|

|  |  |
| --- | --- |
|  |  |
| Projekt | PIS / FIS |
| Dokument-Nr | STMA4000D0001 |
| Index | B |
| Status | Freigegeben |
| Klassifizierung | Intern |
| Dokumenttyp | Plan |
| Anzahl Seiten | 56 |
|  |  |
| SW-Qualitätssicherungsplan |  |
|  |  |
|  |  |
| PIS / FIS Train Software |  |

|  |  |  |  |
| --- | --- | --- | --- |
|  | Organisation / Abt. | Name | Datum / Signatur |
|  |  |  |  |
| Erstellt | Stadler Mannheim GmbH | Häberle |  |
| Geprüft | Stadler Mannheim GmbH | Kasprzik |  |
| Freigegeben | Stadler Mannheim GmbH | Plessow |  |

Inhalt

[1 Einleitung 6](#_Toc206681413)

[1.1 Zweck und Anwendungsbereich 6](#_Toc206681414)

[1.2 Referenzen 7](#_Toc206681415)

[1.2.1 Definition 7](#_Toc206681416)

[1.2.2 Standards und Normen 7](#_Toc206681417)

[1.2.3 Referenzen 7](#_Toc206681418)

[1.3 Abkürzungen und Begriffsdefinitionen 8](#_Toc206681419)

[1.4 Aktualisierung des SW-Qualitätssicherungsplans 8](#_Toc206681420)

[2 Projektorganisation und Verantwortlichkeiten 9](#_Toc206681421)

[2.1 Mitarbeiter-Qualifizierung 9](#_Toc206681422)

[2.2 Gesamtprojekt-Verantwortung 9](#_Toc206681423)

[2.3 Definition der Projektrollen im SW-Projekt 10](#_Toc206681424)

[2.4 Zuteilung der Projektrollen 13](#_Toc206681425)

[2.5 Dokumentenbezogene Verantwortlichkeiten 14](#_Toc206681426)

[3 Lebenszyklus-Modell 16](#_Toc206681427)

[3.1 Lebenszyklusmodell für SW-Entwicklung 16](#_Toc206681428)

[3.1.1 Systementwicklungsphase (extern) 19](#_Toc206681429)

[3.1.2 SW-Planungsphase 19](#_Toc206681430)

[3.1.3 SW-Anforderungsphase 20](#_Toc206681431)

[3.1.4 SW-Architektur- und Entwurfsphase 21](#_Toc206681432)

[3.1.5 SW-Komponentenentwurfsphase 22](#_Toc206681433)

[3.1.6 SW-Implementierungsphase 22](#_Toc206681434)

[3.1.7 SW-Komponententestphase 22](#_Toc206681435)

[3.1.8 SW-Integrationsphase 23](#_Toc206681436)

[3.1.9 SW-Validierungsphase 25](#_Toc206681437)

[3.1.10 SW-Bereitstellungsphase 26](#_Toc206681438)

[3.1.11 SW-Wartungsphase 27](#_Toc206681439)

[3.1.12 SW-Begutachtungsphase (extern) 27](#_Toc206681440)

[3.2 Projektdokumente-Cross-Referenz 28](#_Toc206681441)

[4 SW-Konfigurationsmanagementplan 30](#_Toc206681442)

[4.1 Konfigurationskontrolle 30](#_Toc206681443)

[4.2 Dokumentationsmanagement 30](#_Toc206681444)

[4.2.1 Identifikation/Versionierung von Dokumenten 30](#_Toc206681445)

[4.2.2 Pflege der Softwaredokumentation 31](#_Toc206681446)

[4.2.3 Archivierung der Softwaredokumentation 31](#_Toc206681447)

[4.2.4 Ordnerstruktur 32](#_Toc206681448)

[4.3 SW-Konfigurationsmanagement 33](#_Toc206681449)

[4.3.1 Gesamtsoftware-Versionierung und Änderungsklassen 33](#_Toc206681450)

[4.3.2 Versionierung der Microservices und der Schnittstellen (Entwicklungsversionen) 34](#_Toc206681451)

[4.3.3 Archivierung der Software 35](#_Toc206681452)

[5 SW-Verifikationsplan 36](#_Toc206681453)

[6 SW-Validierungsplan 36](#_Toc206681454)

[6.1 Validierungsprozess 36](#_Toc206681455)

[6.2 Test der Gesamtsoftware 37](#_Toc206681456)

[6.3 SW-Validierungsbericht 39](#_Toc206681457)

[7 SW-Bereitstellungsplan 39](#_Toc206681458)

[8 SW-Wartungsplan 41](#_Toc206681459)

[8.1 Beteiligte Rollen 42](#_Toc206681460)

[8.2 Phase 1: Fahrgastbetrieb (3.0.0) 42](#_Toc206681461)

[8.3 Phase 2: Vorfallmanagement 43](#_Toc206681462)

[8.4 Phase 3: Analyse von Änderungsanfragen 44](#_Toc206681463)

[8.5 Phase 4: Planung & Vorbereitung 45](#_Toc206681464)

[8.6 Phase 5: Implementierung & Validierung 46](#_Toc206681465)

[8.7 Phase 6: Software-Veröffentlichung & Übergabe 48](#_Toc206681466)

[9 Richtlinien, Prozeduren und Werkzeuge 49](#_Toc206681467)

[9.1 Richtlinien und Prozeduren 49](#_Toc206681468)

[9.2 Werkzeuge (Tools) 50](#_Toc206681469)

[10 Anforderungsverfolgung 52](#_Toc206681470)

[11 Spezifikationen für SW externer Lieferanten 53](#_Toc206681471)

[11.1 Firmware 53](#_Toc206681472)

[11.2 HW-Komponenten von externen Lieferanten 54](#_Toc206681473)

[12 Umsetzung des SW-Qualitätssicherungsplans 54](#_Toc206681474)

[13 Freigabe der Entwicklungsergebnisse 54](#_Toc206681475)

[14 Werkzeuge und Techniken 54](#_Toc206681476)

[15 Change-Log (Änderungsverzeichnis) 56](#_Toc206681477)

|  |  |
| --- | --- |
|  | **Wichtige Informationen**  Werden durch gerahmte Kästen hervorgehoben. |

|  |  |
| --- | --- |
|  | **Hinweise**  Werden durch gerahmte Kästen hervorgehoben. |

Tabellenverzeichnis

[Tabelle 1: Standards und Normen 7](#_Toc206681478)

[Tabelle 2: Referenzdokumente 7](#_Toc206681479)

[Tabelle 3: Verantwortlichkeiten der Teams 9](#_Toc206681480)

[Tabelle 4: Definition der Rollen im Projekt 12](#_Toc206681481)

[Tabelle 5: Übersicht der projektspezifischen Rollenzuteilungen 13](#_Toc206681482)

[Tabelle 6: Dokumentenbezogene Dokument-Verantwortlichkeiten 15](#_Toc206681483)

[Tabelle 7: Projektbezogene Dokumente-Cross-Referenz zur Norm 29](#_Toc206681484)

[Tabelle 8: Ordnerstruktur der Ablage 32](#_Toc206681485)

[Tabelle 9: Übersicht SW-Versionierung 33](#_Toc206681486)

[Tabelle 10: Übersicht SW-Änderungsklassen 33](#_Toc206681487)

[Tabelle 11: Übersicht MS-SW-Versionierung 34](#_Toc206681488)

[Tabelle 12: Übersicht SW-Änderungsklassen 34](#_Toc206681489)

[Tabelle 13: Lebenszyklusphasen, Prozeduren und Richtlinien 49](#_Toc206681490)

[Tabelle 14: Werkzeuge (Tools) 51](#_Toc206681491)

[Tabelle 15: Redmine-Ticketstatus 52](#_Toc206681492)

[Tabelle 16: COTS-Softwarekomponenten 53](#_Toc206681493)

[Tabelle 17: Werkzeuge und Techniken 55](#_Toc206681494)

Abbildungsverzeichnis

[Abbildung 1 – Unabhängigkeit der Rollen für Basic Integrity 10](#_Toc206681495)

[Abbildung 2 – Legende Rollen 10](#_Toc206681496)

[Abbildung 3 – Lebenszyklus-Modell (V-Modell) 16](#_Toc206681497)

[Abbildung 4 – Phase 2: Vorfallmanagement 43](#_Toc206681498)

[Abbildung 5 – Phase 3: Analyse von Änderungswünschen 44](#_Toc206681499)

[Abbildung 6 – Phase 4 Planung & Vorbereitung 45](#_Toc206681500)

[Abbildung 7 – Phase 5: Implementierung & Validierung 46](#_Toc206681501)

[Abbildung 8 – Phase 6: SW-Veröffentlichung & Übergabe 48](#_Toc206681502)

# Einleitung

## Zweck und Anwendungsbereich

Dieser SW-Qualitätssicherungsplan hat zum Ziel, alle technischen und organisatorischen Aktivitäten zu identifizieren, zu überwachen und zu kontrollieren, welche erforderlich sind, um nachzuweisen, dass der SW-Entwicklungsprozess die für das Projekt **FIS-Software** geforderten Qualitätsmerkmale erfüllt.

