

Successor: [Implementation Phase](#_Implementation_Phase)

The changes to the design phase include the incorporation of security requirements and analysis of the design from a security perspective.

## [[AVXDL-Design-1] Apply Security Requirements and Risk Information to Design](#_Apply_Security_Requirements) ([SSDF PW.1](#PW_1) / [MSSDL P4](#MS_P4), [MSSDL P5](#MS_P5))

The design should take into consideration established security requirements and risk information.

## [[AVXDL-Design-2] Security Design Review](#_Security_Design_Review) ([SSDF PW.2](#PW_2))

Help ensure the software will meet the security requirements and satisfactorily address the identified risk information.

## [[AVXDL-Design-5] Design Gate](#_Design_Gate_[AVCDL-Design-5]) ([MSSDL P3](#MS_P3))

Design phase exit is conditional (formally gated) on completion of the SDL process for this phase.

Design phase product dependencies are visualized in Figure 8.

## ISO 21434 Required Work Products

[WP-06-03] Cybersecurity assessment report

[WP-07-X1] Risk treatment basis

# 9.3.1 Apply Security Requirements and Risk Information to Design [AVXDL-Design-1]

## Owner

**group:** [development](#_11.2_Groups_[development])

**NCWF role:** [Software Developer](#_12.8_Software_Developer)

## Administration

|  |  |  |  |
| --- | --- | --- | --- |
| security | devops | development | risk |
| A | - | R | - |

Determine which security requirements the software’s design should meet and determine what security risks the software is likely to face during production operation and how those risks should be mitigated by the software’s design. Addressing security requirements and risks during software design instead of later helps to make software development more efficient.

## Training Provided

**yes**

## Phase Requirement Dependencies

[[AVCDL-Requirements-2]](#_Requirements_Gate_[AVCDL-Requiremen) Requirements Gate

## External Group Product Dependencies

|  |  |
| --- | --- |
| **Group** | **Inputs** |
| Devops | **none** |
| Development | Detailed functional requirements |
| Risk | **none** |

## AVXDL Products

* [design showing security considerations](#ref_design_showing_security_consideratio)

## ISO 21434 Required Work Products

[WP-08-02] Identified assets and cybersecurity properties

[WP-09-08] Verification report of cybersecurity concept

[WP-10-01] Refined cybersecurity specification

# 9.3.2 Security Design Review [AVXDL-Design-2]

## Owner

**group:** [security](#_11.3_Groups_[security])

**NCWF role:** [Systems Requirements Planner](#_12.10_Systems_Security)

## Administration

|  |  |  |  |
| --- | --- | --- | --- |
| security | devops | development | risk |
| R | - | A | C |

Help ensure the software will meet the security requirements and satisfactorily address the identified risk information.

## Training Provided

**yes**

## Phase Requirement Dependencies

[[AVCDL-Design-1]](#_Apply_Security_Requirements) Apply Security Requirements and Risk Information to Design

## External Group Product Dependencies

|  |  |
| --- | --- |
| **Group** | **Inputs** |
| Devops | **none** |
| Development | Element detailed design |
| Risk | **none** |

## AVXDL Products

* [security design review report](#ref_security_design_review_report)

## ISO 21434 Required Work Products

[WP-06-03] Cybersecurity assessment report

# 9.3.5 Design Gate [AVXDL-Design-5]

## Owner

**group:** [security](#_11.3_Groups_[security])

**NCWF role:** [Secure Software Assessor](#_12.6_Secure_Software)

## Administration

|  |  |  |  |
| --- | --- | --- | --- |
| security | devops | development | risk |
| R | - | A | A |

Design phase exit is conditional (formally gated) on completion of the SDL process for this phase. The security advisor assigned to the release must certify that the project team has satisfied security requirements for this phase.

## Training Provided

**none**

## Phase Requirement Dependencies

[[AVCDL-Design-2]](#_Security_Design_Review) Security Design Review

[[AVCDL-Design-3]](#_Attack_Surface_Reduction) Attack Surface Reduction

[[AVCDL-Deisgn-4]](#_Threat_Modeling_[AVCDL-Design-4]) Threat Modeling

## External Group Product Dependencies

|  |  |
| --- | --- |
| **Group** | **Inputs** |
| Devops | **none** |
| Development | **none** |
| Risk | **none** |

## AVXDL Products

* [formal gate signoff attesting to completion of all phase products](#ref_design_phase_gate)

## ISO 21434 Required Work Products

**none**

# 9.4 Implementation Phase

Predecessor: [Design Phase](#_Design_Phase)

Successor: [Verification Phase](#_Verification_Phase)

The implementation phase of development is based on **MSSDL Implementation Phase** (MSSDL P8-10) and **SSDF Prepare Well-Secured Software** (PW.4-7, 9).

## [[AVXDL-Implementation-1] Use Approved Tools](#_Use_Approved_Tools) ([MSSDL P8](#MS_P8))

Generally speaking, development teams should strive to use the latest version of approved tools to take advantage of new security analysis functionality and protections.

## [[AVXDL-Implementation-2] Configure Build Process to Improve Security](#_Configure_Build_Process) ([SSDF PW.6](#PW_6))

## [[AVXDL-Implementation-10] Implementation Gate](#_Implementation_Gate_[AVCDL-Implemen) ([MSSDL P3](#MS_P3))

Implementation phase exit is conditional (formally gated) on completion of the SDL process for this phase.

Implementation phase product dependencies are visualized in Figure 9.