Der SW-Qualitätssicherungsplan

* identifiziert das zu erfüllende Sicherheitsintegritätsniveau
* definiert die organisatorischen Verantwortlichkeiten für die individuellen Aktivitäten/Projektrollen
* beschreibt das Lebenszyklusmodell mit seinen phasenspezifischen Aktivitäten, damit verbundenen Eingangs- und Ausgangsdokumenten und die Erfüllungskriterien der jeweiligen Phasen
* beschreibt die Struktur der zu erstellenden Dokumentation
* definiert die individuellen Verantwortlichkeiten zur Erstellung der Dokumente, sowie die Regeln für dessen Verteilung und Ablage
* beinhaltet den SW-Konfigurationsmanagementplan
* referenziert den SW-Verifikationsplan
* beinhaltet den SW-Validierungsplan
* beinhaltet den SW Bereitstellungsplan
* beinhaltet den SW-Wartungsplan
* definiert die im Projekt einzusetzenden Werkzeuge inkl. zugehöriger Klasse
* definiert die Auswahl der gemäß zu erfüllendem Sicherheitsintegritätsniveau anzuwendenden Methoden und Verfahren.

Gemäß Systemaqnforderung wird für das SW-Projekt FIS **Basisintegrität** gemäß [N\_1] gefordert.

Das Qualitätsmanagementsystem der Stadler Mannheim GmbH ist nach ISO 9001 zertifiziert (siehe [20]).

## Referenzen

### Definition

Referenznummern sind in eckigen Klammern dargestellt. Eine zusätzliche Referenz bildet der SW-Dokumentenplan, welcher alle projekt-relevanten Dokumente beinhaltet. Die Referenznummern entsprechen denen des SW-Dokumentationsplans.

Die referenzierten Dokumente sind nachfolgend ersichtlich.

|  |  |
| --- | --- |
|  | **Generisches Dokument**  Alle Projektdokumente sind in der Projekt-Cross-Referenz-Tabelle im Kap. 3.3 aufgeführt. |

### Standards und Normen

|  |  |  |
| --- | --- | --- |
| **Ref.** | **Identifikation** | **Titel** |
| [N1D] | EN 50657:2017 | Bahnanwendungen – Anwendungen für Schienenfahrzeuge – Software auf Schienenfahrzeugen |

Tabelle 1: Standards und Normen

### Referenzen

|  |  |  |
| --- | --- | --- |
| **Ref.** | **Identifikation** | **Titel** |
| [STMA\_1] | STMA2022D0024 | [Prozessbeschreibung STAMA SW-Entwicklung](https://stama-wiki.stadlerrail.com/doku.php?id=stama:dev-process) |
| [STMA\_2] | 90124014 | ISO 9001 Zertifikat Stadler Mannheim GmbH |
| [STAG\_1] | 240429 | STADLER\_CI-CD-Guidelines |
| [PIS\_1] | STMA4000D0001 | FIS-DID (Dokumentenidentifikation) |
| [PIS\_3] | STMA4000D0003 | Glossar, projektübergreifend |
| [PIS\_4] | STMA4000D0004 | Generische Verifkationschecklisten (VCL) |
| [PIS\_5] | STMA4000D0005 | EN50657\_Bewertung\_Anhang\_A |
| [PIS\_6] | STMA4000D0006 | Programmierrichtlinien |

Tabelle 2: Referenzdokumente

## Abkürzungen und Begriffsdefinitionen

|  |  |
| --- | --- |
|  | **Gemeinsames Glossar**  Die Abkürzungen und Begriffsdefinitionen werden im Glossar [PIS\_3] dokumentübergreifend und projektübergreifend geführt. |

## Aktualisierung des SW-Qualitätssicherungsplans

Der SW-Qualitätssicherungsplan wird bei Änderung oder spätestens nach 24 Monaten überprüft. Nach Freigabe einer Revision ist der Abteilungsleiter für dessen Verteilung/Veröffentlichung an alle im Projekt involvierten Personen verantwortlich.

Freigegebene Revisionen dieses Dokumentes, die nicht mehr gültig sind, werden während der Überprüfung entsprechend im Dokument und Dateinamen gekennzeichnet.

Der generische SW-Qualitätssicherungsplan (QSP) definiert die in [N\_1] genannten Planungsaufgaben, soweit diese projektübergreifend gelten.

Zum jeweiligen Projektstart wird er während der Erstellung des projektspezifischen SW-Qualitätssicherungsplan (PQSP) genutzt.

Eine Anpassung kann auch im Projektfortschritt erforderlich sein.   
Dies bezieht sich speziell auf Änderungen bezüglich:

* Qualitätszielen
* Qualitätssicherungsprozessen
* Verantwortlichkeiten im Qualitätsmanagement
* Verantwortlichkeiten die in diesem Dokument genannt werden
* Sonstige Anpassungen bezüglich der oben beschriebenen Inhalte

Jedes Update wird durch einen neuen Revisionsstand identifiziert und dessen Änderungen im Änderungsverzeichnis beschrieben.

Der SW-Qualitätssicherungsplan behält Gültigkeit, bis das Projekt geschlossen wird (z.B. bis zum Ende der Gewährleistungsperiode).

# Projektorganisation und Verantwortlichkeiten

## Mitarbeiter-Qualifizierung

Sämtliche administrativen Aufgaben, Entwicklungsaktivitäten und Qualitätssicherungsaufgaben werden von qualifizierten Mitarbeitern ausgeführt, die ihre Qualifikation durch entsprechende Ausbildung, berufliche Erfahrung und weitere Schulungsmaßnahmen erlangt haben.

Die Qualifikation der Mitarbeiter wird in einer STAMA-Kompetenzmatrix festgehalten. Die Qualifizierungs- und Schulungsnachweise werden in der Personalabteilung verwahrt und können auf Anfrage eingesehen werden.

## Gesamtprojekt-Verantwortung

Die folgenden Verantwortungsbereiche gelten für die Entwicklung der Gerätesoftware und Integration in den Fahrzeugprojekten:

|  |  |
| --- | --- |
|  | **Benennung der Projektverantwortlichkeiten**  Die projektspezifische Zuordnung erfolgt im PQSP. |

| **Person** | **Funktion** | **Projektrolle** | **Bereich/Abteilung** |
| --- | --- | --- | --- |
| Clarissa  Gosemann | Abteilungsleiterin Projektmanagement | AL PM | Projektmanagement |
| Christian Kasprzik | SW-Engineering Lead | AL SW-Eng | SW-Engineering |
| Andreas  Häberle | Quality-Assurance  Lead | QA-Eng | SW-Engineering |

Tabelle 3: Verantwortlichkeiten der Teams

## Definition der Projektrollen im SW-Projekt

Nachfolgende Tabelle beinhaltet gemäß [N\_1], Anhang B:

* die Benennung der Projektrollen
* die für jede Projektrolle definierten Verantwortlichkeiten
* die hierzu erforderliche Qualifikation zur Ausübung der Rolle

|  |  |
| --- | --- |
|  | **Generell erforderliche Qualifikationen für jede Rolle sind**   * Grundlagentraining IEC 61508 * Basiskenntnisse SW-Versionsmanagement |



Abbildung 1 – Unabhängigkeit der Rollen für Basic Integrity



Abbildung 2 – Legende Rollen

|  |  |
| --- | --- |
|  | **Forderung der Unabhängigkeit**  Aufgrund der im Folgenden beschriebenen personellen Verantwortlichkeiten ist die in [N\_1] geforderte Unabhängigkeit sichergestellt. |

Die dokumentenbezogenen Verantwortlichkeiten sind im Detail im Kapitel 2.5 in der dort befindlichen Tabelle den jeweiligen Rollen zugewiesen. Daher beschränken sich die hier genannten Verantwortlichkeiten auf auszuführende Aktivitäten.

| **Rolle (Benennung FIS)** | **Abkürzung** | **Verantwortlichkeit** |
| --- | --- | --- |
| Projektmanager (TPM) | PM | 1) Verantwortlich für Projektplanung   (Bereitstellungen, Terminplanung,   Planung der Arbeitspakete).  2) Verantwortlich für die Definition des Projektteams   gemäß den Vorgaben aus [N\_1], Kapitel 5.1.  *unterstützt durch SW-Eng-Lead*  3) Überwachung und Kontrolle der Ausführung   der definierten Arbeitspakete (inklusive Testergebnisse)  4) Prüfung auf Konformität zum Stadler QMS  *unterstützt durch das QA-Team*  5) Freigabe und Verteilen der Projektdokumente  6) Verantwortlich für SW-Bereitstellung   inkl. Begleitdokumentation  7) Verwaltung der erforderlichen Werkzeuge   zur Nutzung durch das Projektteam  *unterstützt durch das PIT-Team* |
| Anforderungs- manager (TPM) | RQM | 1) Verantwortlich für die Top-Level-Anforderungsverfolgung  2) Inhaltliche Prüfung der Gesamt-SW-Testspezifikation  *unterstützt durch das QA-Team*  3) Definition der externen Schnittstellen |
| Entwerfer (SW-Architekt) | DES | 1) Auswahl der benötigten Werkzeuge  2) Entwurf der SW-Architektur |
| Implementierer (SW-Lead-Dev  und SW-Dev-Team) | IMP | 1) Programmierung der SW  2) Bereitstellung und Prüfung  der SW-Schnittstellendokumentation 3) Definition und Entwurf der Komponententests  und der SW/ SW-Integrationstests 4) Verantwortlich für die SW-Erstellung und   SW-Quellcodeverifikation   (*ausgeführt* *durch 2. Person oder Lead-Developer)* |
| Tester  Validierungstests (QA-Tester) | TST | 1) Verantwortlich für die Ausführung der Validierungstests  2) Verantwortlich für die Entwicklung der   HW- bzw. SW-Testumgebung bzw. des Fahrzeugmodells |
| Integrator (QA-Tester) | INT | 1) Verantwortlich für die Ausführung der Integrationstests  2) Verantwortlich für die Entwicklung der HW- bzw. SW-Testumgebung |
| Verifizierer (QA-Lead) | VER | 1) Durchführung der lebenszyklusphasen-spezifischen   Verifikationsschritte und Erstellung der jeweiligen   Verifikationsberichte (inkl. der Phase SW-Planung) |
| Validierer (QA-Lead) | VAL | 1) Durchführung der SW-Validierungsaktivitäten  2) Prüfung aller phasen-spezifischen Verifikationsberichte  3) Verantwortlich für den SW-Validierungsbericht |
| Konfigurations- manager (TPL) | CFM | 1) Verantwortlich für das Aufsetzen, die Pflege und   die Kontrolle des Konfigurationsmanagements   (Dokumentation, SW)  2) Verantwortlich für die Themen IT-Sicherheit und   Archivierung  *unterstützt durch das IT-Team und den LISO*  3) Änderungsbewertung im Wartungsprozess |

Tabelle 4: Definition der Rollen im Projekt

## Zuteilung der Projektrollen

Für das FIS-Projekt sind bezüglich der Unabhängigkeit der Rollen die Vorgaben für **Basisintegrität** aus der Norm [N\_1] anzuwenden.

Die nachfolgende Tabelle beschreibt die Zuteilung der Rollen auf die Projektressourcen, unter Beachtung bzw. Einhaltung der personellen Unabhängigkeiten gemäß des für das Projekt geltenden SIL-Niveaus. Die unten dargestellte Matrix erfüllt die Anforderungen der Unabhängigkeit gemäß [N\_1]

|  |  |
| --- | --- |
|  | **Bereiche mit Besetzung derselben Person**  Zwei unabhängige Bereiche [RQM-DES-IMP] und [TST-INT-VER-VAL]  Der PM kann ebenfalls eine Rolle in einer der Gruppen ausüben. |

|  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **Name** | **Projektrolle** | | | | | | | | | | **Bereich** |
|  | PM | RQM | DES | IMP | INT | TST | VER | VAL | CFM | ASR |  |
| Clarissa Gosemann  *delegiert siehe PQSP* | X |  |  |  |  |  |  |  | X |  | Technische Projektleitung  und Produktmanagement |
| Andreas Häberle *delegiert siehe PQSP* |  |  |  |  |  |  | X | X |  |  | Safety und Prozesse |
| Sebastian Subbici  (Product Owner) |  | X |  |  |  |  |  |  | X |  |  |
| SW-Architekt *siehe PQSP* |  |  | X |  |  |  |  |  |  |  |  |
| SW-Dev  *siehe PQSP* |  |  |  | X |  |  |  |  |  |  |  |
| QA-Tester  *siehe PQSP* |  |  |  |  | X | X |  |  |  |  | Fahrgastinformationssysteme |
| --- |  |  |  |  |  |  |  |  |  | X | Nicht notwendig für FIS. |

Tabelle 5: Übersicht der projektspezifischen Rollenzuteilungen

|  |  |
| --- | --- |
|  | **Rolle ASR**  Für Basisintegrität wird die Rolle ASR (Assessor) nicht belegt.  Der Prozess wird in Zusammenarbeit mit dem Qualitätsbeauftragten von STAMA definiert und durch interne Audits regelmäßig überprüft. |

## Dokumentenbezogene Verantwortlichkeiten

Nachfolgende Tabelle beschreibt die dokumentenbezogenen Verantwortlichkeiten (gemäß [N\_1], Anhang A, Tabelle A.1 bzw. Anhang C, Tabelle C.1). Wo sinnvoll, können normativ definierte Dokumente zusammengefasst werden, dies ist in der Spalte „Kommentar“ ersichtlich.

Die technische Dokumentenprüfung erfolgt nach 4-Augen-Prinzip in internen Reviews.

In der folgenden Tabelle sind nur die in [N\_1] sowie die laut STAMA Entwicklungsprozess [STMA\_1] geforderten Dokumente enthalten. Die Nummerierung bleibt erhalten.

| **Nr** | **Dokument** | **Autor** | **Review** | **Freigabe** | **Kommentar** |
| --- | --- | --- | --- | --- | --- |
| 1 | SW-Qualitätssicherungsplan | VER | CFM | AL |  |
|  | SW-PQSP | PM | VER | AL |  |
| 3 | SW-Konfigurationsmanagementplan | CFM | VAL | PM | In 1 enthalten |
| 4 | SW-Verifikationsplan | VER | RQM | PM |  |
| 5 | SW-Validierungsplan | VAL | RQM | PM | In 1 enthalten |
| 2 | SW-Planungsphasenverifikationsbericht | VER | CFM | PM | In 1 enthalten |
| 6 | SW-Anforderungsspezifikation | RQM | DES | PM |  |
| 7 | Gesamt-SW-Testspezifikation | TST | VER | PM |  |
| 8 | SW-Anforderungsverifikationsbericht | VER | RQM | PM |  |
| 9 | SW-Architekturspezifikation | DES | VER | PM |  |
| 10 | SW-Entwurfsspezifikation | RQM | VER | PM | Nicht für BI |
| 11 | SW-Schnittstellenspezifikation | RQM | VER | PM | Async API |
| 12 | SW-Integrationstestspezifikation | IMP | INT | PM | In 7 enthalten |
| 13 | HW/ SW-Integrationstestspezifikation | INT | VER | PM | In 7 enthalten |
| 14 | SW-Architektur- und Entwurfsverifikationsbericht | VER | DES | PM |  |
| 18 | SW Quellcode | IMP | n.a. | n.a. | *Kein Dokument* |
| 21 | SW-Integrationstestbericht | INT | VER | PM | In 24 enthalten |
| 22 | HW/ SW-Integrationstestbericht | INT | VER | PM | In 24 enthalten |
| 22a | Schnittstellen-Integrationstestbericht | INT | VER | PM | In 24 enthalten |
| 24 | Gesamt-SW-Testbericht | TST | VER | PM |  |
| 25 | SW-Validierungsbericht | VAL | VER | PM | Nicht in BI |
| 27 | Freigabemitteilung (Release Note) | DES | VER | PM |  |
| 36 | SW-Bereitstellungsplan | VER | RQM | PM | In 1 enthalten |
| 37 | SW-Bereitstellungshandbuch | IMP | VER | PM | In 27 enthalten |
| 38 | Freigabemitteilung (Release Note) | DES | VER | PM | s.o. Nr. 27 |
| 41 | SW-Wartungsplan | VER | RQM | PM | In 1 enthalten |
| 42 | SW-Änderungsaufzeichnung | RQM | VER | PM | (Jira),  referenziert in 27 |
| 43 | SW-Wartungsaufzeichnung | IMP | VER | PM | (Jira),  referenziert in 27 |
| 44 | SW-Wartungsverifikationsbericht | VER | VAL | PM | *Wird von QMB gefordert.* |

Tabelle 6: Dokumentenbezogene Dokument-Verantwortlichkeiten

# Lebenszyklus-Modell

## Lebenszyklusmodell für SW-Entwicklung

Die folgende Abbildung zeigt das Lebenszyklus-Modell für SW-Entwicklung. Gemäß Tabelle 9 werden für Basisintegrität nicht alle in der Abbildung genannte Dokumente erzeugt.