## ISO 21434 Required Work Products

[WP-10-06] Integration and verification reports

[WP-10-08] Software unit design and software unit implementation

# 9.5 Verification Phase

Predecessor: [Implementation Phase](#_Implementation_Phase)

Successor: [Release Phase](#_Release_Phase)

The verification phase of development is based on **MSSDL Verification Phase** (MSSDL P11-3) and **SSDF Produce Well-Secured Software** (SSDF PW.8).

## [[AVXDL-Verification-1] Fuzz Testing](#_Fuzz_Testing_[AVCDL-Verification-1]) ([MSSDL P12](#MS_P12))

Fuzz testing is a specialized form of dynamic analysis used to induce program failure by deliberately introducing malformed or random data to an application.

## [[AVXDL-Verification-2] Penetration Testing](#_Penetration_Testing_[AVCDL-Verifica) ([SSDF PW.8](#PW_8))

Penetration testing identifies vulnerabilities before software is released so they can be corrected before release, which prevents exploitation.

## [[AVXDL-Verification-3] Threat Model Review](#_Threat_Model_Review) ([MSSDL P13](#MS_P13))

The threat models should be reviewed to ensures that any design or implementation changes to the system have been accounted for, and that any new attack vectors created as a result of the changes have been reviewed and mitigated.

## [[AVXDL-Verification-4] Attack Surface Analysis Review](#_Attack_Surface_Analysis) ([MSSDL P13](#MS_P13))

The attack surface analysis should be reviewed to ensures that any design or implementation changes to the system have been accounted for, and that any new attack vectors created as a result of the changes have been reviewed and mitigated.

## [[AVXDL-Verification-5] Verification Gate](#_Verification_Gate_[AVCDL-Verificati) ([MSSDL P3](#MS_P3))

Verification phase exit is conditional (formally gated) on completion of the SDL process for this phase.

Verification phase product dependencies are visualized in Figure 10.

## ISO 21434 Required Work Products

[WP-10-03] Verification report for the refined cybersecurity specification

[WP-10-04] Vulnerability analysis report

[WP-11-02] Validation report

# 9.6 Release Phase

Predecessor: [Verification Phase](#_Verification_Phase)

Successor: [Operation Phase](#_Operation_Phase)

The release phase of development is based on **MSSDL Release Phase** (MSSDL P14-6).

## [[AVXDL-Release-1] Final Security Review](#_Final_Security_Review) ([MSSDL P15](#MS_P15))

The Final Security Review (FSR) is a deliberate examination of all the security activities performed on a software application prior to release.

## [[AVXDL-Release-2] Archive](#_Archive_[AVCDL-Release-2]) ([MSSDL P16](#MS_P16))

All pertinent information and data must be archived to allow for post-release servicing of the software.

## [[AVXDL-Release-3] Release Gate](#_Release_Gate_[AVCDL-Release-3]) ([MSSDL P16](#MS_P16))

Software release is conditional (formally gated) on completion of the SDL process.

Release phase product dependencies are visualized in Figure 11.

## ISO 21434 Required Work Products

[WP-06-04] Release for post-development report

# 9.7 Operation Phase

Predecessor: [Release Phase](#_Release_Phase)

Successor: [Requirements Phase](#_Requirements_Phase) or [Decommissioning Phase](#_Decommissioning_Phase)

The operation phase is based on **SSDF Vulnerability Report Practices** (SSDF RV).

## [[AVXDL-Operation-1] Identify and Confirm Vulnerabilities](#_Identify_and_Confirm) ([SSDF RV.1](#RV_1))

Help ensure vulnerabilities are identified more quickly so they can be remediated more quickly, reducing the window of opportunity for attackers.

## [[AVXDL-Operation-2] Assess and Prioritize the Remediation](#_Assess_and_Prioritize) ([SSDF RV.2](#RV_2))

Help ensure vulnerabilities are remediated as quickly as necessary, reducing the window of opportunity for attackers.

## [[AVXDL-Operation-3] Root Cause](#_Root_Cause_Vulnerabilities) Vulnerabilities ([SSDF RV.3](#RV_3))

Help reduce the frequency of vulnerabilities in the future.

## [[AVXDL-Operation-4] Secure Deployment](#_Secure_Deployment_[AVCDL-Operation-)

Software must be deployed in a secure manner.

Operation phase product dependencies are visualized in Figure 12.

## ISO 21434 Required Work Products

[WP-07-03] Cybersecurity event assessment

[WP-07-04] Vulnerability analysis

[WP-07-05] Rationale for the managed vulnerability

[WP-13-02] Cybersecurity incident response information

# 9.8 Decommissioning Phase

Predecessor: [Operation Phase](#_Operation_Phase)

Successor: N/A

Decommissioning is a part of the lifecycle of an item or component and is considered in the concept and product development phases.

Decommissioning is different from end of support. An organization can end support for an item or component, but that item or component can still function as designed in the field. Both decommissioning and end of support present cybersecurity implications, but those implications are considered separately.

Every product release should include a [decommissioning plan](#_Decommissioning_Plan_[AVCDL-Foundat) containing information as to how to properly dispose of the security-related information constrained within the product.

## [[AVXDL-Decommissioning-1] Apply Decommissioning Protocol](#_Apply_Decommissioning_Protocol)

The decommissioning protocol specified in the decommissioning plan should be applied to the system coming out of service.

Decommissioning phase product dependencies are visualized in Figure 13.