Abbildung 3 – Lebenszyklus-Modell (V-Modell)

**Erläuterung des Diagramms:**

Das Lebenszyklus-Modell basiert auf [N\_1] Kapitel 5.3.

Der Dokumentationsumfang basiert auf der Kategorie HR („Highly Recommended“) in [N\_1] Tabelle A.1 für Software.

Alle Systemanforderungen werden entsprechend [N\_1] vor der Software-Entwicklung spezifiziert. Alle Anforderungen werden eindeutig identifiziert.

Alle Planungsdokumente müssen zum Start der Entwicklung der SW-Anforderungsspezifikation verfügbar sein.

Der SW-Konfigurationsmanagementplan, der SW-Validierungsplan, der SW-Bereitstellungsplan und der SW-Wartungsplan sind Bestandteil des SW-Qualitätssicherungsplans.

Der SW-Dokumentationsplan enthält die aktuellen Dokumente inkl. aktueller Revisionsstände und weiterer Informationen wie Identifikationsnummer, Status und Phasenbezug. Er wird zum Projektstart aufgesetzt und kontinuierlich gepflegt.

Der Übergang von einer Entwicklungsphase in die nächste unterliegt grundsätzlich der Verantwortung des Verifizierers, nach Prüfung und Freigabe der betreffenden Ausgangsdokumente.   
Der Verifizierer muss im jeweiligen phasenspezifischen Verifikationsbericht eine Aussage treffen, ob die essentiellen Anforderungen der Phase erfüllt wurden und ob die erzeugten Ausgangsdokumente für die nachfolgenden Phasen geeignet sind (dokumentiert über den jeweiligen phasenspezifischen Verifikationsbericht).  
Der Validierer muss im abschließenden SW-Validierungsbericht eine Aussage treffen, ob die SW für den Betrieb freigegeben werden kann (gemäß [N\_1], Kap. 6.3.4.16).

Die folgenden Unterkapitel beschreiben für jede Lebenszyklus-Phase im Detail:

* Die benötigten Eingangsdokumente
* Die auszuführenden Aktivitäten
* Die zu generierenden Ausgangsdokumente

Falls sinnvoll, können zusätzliche Dokumente in jeder Phase verwendet/ referenziert werden, auch wenn sie in den nachfolgenden Unterkapiteln nicht benannt sind.

|  |  |
| --- | --- |
|  | **Vollständige Anforderungsverfolgung**  Die Anforderungen müssen über den gesamten Projektverlauf verfolgbar sein von Kundenanforderung oder „Open Point Item“ über die internen Software-Anforderungen, Testfall und Testbericht. |

### Systementwicklungsphase (extern)

#### Eingangsdokumente

Extern festgelegt

#### Aktivität

Extern festgelegt

Grundsätzlich werden in dieser Phase die für das SW-Projekt relevanten Systemanforderungen definiert und dem SW-Projekt zugewiesen.

#### Ausgangsdokumente

1. System-Anforderungsspezifikation   
   inkl. Definition des Sicherheitsanforderungsniveaus

### SW-Planungsphase

#### Eingangsdokumente

1. System-Anforderungsspezifikation
2. Anzuwendende Programmierrichtlinien

#### Aktivität

Während der Planungsphase werden neben dem SW-Qualitätssicherungsplan (dieses Dokument) weitere Planungsdokumente erstellt und der Projektablauf definiert. Die Planungsdokumente werden im Verlauf des Software-Lebenszyklus’ entsprechend mit Details versehen und aktualisiert.

Der SW-Konfigurationsmanagementplan und der SW-Validierungsplan sind Bestandteile dieses Dokuments, sie werden inhaltlich vom CFM bzw. VAL erstellt.

Der SW-Verifikationsplan wird vom VER erstellt.

Die Vollständigkeit und Konsistenz dieser Planungsdokumente wird vom VER geprüft und bestätigt (gemäß [N\_1], Kap. 6.5.4.8 a) und b)), sowie dokumentiert im SW-Planungsphasen-Verifikationsbericht.

Maßnahmen bzw. Techniken der Tabellen nach [N\_1] Anhang A werden im Anhang A des vorliegenden SW-Qualitätssicherungsplans aufgelistet. Die mit M (mandatory) bzw. HR (highly recommended) markierten Techniken/ Maßnahmen sind jeweils gemäß dem geforderten Sicherheitsintegritätsniveau anzuwenden. Abweichungen sind, sofern zulässig, zu begründen.

#### Ausgangsdokumente

1. SW-Qualitätssicherungsplan   
   *mit integrierten weiteren Planungsdokumenten*
2. SW-Verifikationsplan
3. SW-Planungsphasen-Verifikationsbericht

### SW-Anforderungsphase

#### Eingangsdokumente

1. System-Anforderungsspezifikation
2. Alle Planungsdokumente

#### Aktivität

In der SW-Anforderungsphase wird die SW-Anforderungsspezifikation erstellt, abgeleitet aus den in redmine erstellten SW-relevanten FIS-Systemanforderungen.

Eine Gesamt-SW-Testspezifikation als Basis der auszuführenden Validierungstests wird nicht erstellt, die Testspezifikation repräsentiert sich als Sammlung dedizierter Testfall-Objekte (Python-Skripte), die über GitLab verwaltet werden.

Die SW-Anforderungsverifikation prüft die erstellten Dokumente auf formale und inhaltliche Aspekte und dokumentiert dies im SW-Anforderungsverifikationsbericht.

Anzuwendende Maßnahmen bzw. Techniken gemäß [N\_1] Tabelle A.2 bzw. Tabelle A.7 werden im Anhang A dieses Dokuments definiert.

#### Ausgangsdokumente

1. SW-Anforderungsspezifikation
2. Gesamtsoftwaretestspezifikation
3. SW-Anforderungsverifikationsbericht

### SW-Architektur- und Entwurfsphase

#### Eingangsdokumente

1. SW-Anforderungsspezifikation
2. SW-Verifikationsplan (benötigt zur phasenspezifischen Verifikation)

#### Aktivität

In der SW-Architektur- und Entwurfsphase werden die SW-Anforderungen aufgenommen und auf deren Basis ein Architekturmodell entwickelt. Im SW-Entwurf werden dann die SW-Anforderungen auf die diversen SW-Komponenten zugewiesen und die FIS-Systemschnittstellen definiert. Alle genannten Themen werden im FIS-Projekt in einer SW-Architektur- und Designspezifikation aufgenommen.

Eine SW-Integrationstestspezifikation sowie eine SW/HW-Integrationstest-spezifikation als Basis der auszuführenden Integrationstests werden nicht erstellt, die Testspezifikationen repräsentiert sich als Sammlung dedizierter Testfall-Objekte (Python-Skripte), die über GitLab verwaltet werden und Bestandteil des Gesamt-SW-Tests sind.

Die SW-Architektur- und Entwurfsverifikation prüft die erstellten Dokumente auf formale und inhaltliche Aspekte und dokumentiert dies im SW-Architektur- und Entwurfsverifikationsbericht.

In dieser Phase wird die Programmierumgebung festgelegt. Zur Softwareentwicklung sind die Codier-Richtlinien und Entwurfs- und Codierstandards gemäß Programmierrichtlinien [PIS\_6] anzuwenden. Diese beinhalten zusätzliche Vorgaben zur Quellcode Dokumentation.

Anzuwendende Maßnahmen bzw. Techniken gemäß [N\_1] Tabelle A.3 bzw. Tabelle A.6 werden im Anhang A dieses Dokuments definiert.

#### Ausgangsdokumente

1. SW-Architektur/Entwurfsspezifikation (inkl. Schnittstellenspezifikation)
2. SW-Architektur- und Entwurfsverifikationsbericht

### SW-Komponentenentwurfsphase

Diese Phase ist integriert in der o.g. Architektur- und Entwurfsphase. Es werden hierzu keine dedizierten Dokumente erstellt.

### SW-Implementierungsphase

#### Eingangsdokumente

1. SW-Anforderungsspezifikation
2. SW-Architektur/ Entwurfsspezifikation
3. Programmierrichtlinien

#### Aktivität

In dieser Phase wird auf Basis der SW-Anforderungen und der Architektur- und Entwurfsspezifikation die SW implementiert.