## ISO 21434 Required Work Products

**none**

# 10. Requirement Role Assignments

The following table shows the ***AVXDL*** process requirement role assignments.

|  |  |  |  |
| --- | --- | --- | --- |
| **requirement** | **name** | **group** | **NCWF title** |
| [Foundation-1](#_9.1.1_Training_[AVCDL-Foundation-1]) | Training | [security](#_11.3_Groups_[security]) | [cyber instructor](#_12.3_Cyber_Instructor) |
| [Foundation-2](#_9.1.2_Roles_and) | Roles and Responsibilities | [security](#_11.3_Groups_[security]) | [systems requirements planner](#_12.10_Systems_Security) |
| [Foundation-10](#_9.1.10_Deployment_Plan) | Deployment Plan | [security](#_11.3_Groups_[security]) | [information systems security developer](#_12.4_Information_Systems) |
|  |  |  |  |
| [Requirements-1](#_9.2.1_Security_Requirements) | Definition of Security Requirements | [security](#_11.3_Groups_[security]) | [security architect](#_12.7_Security_Architect) |
| [Requirements-2](#_9.2.2_Requirements_Gate) | Requirements Gate | [security](#_11.3_Groups_[security]) | [secure software assessor](#_12.6_Secure_Software) |
|  |  |  |  |
| [Design-1](#_9.3.1_Apply_Security) | Take Security Requirements and Risk Information into Account During Software Design | [development](#_11.2_Groups_[development]) | [software developer](#_12.8_Systems_Requirements) |
| [Design-5](#_9.3.5_Design_Gate) | Design Gate | [security](#_11.3_Groups_[security]) | [secure software assessor](#_12.6_Secure_Software) |
|  |  |  |  |
| [Implementation-1](#_9.4.1_Use_Approved) | Use Approved Tools | [development](#_11.2_Groups_[development]) | [software developer](#_12.8_Systems_Requirements) |
| [Implementation-9](#_9.4.9_Security_Code) | Security Code Review | [security](#_11.3_Groups_[security]) | [secure software assessor](#_12.6_Secure_Software) |
| [Implementation-10](#_9.4.10_Implementation_Gate) | Implementation Gate | [security](#_11.3_Groups_[security]) | [secure software assessor](#_12.6_Secure_Software) |
|  |  |  |  |
| [Verification-1](#_9.5.1_Fuzz_Testing) | Fuzz Testing | [security](#_11.3_Groups_[security]) | [Vulnerability Assessment Analyst](#_12.11_Vulnerability_Assessment) |
| [Verification-5](#_9.5.5_Verification_Gate) | Verification Gate | [security](#_11.3_Groups_[security]) | [secure software assessor](#_12.6_Secure_Software) |
|  |  |  |  |
| [Release-1](#_9.6.1_Final_Security) | Final Security Review | [security](#_11.3_Groups_[security]) | [secure software assessor](#_12.6_Secure_Software) |
| [Release-2](#_9.6.2_Archive_[AVCDL-Release-2]) | Archive | [devops](#_11.1_Groups_[devops]) | [information systems security developer](#_12.4_Information_Systems) |
| [Release-3](#_9.6.3_Release_Gate) | Release Gate | [security](#_11.3_Groups_[security]) | [secure software assessor](#_12.6_Secure_Software) |
|  |  |  |  |
| [Operation-1](#_9.7.1_Identify_and) | Identify and Confirm Vulnerabilities on an Ongoing Basis | [security](#_11.3_Groups_[security]) | [Cyber Defense Incident Responder](#_12.2_Cyber_Defense) |
| [Operation-2](#_9.7.2_Assess_and) | Assess and Prioritize the Remediation of all Vulnerabilities | [security](#_11.3_Groups_[security]) | [Cyber Defense Forensics Analyst](#_12.1_Cyber_Defense) |
| [Operation-3](#_9.7.3_Root_Cause) | Analyze Vulnerabilities to Identify Their Root Causes | [security](#_11.3_Groups_[security]) | [Cyber Defense Forensics Analyst](#_12.1_Cyber_Defense) |
| [Operation-4](#_9.7.4_Secure_Deployment) | Secure Deployment | [devops](#_11.1_Groups_[devops]) | [information systems security developer](#_12.4_Information_Systems) |
|  |  |  |  |
| [Decommissioning-1](#_9.8.1_Apply_Decommissioning) | Decommissioning Protocol | [devops](#_11.1_Groups_[devops]) | [information systems security developer](#_12.4_Information_Systems) |

Table - requirement role assignments

# 11. Groups

This folder contains documents related to the groups responsible for implementation of the ***AVXDL***.

* [devops](#_11.1_Groups_[devops])
* [development](#_11.2_Groups_[development])
* [security](#_11.3_Groups_[security])