Eine SW-Dokumentation wird nicht explizit erzeugt, die Quelldateien enthalten im Modulkopf jeweils entsprechende Angaben zur Beschreibung der implementierten Funktionen.

#### Ausgangsdokumente

1. Quellcode und unterstützende Dokumentation   
   (z.B. Kompilierreports, Variablenliste, etc.)
2. Softwarebeschreibung, interne Schnittstellen

### SW-Komponententestphase

#### Eingangsdokumente

1. SW-Anforderungsspezifikation
2. SW-Architektur/Entwurfsspezifikation
3. Programmierrichtlinien
4. Software Quellcode
5. Softwarebeschreibung, interne Schnittstellen

Diese Phase findet eng verzahnt mit der SW-Implementierungsphase statt. Die implementierten SW-Module werden einer Code-Prüfung ,hinsichtlich der Einhaltung der im Projekt vorhandenen Programmierrichtlinien unterzogen, sowie auf Korrektheit der Logik mittels Code-Review geprüft. Die Ergebnisse werden im Quellcodeverifikationsbericht oder über die Ausgaben der automatischen Werkzeuge der Quellcodeprüfung dokumentiert.

Es werden auf Komponentenebene keine sonstigen Tests (Black/ White-Box-Tests) ausgeführt (gemäß Basisintegrität nicht gefordert).

|  |  |
| --- | --- |
|  | **Quellcodeverifikationsbericht**  Die konkrete Umsetzung des Quellcodeverifikationsberichts, wird im PQSP definiert. |

#### Ausgangsdokumente

1. Quellcodeverifikationsbericht  
   mit Prüfung der Standards für die Implementierung

### SW-Integrationsphase

|  |  |
| --- | --- |
|  | **Integrationstests von hardwarenaher Software / Firmware**  Bei hardwarenaher Software / Firmware findet diese Phase eng verzahnt mit der SW-Validierung statt. Die zugehörigen Tests sind Bestandteil des Gesamtsystems. |

#### Eingangsdokumente

1. Ergebnisse aus dem Generierungsprozess (Download-File, Log-Files etc.)
2. Systemtest ggf. Python-Skripte (Bestandteil des Validierungstests)
3. SW-Verifikationsplan (benötigt zur phasenspezifischen Verifikation)

#### Aktivität

In dieser Phase findet die SW-Integration bzw. die SW/ HW-Integration statt. Die diversen SW-Komponenten werden zu einer Gesamtsoftware integriert, im Anschluss findet ein SW/ HW-Integrationstest statt. Die Integrationstests sind im FIS-Projekt ein Bestandteil der SW-Validierungstests (siehe folgendes Kapitel).

Die SW-Integrationsverifikation prüft die erstellten Integrationstestergebnisse auf formale und inhaltliche Aspekte und dokumentiert diese als Teil des Gesamtsoftwaretestberichts.

#### Ausgangsdokumente

1. Integrationstestergebnisse (Teil des Gesamtsoftwaretests)
2. Integrationsverifikationsbericht als Teil des Gesamtsoftwaretestberichts.

### SW-Validierungsphase

#### Eingangsdokumente

1. System-Anforderungsspezifikation [10]
2. SW-Anforderungsspezifikation
3. Gesamtsoftwaretestspezifikation
4. SW-Verifikationsplan
5. SW-Validierungsplan
6. gesamte HW- und SW-Dokumentation, einschließlich der phasenspezifischen Verifikationsberichte

#### Aktivität

In dieser Phase findet der Gesamtsoftwaretest (inkl. Integrationstest) auf Basis der vorliegenden Gesamtsoftwaretestspezifikation statt.

Ziel dieser Phase ist es, die integrierte Soft- und Hardware zu testen, um die Einhaltung der Software-Anforderungsspezifikation mit besonderem Augenmerk auf die Funktionalität sicherzustellen und zu prüfen, ob sie für den beabsichtigten Einsatz geeignet ist.

Als abschließendes Dokument wird der SW-Validierungsbericht erstellt, der alle Verifikations- und Validierungsergebnisse zusammenfasst und deren Ergebnisse bewertet.

Die Release Note wird erstellt, wobei für Update-Versionen jeweils immer auch die Änderungen zur Vorgängerversion aufgeführt werden, um eine nachvollziehbare Änderungsdokumentation aufweisen zu können.

Der VAL ist berechtigt, die Freigabe von Software zu verhindern, wenn die Validierungskriterien nicht erfüllt sind, oder gegebenenfalls zusätzliche Überprüfungen, Analysen und Tests anzufordern.

#### Ausgangsdokumente

1. Gesamtsoftwaretestbericht
2. SW-Validierungsbericht
3. SW Release Note (inkl. Bereitstellungshandbuch)

### SW-Bereitstellungsphase

#### Eingangsdokumente

1. SW-Bereitstellungsplan (siehe Kap. 7)
2. Alle relevanten Dokumente aus den vorangegangenen Phasen.

#### Aktivität

Die SW-Bereitstellung definiert sich gemäß SW-Bereitstellungsplan in Kap. 7 und wird mit jedem SW-Release ausgeführt.

Die erzeugten Ausgangsdokumente werden unter Verantwortung des VER auf inhaltliche und formale Aspekte hin geprüft. Diese Prüfung ist Bestandteil der Dokumentenprüfung, ein dedizierter Verifikationsbericht wird nicht erstellt.

#### Ausgangsdokumente

1. SW Release Note (bereits in der Validierungsphase erstellt)

### SW-Wartungsphase

#### Eingangsdokumente

1. Software-Wartungsplan (siehe Kapitel 8)
2. Alle maßgeblichen Entwurfs-, Entwicklungs- und Analyse-Dokumente/ -Ergebnisse sowie Wartungsaufzeichnungen.

#### Aktivität

Der SW-Wartungsprozess wie in Kapitel 8 beschrieben startet mit Freigabe der Erstversion und wird in jeder nachfolgenden Wartungsphase befolgt.

Die Wartungsaufzeichnungen werden mit dem im PQSP definierten Werkzeug geführt.

Die SW Release Note dokumentiert die Änderungen gegenüber der zuvor freigegebenen Software, sowie über bekannte Probleme und ggfs. betriebliche Einschränkungen.

Die erzeugten Ausgangsdokumente werden unter Verantwortung des VER auf inhaltliche und formale Aspekte hin geprüft. Diese Prüfung ist Bestandteil der Dokumentenprüfung, ein dedizierter Verifikationsbericht wird nicht erstellt.

Anzuwendende Maßnahmen bzw. Techniken gemäß [N\_1] Tabelle A.10 werden im Anhang A dieses Dokuments definiert.

|  |  |
| --- | --- |
|  | **Wartungsphase**  Die Wartungsphase beginnt nach der erfolgreichen Typtestphase, bzw. dem Start des Fahrgastbetriebes. |

#### Ausgangsdokumente

1. SW Release Note (bereits in der Validierungsphase erstellt)
2. SW-Wartungsaufzeichnungen (Ticketsystem)
3. SW-Änderungsaufzeichnungen (GitLab, dokumentarisch Teil der SW-Release Note)

### SW-Begutachtungsphase (extern)

Diese Phase entfällt für Basisintegrität.

## Projektdokumente-Cross-Referenz

Die Anwendung der Norm [N\_1] (Abschnittstabelle A.1) auf das Projekt wird durch eine Cross-Referenz-Tabelle umgesetzt. Projektspezifische Dokumente sind dabei den jeweiligen Entwicklungsphasen bzw. den dort normativ gelisteten Dokumenten zugeordnet. Die Ref.-Nummer entspricht der Festlegung im Dokumentationsplan.

| **Ref.** | **Dokumentbezeichnung gemäß [N\_1]** | **Projekt Cross-Referenz-Dokument** |
| --- | --- | --- |
| [1] | SW-Planungsphase | |
| [10] | SW-Qualitätssicherungsplan | STMA4000D0001 SW-Qualitätssicherungsplan |
| [11] | SW-Konfigurationsmanagementplan | in [10] enthalten (Kap. 4) |
| [12] | SW-Verifikationsplan | STMA4000D004 SW-Verifikationsplan |
| [13] | SW-Validierungsplan | in [10] enthalten (Kap. 6) |
| [14] | SW-Wartungsplan | in [10] enthalten (Kap. 8) |
| [19] | n.a. | SW-Planungsphasenverifikationsbericht |
| [2] | SW-Anforderungsphase | |
| [20] | SW-Anforderungsspezifikation | Siehe PQSP |
| [21] | SW-Gesamtsoftwaretestspezifikation | Siehe PQSP |
| [29] | SW-Anforderungsverifikationsbericht | Siehe PQSP |
| [3] | SW-Architektur- und Entwurfsphase | |
| [30] | SW-Architekturspezifikation | Siehe PQSP |
| [31] | SW-Entwurfsspezifikation | in [30] enthalten |
| [32] | SW-Schnittstellenspezifikation | in [30] enthalten |
| [33] | SW-Integrationstestspezifikation | in [21] enthalten |
| [34] | HW/SW-Integrationstestspezifikation | in [21] enthalten |
| [39] | SW-Architektur- und Entwurfsverifikationsbericht | Siehe PQSP |
| PQSP | SW-Komponentenentwurfsphase | |
| [40] | SW-Komponentenentwurfsspezifikation | *Für BI(SIL0) optional, nicht erstellt* |
| [41] | SW-Komponententestspezifikation | *Für BI(SIL0) optional, nicht erstellt* |
| [49] | SW-Komponentenentw.verifikationsb. | *Für BI(SIL0) optional, nicht erstellt* |
| [5] | SW-Implementierungsphase | |
| [50] | SW-Quellcode und unterstützende Dokumentation | n.a. |
| [6] | SW-Komponententestphase |  |
| [60] | SW-Komponententestbericht | *Für BI(SIL0) optional, nicht erstellt* |
| [69] | SW-Quellcodeverifikationsbericht | Siehe PQSP (optional) |
| [7] | SW-Integrationsphase |  |
| [70] | SW-Integrationstestbericht | in [80] enthalten |
| [71] | HW/SW-Integrationstestbericht | in [80] enthalten |
| [79] | SW-Integrationsverifikationsbericht | *Für BI(SIL0) optional, nicht erstellt* |
| [8] | **SW-Validierungsphase** | |
| [80] | Gesamtsoftwaretestbericht | Siehe PQSP |
| [88] | Werkzeug-Validierungsbericht | *Für BI(SIL0) optional, nicht erstellt* |
| [89] | SW-Validierungsbericht | Siehe PQSP |
| [9] | SW-Bereitstellungs- und Wartungsphase | |
| [90] | Software-Änderungs- und Wartungsaufzeichnungen | SW-Wartungsprozess <A> redmine oder <B>Jira bzw. Bestandteil der SW Release Note |
| [91] | Software-Freigabemitteilung  (SW-Release Note) | Siehe PQSP |
| [92] | SW-Bereitstellungsplan | in [10] enthalten |
| [93] | SW-Bereitstellungshandbuch | in [91] enthalten |
| [94] | SW-Bereitstellungsaufzeichnungen | *Für BI(SIL0) optional, nicht erstellt* |
| [95] | SW-Wartungsplan | in [10] enthalten |
| [96] | SW-Änderungsaufzeichnungen | Siehe PQSP |
| [97] | SW-Wartungsaufzeichnungen | Siehe PQSP |
| [99] | SW-Bereitstellung & Wartungsverifizierungsbericht | *Für BI(SIL0) optional, nicht erstellt* |

Tabelle 7: Projektbezogene Dokumente-Cross-Referenz zur Norm

# SW-Konfigurationsmanagementplan

## Konfigurationskontrolle

Die Konfigurationskontrolle hat die Aufgabe, freigegebene Stände von Dokumenten oder von Software festzuhalten und jeweils die Änderungen zum vorherigen Stand aufzuzeigen.

Alle Stände der freigegebenen Software werden im Nexus (repo.stadlerrail.com) verwaltet.

Die Dokumente werden auf dem Netzlaufwerk verwaltet und können zusätzlich über ein projektspezifisches GitLab Repository versioniert werden..

## Dokumentationsmanagement

### Identifikation/Versionierung von Dokumenten

Dokumente welche im Laufe des Software-Zyklus zu erstellen sind, werden im Freigabestatus versioniert. Hierzu werden die internen Firmen-Vorgaben gemäß Prozessbeschreibung SW-Entwicklung [1] berücksichtigt.

Die Dokumenten-Stammnummer für die FIS-SW-Dokumentation lautet:

**Projektübergreifende Dokumente:**

STMA4000{lang}{nnnn}  
 {lang} = Sprache: D: Deutsch, E: Englisch  
 {nnnn} = 4-stellige Nummer

Der QSP wird unter dem folgenden Verzeichnis abgelegt:

**STMA4000D0001\_\_SW-QSP\_\_A.docx** [F:\Documentation\STAMA\Documents\4000\_FIS](file:///F:\Documentation\STAMA\Documents\4000_FIS)

**Projektspezifische Dokumente:**

STMA{jahr}{{lang}{nnnn}  
 {jahr} = Jahr, vierstellig  
 {lang} = Sprache: D: Deutsch, E: Englisch  
 {nnnn} = 4-stellige Nummer

Projekte können ihre Dokumente in entsprechenden Nummernbereichen reservieren.

Der PQSP wird unter dem folgenden Verzeichnis abgelegt:

f:\Orders\STAMA\{Kunde}\{Projekt}\Center FIS\300\_System\310\_SW-Planung\ **STMA{jjjj}D{nnnn}\_{revision}\_\_PQSP\_FIS\_{Kunde}\_{Projekt}.docx**  
  
z.B.   
f:\Orders\STAMA\STAP\FLIRT - DB PFZ\Center FIS\   
 300\_System\310\_SW-Planung\  
 STMA2025D0041\_B\_\_PQSP\_FIS\_{Kunde}\_{Projekt}.docx

|  |  |
| --- | --- |
|  | **Abgeschlossene Projekte**  Ist ein Projekt abgeschlossen wird es in den Ordner \_Abgeschlossene Projekte innerhalb des Kundenordners verschoben. |

### Pflege der Softwaredokumentation

Die Dokumentenprüfung erfolgt nach dem 4-Augen-Prinzip und wird in der Versionsverwaltung GitLab [www.gitlab.com/stadlerrail/stama/] verwaltet.

### Archivierung der Softwaredokumentation

Die Ablage und Archivierung der Softwaredokumentation erfolgt mittels Versionsmanagement und wird im PQSP genauer spezifiziert.

### Ordnerstruktur

Die Ablage von Dokumenten, welche im Softwareentwicklungsprozess erstellt werden (Dateien und Quellcode), erfolgt gemäß folgender Ablage.

| **Nr** | **Name** | **Verantw.** | **Beschreibung** |
| --- | --- | --- | --- |
| 0 | 00\_Organisation | TPM |  |
| 01 | 01\_System\_Anforderungen | RQM |  |
| 02 | 02\_Softwareplanung | AL |  |
| 03 | 03\_Softwarearchitektur | DES |  |
| 04 | 04\_Gestamtestspezifikation | TST |  |
| 05 | 05\_Software-Quellcode | ENT |  |
| 06 | 06\_Softwarefreigabe | QA |  |
| 07 | 07\_Release-Notes | TPM |  |

Tabelle 8: Ordnerstruktur der Ablage

## SW-Konfigurationsmanagement

### Gesamtsoftware-Versionierung und Änderungsklassen

Ein Softwarestand wird mit einer 3-Stelligen Nummer punktgetrennt festgehalten. Anhand der Nummerierung kann die Änderung bzw. Kompatibilität zwischen verschiedenen Softwareständen abgeleitet werden. Dies wird mit der Definition von Änderungsklassen festgehalten. Mit jeder neuen Software wird gemäß folgendem Schema die Version hochgezählt:

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Übersicht Aufbau der Softwareversion** | | | | |
| Version | Softwarestand | Neuer  Meilenstein | Neue  Funktionen | Fehlerbeseitigung |
| 1.0.0 | Freigegeben | Ja | Ja | Ja |
| 1.1.0 | Freigegeben | Nein | Ja | Ja |
| 1.1.1 | Freigegeben | Nein | Nein | Ja |
| **SW-Änderungsklasse** | | **A** | **B** | **C** |

Tabelle 9: Übersicht SW-Versionierung

Die Einstufung der Änderungsklasse definiert im Prozess maßgebend den Testumfang.

|  |  |
| --- | --- |
| **Einstufung SW-Änderungsklasse** | |
| **Klasse** | **Beschreibung** |
| **A** | wenn die neue Software für den nächsten Meilenstein generiert wurde  **Wird getestet durch QA-Team.** |
| **B** | wenn sich die neue Software zur vorhergehenden Software  nicht rückwirkungsfrei auf andere Funktionen oder Module verhält |
| **C** | wenn sich die neue Software zur vorhergehenden Software  rückwirkungsfrei auf andere Funktionen oder Module verhält |