The following shows the mapping of requirements to group:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| AVXDL phase | AVXDL requirement | Title | [security](#_11.3_Groups_[security]) | [devops](#_11.1_Groups_[devops]) | [development](#_11.2_Groups_[development]) | risk |
| [Foundation](#_Foundation_Phase_1) | [Foundation-1](#_Training_[AVCDL-Foundation-1]) | Training | **R** | **I** | **I** | **I** |
| [Foundation-2](#_Roles_and_Responsibilities) | Roles and Responsibilities | **R** | **C** | **C** |  |
| [Foundation-3](#_Toolchain_Support_[AVCDL-Foundation) | Toolchain Support | **C** | **R** | **C** |  |
| [Foundation-4](#_Definition_of_Security) | Definition of Security Requirements | **R** | **I** | **I** |  |
| [Foundation-5](#_Protect_the_Code) | Protect the Code | **C** | **R** |  |  |
| [Foundation-6](#_Deprecate_Unsafe_Functions) | Ensure Release Integrity | **C** | **R** | **C** |  |
| [Foundation-7](#_Incident_Response_Plan) | Incident Response Plan | **R** |  | **C** |  |
| [Foundation-8](#_Philosophy) | Decommissioning Plan | **R** | **C** | **C** |  |
| [Foundation-9](#_Threat_Prioritization_Plan) | Threat Prioritization Plan | **R** |  | **I** | **I** |
| [Foundation-10](#_Deployment_Plan_[AVCDL-Foundation-1) | Deployment Plan | **C** | **R** | **C** |  |
|  |  |  |  |  |  |  |
| [Requirements](#_Requirements_Phase) | [Requirements-1](#_Security_Requirements_Definition) | Definition of Security Requirements | **R** |  | **I** |  |
| [Requirements-2](#_Requirements_Gate_[AVCDL-Requiremen) | Requirements Gate | **R** |  | **A** |  |
|  |  |  |  |  |  |  |
| [Design](#_Design_Phase) | [Design-1](#_Apply_Security_Requirements) | Take Security Requirements and Risk Information into Account During Software Design | **A** |  | **R** |  |
| [Design-2](#_Security_Design_Review) | Review the Software Design to Verify Compliance with Security Requirements and Risk Information | **R** |  | **A** | **C** |
| [Design-3](#_Attack_Surface_Reduction) | Attack Surface Reduction | **R** |  | **A** |  |
| [Design-4](#_Threat_Modeling_[AVCDL-Design-4]) | Threat Modeling | **R** |  | **A** | **A** |
| [Design-5](#_Design_Gate_[AVCDL-Design-5]) | Design Gate | **R** |  | **A** | **A** |
|  |  |  |  |  |  |  |
| [Implementation](#_Implementation_Phase) | [Implementation-1](#_Use_Approved_Tools) | Use Approved Tools | **C** | **C** | **R** |  |
| [Implementation-2](#_Configure_Build_Process) | Configure the Compilation and Build Process to Improve Executable Security | **C** | **R** | **C** |  |
| [Implementation-3](#_Use_Secure_Settings) | Configure the Software to Have Secure Settings by Default | **R** |  | **A** |  |
| [Implementation-4](#_Reuse_Well-Secured_Software) | Reuse Existing, Well-Secured Software When Feasible Instead of Duplicating Functionality | **C** | **I** | **R** |  |
| [Implementation-5](#_Code_Securely_[AVCDL-Implementation) | Create Source Code Adhering to Secure Coding Practice | **C** |  | **R** |  |
| [Implementation-6](#_Deprecate_Unsafe_Functions) | Deprecate Unsafe Functions | **C** |  | **R** |  |
| [Implementation-7](#_Static_Analysis_[AVCDL-Implementati) | Static Analysis | **C** | **R** | **C** |  |
| [Implementation-8](#_Dynamic_Program_Analysis) | Dynamic Program Analysis | **C** |  | **R** |  |
| [Implementation-9](#_Security_Code_Review) | Security Code Review | **R** |  | **C** |  |
| [Implementation-10](#_Implementation_Gate_[AVCDL-Implemen) | Implementation Gate | **R** | **A** | **A** |  |
|  |  |  |  |  |  |  |
| [Verification](#_Verification_Phase) | [Verification-1](#_Fuzz_Testing_[AVCDL-Verification-1]) | Fuzz Testing | **R** | **C** | **C** |  |
| [Verification-2](#_Penetration_Testing_[AVCDL-Verifica) | Penetration Testing | **R** | **C** | **C** |  |
| [Verification-3](#_Threat_Model_Review) | Threat Model Review | **R** |  | **A** | **A** |
| [Verification-4](#_Attack_Surface_Analysis) | Attack Surface Analysis Review | **R** |  | **A** |  |
| [Verification-5](#_Verification_Gate_[AVCDL-Verificati) | Verification Gate | **R** |  | **A** | **A** |
|  |  |  |  |  |  |  |
| [Release](#_Release_Phase) | [Release-1](#_Final_Security_Review) | Final Security Review | **R** | **C** | **C** | **C** |
| [Release-2](#_Archive_[AVCDL-Release-2]) | Archive |  | **R** | **C** |  |
| [Release-3](#_Release_Gate_[AVCDL-Release-3]) | Release Gate | **R** | **A** | **A** | **A** |
|  |  |  |  |  |  |  |
| [Operation](#_Operation_Phase) | [Operation-1](#_Identify_and_Confirm) | Identify and Confirm Vulnerabilities on an Ongoing Basis | **R** |  | **C** |  |
| [Operation-2](#_Assess_and_Prioritize) | Assess and Prioritize the Remediation of all Vulnerabilities | **R** |  | **C** | **C** |
| [Operation-3](#_Root_Cause_Vulnerabilities) | Analyze Vulnerabilities to Identify Their Root Causes | **R** |  | **C** |  |
| [Operation-4](#_Secure_Deployment_[AVCDL-Operation-) | Secure Deployment | **C** | **R** | **C** |  |
|  |  |  |  |  |  |  |
| [Decommissioning](#_Decommissioning_Phase) | [Decommissioning-1](#_Apply_Decommissioning_Protocol) | Apply Decommissioning Protocol | **I** | **R** |  |  |

Table - requirement-group mapping

# 11.1 Groups [devops]

The following process requirements are the responsibility of devops:

|  |  |
| --- | --- |
| Requirement | Description |
| [Foundation-3](#_Toolchain_Support_[AVCDL-Foundation) | Toolchain Support |
| [Foundation-5](#_Protect_the_Code) | Protect the Code |
| [Foundation-6](#_Ensure_Release_Integrity) | Ensure Release Integrity |
| [Foundation-10](#_Deployment_Plan_[AVCDL-Foundation-1) | Deployment Plan |
| [Implementation-2](#_Configure_Build_Process) | Configure the Compilation and Build Process to Improve Executable Security |
| [Implementation-7](#_Static_Analysis_[AVCDL-Implementati) | Static Analysis |
| [Release-2](#_Archive_[AVCDL-Release-2]) | Archive |
| [Operation-4](#_Secure_Deployment_[AVCDL-Operation-) | Secure Deployment |
| [Decommissioning‑1](#_Apply_Decommissioning_Protocol) | Decommissioning Protocol |

Table - devops requirement responsibilities

# 11.2 Groups [development]

The following process requirements are the responsibility of development:

|  |  |
| --- | --- |
| Requirement | Description |
| [Design-1](#_Apply_Security_Requirements) | Take Security Requirements and Risk Information into Account During Software Design |
| [Implementation-1](#_Use_Approved_Tools) | Use Approved Tools |
| [Implementation-4](#_Reuse_Well-Secured_Software) | Reuse Existing, Well-Secured Software When Feasible Instead of Duplicating Functionality |
| [Implementation-5](#_Code_Securely_[AVCDL-Implementation) | Create Source Code Adhering to Secure Coding Practice |
| [Implementation‑6](#_Deprecate_Unsafe_Functions) | Deprecate Unsafe Functions |
| [Implementation-8](#_Dynamic_Program_Analysis) | Dynamic Program Analysis |

Table - development requirement responsibilities

# 11.3 Groups [security]

The following process requirements are the responsibility of security:

|  |  |
| --- | --- |
| Requirement | Description |
| [Foundation-1](#_Training_[AVCDL-Foundation-1]) | Training |
| [Foundation-2](#_Roles_and_Responsibilities) | Roles and Responsibilities |
| [Foundation-4](#_Definition_of_Security) | Definition of Security Requirements |
| [Foundation-7](#_Incident_Response_Plan) | Incident Response Plan |
| [Foundation-8](#_Decommissioning_Plan_[AVCDL-Foundat) | Decommissioning Plan |
| [Foundation-9](#_Threat_Prioritization_Plan) | Threat Prioritization Plan |
| [Requirements-1](#_Security_Requirements_Definition) | Definition of Security Requirements |
| [Requirements-2](#_Requirements_Gate_[AVCDL-Requiremen) | Requirements Gate |
| [Design-2](#_Security_Design_Review) | Review the Software Design to Verify Compliance with Security Requirements and Risk Information |
| [Design-3](#_Attack_Surface_Reduction) | Attack Surface Reduction |
| [Design-4](#_Threat_Modeling_[AVCDL-Design-4]) | Threat Modeling |
| [Design-5](#_Design_Gate_[AVCDL-Design-5]) | Design Gate |
| [Implementation-3](#_Use_Secure_Settings) | Configure the Software to Have Secure Settings by Default |
| [Implementation-9](#_Security_Code_Review) | Security Code Review |
| [Implementation‑10](#_Implementation_Gate_[AVCDL-Implemen) | Implementation Gate |
| [Verification-1](#_Fuzz_Testing_[AVCDL-Verification-1]) | Fuzz Testing |
| [Verification-2](#_Penetration_Testing_[AVCDL-Verifica) | Penetration Testing |
| [Verification-3](#_Threat_Model_Review) | Threat Model Review |
| [Verification-4](#_Attack_Surface_Analysis) | Attack Surface Analysis Review |
| [Verification-5](#_Verification_Gate_[AVCDL-Verificati) | Verification Gate |
| [Release-1](#_Final_Security_Review) | Final Security Review |
| [Release-3](#_Release_Gate_[AVCDL-Release-3]) | Release Gate |
| [Operation-1](#_Identify_and_Confirm) | Identify and Confirm Vulnerabilities on an Ongoing Basis |
| [Operation-2](#_Assess_and_Prioritize) | Assess and Prioritize the Remediation of all Vulnerabilities |
| [Operation-3](#_Root_Cause_Vulnerabilities) | Analyze Vulnerabilities to Identify Their Root Causes |

Table - security requirement responsibilities

# 12. NCWF Roles

**NIST SP 800-181** [***National Initiative for Cybersecurity Education (NICE) Cybersecurity Workforce Framework (NCWF)***] provides an exhaustive breakdown of cybersecurity roles and their associated tasks.

The following NCWF roles have been identified as necessary in support of implementation of ***AVXDL***:

* [Cyber Defense Forensics Analyst](#_12.1_Cyber_Defense)
* [Cyber Defense Incident Responder](#_12.2_Cyber_Defense)

**Note:** There will be tasks and abilities called out for roles in NCWF which are not leveraged. Additionally, there will be areas where there is not a 1-to-1 mapping.

# 12.1 Cyber Defense Forensics Analyst (IN-FOR-002)

## *This material is extracted from the NIST NCWF documentation. It is included here for reference only.*

Analyzes digital evidence and investigates computer security incidents to derive useful information in support of system/network vulnerability mitigation.

|  |  |
| --- | --- |
| ID | Task |
| T0027 | Conduct analysis of log files, evidence, and other information to determine best methods for identifying the perpetrator(s) of a network intrusion. |
| T0036 | Confirm what is known about an intrusion and discover new information, if possible, after identifying intrusion via dynamic analysis. |
| T0048 | Create a forensically sound duplicate of the evidence (i.e., forensic image) that ensures the original evidence is not unintentionally modified, to use for data recovery and analysis processes. This includes, but is not limited to, hard drives, floppy diskettes, CDs, PDAs, mobile phones, GPS, and all tape formats. |
| T0049 | Decrypt seized data using technical means. |
| T0075 | Provide technical summary of findings in accordance with established reporting procedures. |
| T0087 | Ensure that chain of custody is followed for all digital media acquired in accordance with the Federal Rules of Evidence. |
| T0103 | Examine recovered data for information of relevance to the issue at hand. |
| T0113 | Identify digital evidence for examination and analysis in such a way as to avoid unintentional alteration. |
| T0165 | Perform dynamic analysis to boot an ‘image’ of a drive (without necessarily having the original drive) to see the intrusion as the user may have seen it, in a native environment. |
| T0167 | Perform file signature analysis. |
| T0168 | Perform hash comparison against established database. |
| T0172 | Perform real-time forensic analysis (e.g., using Helix in conjunction with LiveView). |
| T0173 | Perform timeline analysis. |
| T0175 | Perform real-time cyber defense incident handling (e.g., forensic collections, intrusion correlation and tracking, threat analysis, and direct system remediation) tasks to support deployable Incident Response Teams (IRTs). |
| T0179 | Perform static media analysis. |
| T0182 | Perform tier 1, 2, and 3 malware analysis. |
| T0190 | Prepare digital media for imaging by ensuring data integrity (e.g., write blockers in accordance with standard operating procedures). |