Tabelle 10: Übersicht SW-Änderungsklassen

Der Prozess wird im PQSP spezifiziert

1. Laborrelease: **0.1.0**
2. offizielle Version: **1.0.0 (IBS-Release)**
3. offizielle Version: **2.0.0 (Typtest)**
4. offizielle Version**: 3.0.0 (Fahrzeugzulassung / Fahrgastbetrieb)**

|  |  |
| --- | --- |
|  | **Zähler**  Die Fahrzeugversion hat keinen Konfigurationszähler. |

### Versionierung der Microservices und der Schnittstellen (Entwicklungsversionen)

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Übersicht Aufbau der Softwareversion** | | | | |
| Version | Softwarestand | Neue  Schnittstelle | Neue  Funktionen | Fehlerbeseitigung |
| 1.0.0 | Freigegeben | Ja | Ja | Ja |
| 1.1.0 | Freigegeben | Nein | Ja | Ja |
| 1.1.1 | Freigegeben | Nein | Nein | Ja |
| **SW-Änderungsklasse** | | **A** | **B** | **C** |

Tabelle 11: Übersicht MS-SW-Versionierung

Änderungsklassen:

Die Einstufung der Änderungsklasse gibt im Änderungsprozess maßgebend den Testumfang vor.

|  |  |
| --- | --- |
| **Einstufung SW-Änderungsklasse** | |
| **Einstufung** | **Beschreibung** |
| **A** | wenn die neue Software zur vorhergehenden Software   * nicht schnittstellenkompatibel ist * sich nicht rückwirkungsfrei auf andere Funktionen oder Module verhält |
| **B** | wenn die neue Software zur vorhergehenden Software   * schnittstellenkompatibel ist * sich nicht rückwirkungsfrei auf andere Funktionen oder Module verhält |
| **C** | wenn die neue Software zur vorhergehenden Software   * schnittstellenkompatibel ist * sich rückwirkungsfrei auf andere Funktionen oder Module verhält |

Tabelle 12: Übersicht SW-Änderungsklassen

|  |  |
| --- | --- |
|  | **Zähler und Semantic Versioning**  Semantic Versioning wird von der genutzten Konfiguration in Debian-Paketen nicht unterstützt. Der im Debian Konfiguration genutzte Zähler sollte für Releases immer auf 1 stehen. |

* Die erste Labor Version **0.1.0-1**
* Die erste freigegebene Software erhält den Stand **1.0.0-1 (IBS-Release)**
* Die zweite freigegebene Software mit neuen Funktionen erhält den Stand **1.1.0-1**
* Die dritte freigegebene Software mit Bugfix erhält den Stand **1.1.1-1**
* Die vierte freigegebene Software mit inkompatibler Schnittstelle erhält den Stand **2.0.0-1 (Typtest)**
* **Version 3.0.0-1 (Fahrzeugzulassung**/**Fahrgastbetrieb)**

### Archivierung der Software

Die Archivierung der Software wird im GitLab durchgeführt. Dazu wird für jedes Release ein „Tag“ angelegt. Der „Tag“ enthält im Namen die Versionsnummer des Releases des Softwarepaketes. Auch für die im GitLab verwalteten Testfälle (Python-Skripte) wird ein „Tag“ angelegt.

Die aus dem „Tag“ automatisch erstellte Software wird in Nexus abgelegt  
 https://repo.stadlerrail.com/#browse/browse:raw-stama:STAMA%2FFIS

# SW-Verifikationsplan

Beschrieben im separaten Dokument FIS SW-Verifikationsplan [PIS\_5]

# SW-Validierungsplan

## Validierungsprozess

Der Validierungsprozess umfasst zum einen die Ausführung von SW-Tests (Gesamtsoftwaretests = Validierungstests), sowie zum anderen die abschließende Validierung aller ausgeführten Lebenszyklus-Aktivitäten, dokumentiert im SW-Validierungsbericht.

Dem Validierer obliegt die Verantwortung, anhand der erhaltenen Validierungsergebnisse zu entscheiden, ob die Aktivitäten in den einzelnen Lebenszyklusphasen hinreichend sind, oder ob Änderungs-/ Korrekturmaßnahmen durchgeführt werden müssen.

|  |  |
| --- | --- |
|  | **Verantwortung und Entscheidungsprozess**  Das Ergebnis der Validierung wird der AL und dem TPL vorgelegt.  Die Entscheidungsverantwortung bleibt bei diesen Personen. |

Details sind in den beiden folgenden Unterkapiteln beschrieben.

## Test der Gesamtsoftware

Die SW-Validierung testet die SW-Anforderungen in einer reproduzierbaren Laborumgebung anhand definierter Tests. Die erhaltenen Ergebnisse werden in einem Testbericht dokumentiert. Die Ausführung des Systemtests kann manuell oder automatisiert erfolgen. Die Berichte werden in [J/X] gesammelt und als Gesamttestbericht veröffentlicht.

Der Test der Gesamtsoftware wird auf Basis der SW-Anforderungen ausgeführt.   
Dabei erfolgt die Prüfung hauptsächlich über die durch die Nutzer erreichbaren Schnittstellen. Wenn zwingend notwendig werden Systemfunktionen durch entsprechende funktionsgenaue Softwaresimulationen ersetzt (z.B. Positionsinformation des GPS oder Odometer).

Der Testentwurf erfolgt dabei parallel zur Entwicklungsphase der Software, auch wenn dabei ggf. die Adaptierung der Tests mit den Entwurfsentscheidungen während der Entwicklung abgestimmt werden müssen (z.B. internes Schnittstellendesign).

Die Testfälle werden in der Metasprache Gherkin definiert und mittels Robot-Framework und Python Skripten implementiert.

Nach abgeschlossener Implementierung wird die neue Funktion gegen die implementierte Software getestet. Im ersten Lauf erfolgt die Erstprüfung gemeinsam zwischen Entwicklern und Qualitätssicherung. Nach erfolgreicher Erstprüfung wird der Test automatisch, spätestens für jedes SW-Release ausgeführt.

Neue Features oder bekannte Fehler werden bewertet ob für diese ein neuer Testfall benötigt wird.  
Das Verfahren entspricht sonst, dem oben für Anforderungen beschriebenen Vorgehen.

Für jeden Test können Markierungen (Tags) definiert werden, die für die Auswahl der automatischen Ausführung genutzt werden.

Für die Tests werden Prüfgestelle (Testracks) genutzt, die jede Hardwarekomponente auf der die Software ausgeführt wird mindestens einmal enthalten. Zentrales System der Prüfgestelle ist ein PC, der mit der Softwareversionsverwaltung GitLab verbunden ist.

|  |  |
| --- | --- |
|  | **Sonderfälle und Entscheidungskriterien**  Aufgrund von Tests kann es notwendig sein:   1. Bestimmte identische Systeme mehrfach in den Prüfstand zu verbauen 2. Mehrere Systeme gleicher Funktion mit unterschiedlichem Hardware-Stand zu verbauen. 3. Prüfstände modular und mobil zu gestalten, damit sie im Anwendungsfall gekoppelt oder auch auf dem Fahrzeug selbst genutzt werden können. |

|  |  |
| --- | --- |
|  | **Automatisierung von Hardware-Funktionen**  Für automatisierte Systemtests kann es sinnvoll sein, weitere Elektronik im Prüfgestell einzubringen. Die Lautsprecherausgänge der ACU und der ADU sind z.B. über Soundkarten an einen Testrechner angeschlossen. Die digitalen Eingänge und Ausgänge der ADU und alle Stromversorgungen der Geräte sind über einen Modbus-Koppler auch an den Testrechner angeschlossen. Die Testautomatisierung kann somit alle wichtigen Schnittstellen der Geräte abtesten. |

Die Testautomatisierung ist im GitLab unter einem zentralen Projekt abgelegt. Bei jeder Änderung im Git-Master der FIS-Software wird der Test automatisiert über GitLab auf beiden FIS-Testtürmen durchgeführt. Das Testergebnis kann ebenfalls in GitLab abgerufen werden.

Bei Erzeugung einer Release- oder Testversion, in Form eines Git-Tags im zentralen Projektrepository, wird ein Testdurchlauf für dieses spezifische Release gestartet. Aus den Ergebnissen wird dann der Testreport erzeugt und im Dokumentenverzeichnis des Projektes abgelegt.

Durch die Automatisierung kann eine spezielle Auswahl entfallen und die Prüfung als Gesamtsoftwaretest für jedes SW-Release ausgeführt werden.

## SW-Validierungsbericht

Der SW-Validierungsbericht wird erstellt und beinhaltet die Zusammenfassung aller Verifikations- und Validierungsergebnisse, inkl. der Analyse und Bewertung erkannter Abweichungen.