# 13. SSDF Background Material

The NIST Secure Software Development Framework (**SSDF**) provides a more general approach which calls out a number of practices and provides references to the applicable standards. SSDF’s groups are:

* [Prepare the Organization](#_13.1_Prepare_the) (PO)
* [Protect Software](#_13.2_Protect_Software) (PS)
* [Produce Well-Secured Software](#_13.3_Produce_Well-Secured) (PW)
* [Respond to Vulnerability Reports](#_13.4_Respond_to) (RV)

Within each of these are multiple practices and tasks.

# 13.1 Prepare the Organization (PO) Practices

Predecessor: N/A

Successor: [Protect Software (PS) Practices](#_13.2_Protect_Software)

## *This material is extracted from the NIST SSDF documentation. It is included here for reference only.*

## Define Security Requirements for Software Development (PO.1)

Ensure security requirements for software development are known at all times so they can be taken into account throughout the SDLC, and duplication of effort can be minimized because the requirements information can be collected once and shared. This includes requirements from internal sources, such as the organization’s policies, business objectives, and risk management strategy, and external sources, such as applicable laws and regulations.

## Define Criteria for Software Security Checks (PO.4)

Help ensure the software resulting from the SDLC meets the organization’s expectations by defining criteria for checking the software’s security during development.

**Note:** PO.4 applies to both the foundation and requirements phases

# 13.2 Protect Software (PS) Practices

Predecessor: [Prepare the Organization (PO) Practices](#_13.1_Prepare_the)

Successor: [Produce Well-Secured Software (PW) Practices](#_13.3_Produce_Well-Secured)

## *This material is extracted from the NIST SSDF documentation. It is included here for reference only.*

## Protect All Forms of Code from Unauthorized Access and Tampering (PS.1)

Help prevent unauthorized changes to code, both inadvertent and intentional, which could circumvent or negate the intended security characteristics of the software. For code not intended to be publicly accessible, it helps prevent theft of the software and makes it more difficult for attackers to find vulnerabilities in the software.

**Note:** PS.1 should be in the foundation phase

## Provide a Mechanism for Verifying Software Release Integrity (PS.2)

Help software consumers ensure the software they acquire is legitimate and has not been tampered with.

**Note:** PS.2 should be in the foundation phase

## Archive and Protect Each Software Release (PS.3)

Helps identify, analyze, and eliminate vulnerabilities discovered in the software after release.

**Note:** PS.3 should be in the foundation phase

# 15. ISO/SAE 21434 Background Material

## Note: Summary material regarding ISO/SAE 21434 can be found external to this document.

**Road Vehicles - Cybersecurity Engineering** is intended to address the cybersecurity aspects of electrical and electronic (E/E) systems within road vehicles. Its goal is to enable organizations to:

* define cybersecurity policies and processes
* manage cybersecurity risk
* foster a cybersecurity culture

The following clauses are within the scope of the ***AVCDL***:

* Continuous Cybersecurity Activities (8)
* Concept (9)
* Product Development (10)
* Cybersecurity Validation (11)
* Production (12)
* Operations and Maintenance (13)
* End of Cybersecurity Support and Decommissioning (14)

The following clauses are not within the scope of the ***AVCDL***:

* Cybersecurity Management
  + Organizational Cybersecurity Management (5)
  + Project Dependent Cybersecurity Management (6)
* Distributed Cybersecurity Activities (7)
* Threat Analysis and Risk Assessments Methods (15)

# 16. Reference Documents

This section contains a description of the reference documents relating to the ***AVXDL***.

## 16.1 Standards

1. **Cybersecurity Maturity Model Certification (CMMC)**  
   <https://www.acq.osd.mil/cmmc/docs/CMMC_ModelMain_V1.02_20200318.pdf>
2. **Systems and software engineering - Software life cycle processes**  
   <https://en.wikipedia.org/wiki/ISO/IEC_12207>
3. **Systems and software engineering - System life cycle processes**  
   <https://en.wikipedia.org/wiki/ISO/IEC_15288>
4. **Secure Software Development for Autonomous Vehicles**<https://www.sae.org/standards/content/iso/sae21434.d1/>
5. **Systems Security Engineering - Capability Maturity Model (SSE-CMM)**  
   <https://www.iso.org/standard/44716.html>
6. **Microsoft Security Development Lifecycle (SDL) - simplified implementation**  
   [http://download.microsoft.com/download/F/7/D/F7D6B14F-0149-4FE8-A00F-0B9858404D85/Simplified Implementation of the SDL.doc](http://download.microsoft.com/download/F/7/D/F7D6B14F-0149-4FE8-A00F-0B9858404D85/Simplified%20Implementation%20of%20the%20SDL.doc)
7. **NHTSA Cybersecurity Best Practices for the Safety of Modern Vehicles**  
   <https://www.nhtsa.gov/staticfiles/nvs/pdf/812333_CybersecurityForModernVehicles.pdf>
8. **Guidelines for the Creation of Interoperable Software Identification (SWID) Tags**  
   <https://nvlpubs.nist.gov/nistpubs/ir/2016/NIST.IR.8060.pdf>
9. **Protecting Controlled Unclassified Information in Nonfederal Systems and Organizations**  
   <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-171r2.pdf>
10. **NICE Cybersecurity Workforce Framework (NCWF)**  
    <https://nvlpubs.nist.gov/nistpubs/SpecialPublications/NIST.SP.800-181r1.pdf>
11. **Secure Software Development Framework (SSDF)**  
    <https://nvlpubs.nist.gov/nistpubs/CSWP/NIST.CSWP.04232020.pdf>
12. **Static Analysis Results Interchange Format (SARIF)**  
    <https://docs.oasis-open.org/sarif/sarif/v2.0/csprd02/sarif-v2.0-csprd02.pdf>
13. **Systems Engineering Capability Maturity Model (CMM)**  
    <https://resources.sei.cmu.edu/asset_files/TechnicalReport/1993_005_001_16211.pdf>
14. **Software Package Data Exchange (SPDX®​) Specification**  
    <https://spdx.dev/wp-content/uploads/sites/41/2020/08/SPDX-specification-2-2.pdf>
15. **Proposal for a Recommendation on Cyber Security**  
    <https://unece.org/DAM/trans/doc/2019/wp29grva/ECE-TRANS-WP29-GRVA-2019-02e.pdf>

## Secondary Documents

## All secondary documents as named below are PDFs and are located relative to this file in the following folder:

## ./reference\_documents/secondary\_documents

### General Information

* Ranked / Risked Threat Report [[PDF]](reference_documents/secondary_documents/Ranked%20-%20Risked%20Threat%20Report.pdf)
* Threat Report [[PDF]](reference_documents/secondary_documents/Threat%20Report.pdf)

### [Foundation Phase](#_Philosophy)

* Code Protection Plan [[PDF]](reference_documents/secondary_documents/Code%20Protection%20Plan.pdf)
* Cybersecurity Monitoring Plan [[PDF]](reference_documents/secondary_documents/Cybersecurity%20Monitoring%20Plan.pdf)

### [Requirements Phase](#_Requirements_Phase)

* Product-level Security Goals [[PDF]](reference_documents/secondary_documents/Product-level%20Security%20Goals.pdf)
* Product-level Security Requirements [[PDF]](reference_documents/secondary_documents/Product-level%20Security%20Requirements.pdf)

### [Design Phase](#_Philosophy)

* Attack Surface Analysis Report [[PDF]](reference_documents/secondary_documents/Attack%20Surface%20Analysis%20Report.pdf)
* Design Phase Gate [[PDF]](reference_documents/secondary_documents/Design%20Phase%20Gate.pdf)

### [Implementation Phase](#_Philosophy)

* Build Process Documentation [[PDF]](reference_documents/secondary_documents/Build%20Process%20Documentation.pdf)
* Component / Version - Product / Version Cross-reference Document [[PDF]](reference_documents/secondary_documents/Component%20-%20Version%20-%20Product%20-%20Version%20Cross-reference%20Document.pdf)

### [Verification Phase](#_Philosophy)

* Penetration Testing Report [[PDF]](reference_documents/secondary_documents/Penetration%20Testing%20Report.pdf)
* Updated Attack Surface Analysis [[PDF]](reference_documents/secondary_documents/Updated%20Attack%20Surface%20Analysis.pdf)

### [Release Phase](#_Philosophy)

* Archive Manifest [[PDF]](reference_documents/secondary_documents/Archive%20Manifest.pdf)
* Final Security Review Report [[PDF]](reference_documents/secondary_documents/Final%20Security%20Review%20Report.pdf)

### [Operation Phase](#_Philosophy)

* Cybersecurity Incident Report [[PDF]](reference_documents/secondary_documents/Cybersecurity%20Incident%20Report.pdf)
* Software Deployment Report [[PDF]](reference_documents/secondary_documents/Software%20Deployment%20Report.pdf)

### [Decommissioning Phase](#_Philosophy)

* Decommissioning Report [[PDF]](reference_documents/secondary_documents/Decommissioning%20Report.pdf)

## Working Material

## All working material documents as named below are XSLXs and are located relative to this file in the following folder:

## ./reference\_documents/working\_material

* AVCDL CMMC [[XSLX]](reference_documents/working_material/AVCDL_CMM_CMMC.xlsx)
* AVCDL mappings [[XSLX]](reference_documents/working_material/AVCDL_mappings.xlsx)
* AVCDL roles and responsibilities [[XSLX]](reference_documents/working_material/roles_and_responsibilities.xlsx)
* AVCDL supporting document status [[XSLX]](reference_documents/working_material/AVCDL%20secondary%20document%20status.xlsx)

# 17. Continuous Improvement Progress Summary Example

The following table shows a possible representation for tracking the maturity of the ***AVXDL*** implementation:

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **requirement** | **name** | **state** | **CMM level** | **1** | **2.1** | **2.2** | **2.3** | **2.4** | **3.1** | **3.2** | **3.3** | **4.1** | **4.2** | **5.1** | **5.2** |
| Foundation-1 | Training | partial | 1 | X |  |  |  |  |  |  |  |  |  |  |  |
| Foundation-2 | Roles and Responsibilities | partial | 1 | X |  |  |  |  |  |  |  |  |  |  |  |
| Foundation-3 | Toolchain Support | partial | 1 | X |  |  |  |  |  |  |  |  |  |  |  |
| Foundation-4 | Definition of Security Requirements | partial | 1 | X |  |  |  |  |  |  |  |  |  |  |  |
| Foundation-5 | Protect the Code | yes | 2 | X | X |  |  |  | X |  |  |  |  |  |  |
| Foundation-6 | Ensure Release Integrity | no | 0 |  |  |  |  |  |  |  |  |  |  |  |  |
| Foundation-7 | Incident Response Plan | partial | 1 | X |  |  |  |  |  |  |  |  |  |  |  |
| Foundation-8 | Decommissioning Plan | no | 0 |  |  |  |  |  |  |  |  |  |  |  |  |
| Foundation-9 | Threat Prioritization Plan | partial | 1 |  |  |  |  |  |  |  |  |  |  |  |  |
| Foundation-8 | Deployment Plan | no | 0 |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Requirements-1 | Definition of Security Requirements | partial | 1 | X |  |  |  |  |  |  |  |  |  |  |  |
| Requirements-2 | Requirements Gate | partial | 1 | X |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Design-1 | Consider Security and Risk During Design | partial | 1 | X |  |  |  |  |  |  |  |  |  |  |  |
| Design-2 | Design Review | partial | 1 | X |  |  |  |  |  |  |  |  |  |  |  |
| Design-3 | Attack Surface Reduction | no | 0 |  |  |  |  |  |  |  |  |  |  |  |  |
| Design-4 | Threat Modeling | partial | 1 | X |  |  |  |  |  |  |  |  |  |  |  |
| Design-5 | Design Gate | no | 0 |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Implementation-1 | Use Approved Tools | yes | 2 | X | X |  |  |  | X |  |  |  |  |  |  |
| Implementation-2 | Configure Process for Security | partial | 1 | X |  |  |  |  |  |  |  |  |  |  |  |
| Implementation-3 | Configure Secure Settings by Default | partial | 1 | X |  |  |  |  |  |  |  |  |  |  |  |
| Implementation-4 | Reuse Well-Secured Software | yes | 2 | X | X |  |  |  |  |  |  |  |  |  |  |
| Implementation-5 | Use Secure Coding Practice | partial | 1 | X |  |  |  |  |  |  |  |  |  |  |  |
| Implementation-6 | Deprecate Unsafe Functions | partial | 1 | X |  |  |  |  |  |  |  |  |  |  |  |
| Implementation-7 | Static Analysis | yes | 2 | X |  |  |  |  |  |  |  |  |  |  |  |
| Implementation-8 | Dynamic Program Analysis | partial | 1 | X |  |  |  |  |  |  |  |  |  |  |  |
| Implementation-9 | Security Code Review | partial | 1 | X |  |  |  |  |  |  |  |  |  |  |  |
| Implementation-10 | Implementation Gate | no | 0 |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Verification-1 | Fuzz Testing | partial | 1 | X |  |  |  |  |  |  |  |  |  |  |  |
| Verification-2 | Penetration Testing | yes | 1 | X |  |  |  |  |  |  |  |  |  |  |  |
| Verification-3 | Threat Model Review | no | 0 |  |  |  |  |  |  |  |  |  |  |  |  |
| Verification-4 | Attack Surface Analysis Review | no | 0 |  |  |  |  |  |  |  |  |  |  |  |  |
| Verification-5 | Verification Gate | no | 0 |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Release-1 | Final Security Review | no | 0 |  |  |  |  |  |  |  |  |  |  |  |  |
| Release-2 | Archive | partial | 1 | X |  |  |  |  |  |  |  |  |  |  |  |
| Release-3 | Release Gate | no | 0 |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Response-1 | Identify and Confirm Vulnerabilities | partial | 1 | X |  |  |  |  |  |  |  |  |  |  |  |
| Response-2 | Assess and Prioritize the Remediation | no | 0 |  |  |  |  |  |  |  |  |  |  |  |  |
| Response-3 | Root Cause Vulnerabilities | partial | 1 | X |  |  |  |  |  |  |  |  |  |  |  |
| Response-4 | Secure Deployment | no | 0 |  |  |  |  |  |  |  |  |  |  |  |  |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| Decommissioning-1 | Apply Decommissioning Protocol | no | 0 |  |  |  |  |  |  |  |  |  |  |  |  |

Table - maturity tracking example

# 18. AVXDL Product Dependencies

The following diagrams shows the dependencies between the products from the AVXDL process requirements:

A screenshot of a map

Description automatically generated

Figure - AVXDL product dependencies – [foundation phase](#_Foundation_Phase_1)

A screenshot of a cell phone

Description automatically generated

Figure - AVXDL product dependencies – [requirements phase](#_Requirements_Phase)

A close up of a map

Description automatically generated

Figure - AVXDL product dependencies – [design phase](#_Design_Phase)

A close up of text on a white background

Description automatically generated

Figure - AVXDL product dependencies – [implementation phase](#_Implementation_Phase)

A close up of a map

Description automatically generated

Figure - AVXDL product dependencies – [verification phase](#_Verification_Phase)

A screenshot of a cell phone

Description automatically generated

Figure - AVXDL product dependencies – [release phase](#_Release_Phase)

A close up of a logo

Description automatically generated

Figure - AVXDL product dependencies – [operation phase](#_Operation_Phase)

A screenshot of a cell phone

Description automatically generated

Figure - AVXDL product dependencies – [decommissioning phase](#_Decommissioning_Phase)

# 19. AVCDL Training Path

The following figure shows the dependencies between the training associated with the AVCDL phase requirements. It is presented as a visual reference for assumed prior knowledge.

A picture containing graphical user interface

Description automatically generated

Figure - AVCDL Training Path