Der Validierer muss im Bericht eine Aussage treffen, ob die validierte Software die gemäß der Sicherheits-Integritätsstufe gestellten Anforderungen erfüllt und für den Betrieb geeignet ist bzw. freigegeben werden kann (gemäß [N\_1], Kap. 6.3.4.16).

Der Validierer ist berechtigt, die Freigabe von Software zu verhindern, wenn die Validierungskriterien nicht erfüllt sind, oder gegebenenfalls zusätzliche Überprüfungen, Analysen und Tests anzufordern.

# SW-Bereitstellungsplan

Die Bereitstellung der Software wird durch eine eindeutige SW-Release-Version gekennzeichnet.

Sie erfolgt unter Verantwortung des PM, nach erfolgreichem Abschluss der SW-Validierung.  
Die SW-Validierung wird im SW-Validierungsbericht dokumentiert.

Der Bereitstellungsprozess umfasst:

* Archivierung der freigegebenen SW unter Versionskontrolle (GitLab)
* Freigabe einer Release Note (in [N\_1] als Freigabemitteilung bezeichnet)
* Bereitstellung des Software-Paketes an den Kunden durch den PM  
  SW Release Note  
  SW-Artefakte  
  Sammlung der notwendigen zusätzlichen Dokumente

Die SW Release Note, ist das führende Dokument im Bereitstellungsprozess.  
Es muss auf alle relevanten Dokumente referenzieren.

Sie enthält die folgenden Informationen:

* Allgemeine Dokumenteninformationen
  + Dokumentennummer / Identifikation
  + Status
  + Revision
  + Datum
  + Autor
  + Prüfer / Freigabe
* Bezeichnung der SW-Version
  + Release-Version
  + Prüfsumme
* Allgemeine Hinweise zur SW-Installation
* Angaben zur Konfiguration
* Angaben zum jeweils aktuellen SW-Release:  
   (einschließlich Identifikation aller genannten Punkte)
  + Geänderte Anforderung zum vorherigen SW-Release
  + (noch) fehlende Anforderungen
  + Bekannte oder noch nicht behobene Fehler
* Liste relevanter Projektdokumente mit Revisionsangabe
* HW-Abhängigkeiten
* SW-Abhängigkeiten
* Kompatibilitätsangaben zu Vorgängerversionen
* Angaben zum aktuellen Release
  + zugehörige sonstige Versionen: Libraries, Firmware
  + Sonstige zur Release gehörigen SW/ FW-Angaben (falls vorhanden)
* Download-Verantwortung (wer und wie)

# SW-Wartungsplan

Software-Wartung bezeichnet hier die Anpassung der Software oder ihrer Komponenten.  
Die Wartung beginnt mit dem erfolgreichen Abschluss der Fahrzeugintegration (Typentest) beziehungsweise der Fahrzeugzulassung (Release 3.0.0).

|  |  |
| --- | --- |
|  | **Versionierung der Gesamtsoftware**  Die Releases 1.0.0 und 2.0.0 werden im Rahmen der Fahrzeugzulassung als Testreleases verwendet. 2.0.0 ist im Prinzip der Releasekandidat für 3.0.0. |

Die Ziele der SW-Wartung sind die Erfüllung von Kundenwünschen hinsichtlich funktionaler Anpassungen/ Erweiterungen, die Behebung von Fehlern, die Verbesserung der Leistung oder anderer Softwareeigenschaften oder die Anpassung an eine veränderte Umgebung.

Die SW-Wartung umfasst sowohl organisatorische als auch technische Prozesse.

Je nach Änderungsumfang werden die Verifikationsaktivitäten wiederholt, in der Regel als Delta-Verifikation.

## Beteiligte Rollen

* TPM (Technischer Projektmanager):  
  Koordiniert die Kundenkommunikation, die Release-Planung und die Lieferung
* CPM (Kommerzieller Projektmanager):  
  Schreibt die Angebote für den Kunden und überwacht das Budget
* DES (Software-Designer / Architekt):  
  Bewertet die technische Machbarkeit und definiert die Softwarearchitektur
* PO (Produktverantwortlicher):  
  Verwaltet Anforderungen, Priorisierung des Backlogs und Implementierungsplanung
* IMP (Software-Entwickler):  
  Implementiert Änderungen, Fehlerbehebungen und neue Funktionen
* QAE (Qualitätssicherungsingenieur):  
  Definiert und führt Testfälle aus, stellt die Softwarequalität sicher
* VER (Softwareprüfer / Tester):  
  Führt Testfälle aus, führt Regressions- und Funktionstests durch
* VAL (Validator):  
  Überprüft Testberichte, Dokumentation und Release-Konformität
* TW (Technischer Redakteur):  
   Aktualisiert Softwaredokumentation, Handbücher

## Phase 1: Fahrgastbetrieb (3.0.0)

**Generische Release-Liefergegenstände (Deliverables):**

1. Softwarecontainer / Paket + Konfigurationen
2. Software-Release-Notes
3. Software-Wartungsbericht (Auswirkungsanalyse)
4. Software-Anforderungsspezifikation
5. Software-Test-Spezifikation
6. Software-Testbericht
7. Softwarehandbücher (Zugführer, Personal, Werkstatt, Backoffice)

## Phase 2: Vorfallmanagement

**Trigger:** Neue Vorfallberichte   
 (Dokumentiert hauptsächlich von TPM in Jira)

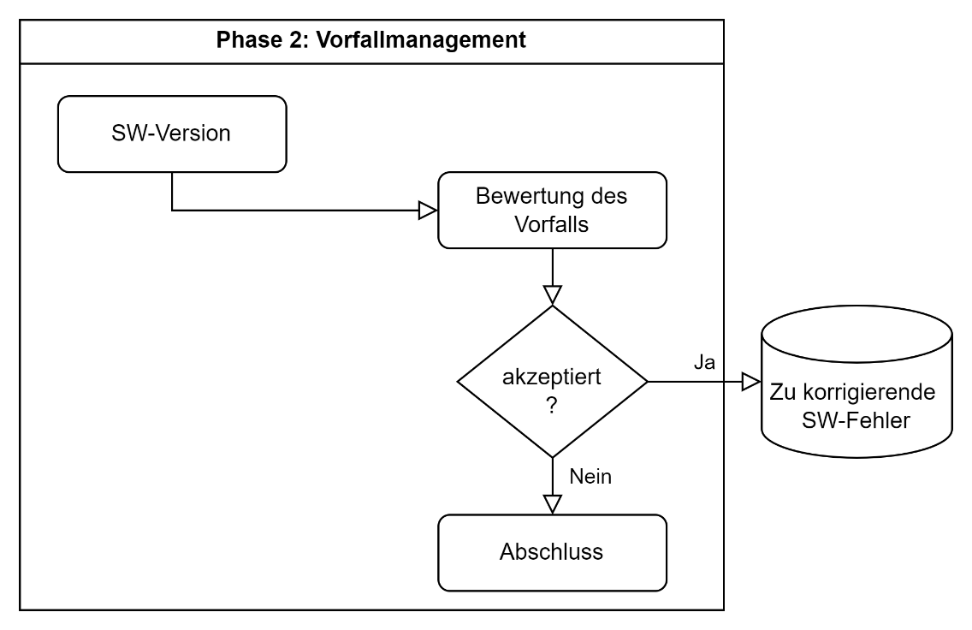


Abbildung 4 – Phase 2: Vorfallmanagement

**Eingang:**

* Kundenberichte: Vorfälle, die von Kunden über TPM gemeldet werden
* Interne Berichte: Vorfälle, die von QAE, IMP, DES - hauptsächlich über Hazard-Lock - identifiziert werden

**Aktivitäten: Problemanalyse**

* QAE: Bewertet und kategorisiert gemeldete Probleme
* PO: Bewertet betroffene Anforderungen
* TPM/ QAE/ PO: Gemeinsame Entscheidung, den Vorfall anzunehmen oder abzulehnen
* Bei Annahme:
  + QAE: Erstellt Testfall zur Validierung
  + QAE: Überprüft die Anwendbarkeit auf andere Releases (Hazard-Lock)
  + DES: Bewertet die Machbarkeit einer Lösung
  + IMP: Bewertet den erforderlichen Implementierungsaufwand

**Ergebnis:**

* Aktualisierte Liste bekannter Probleme
* Aktualisierter Entwurf der Release-Note (im Abschnitt zu bekannten Problemen)
* Neue Testfälle zur Validierung des Vorfalls

## Phase 3: Analyse von Änderungsanfragen

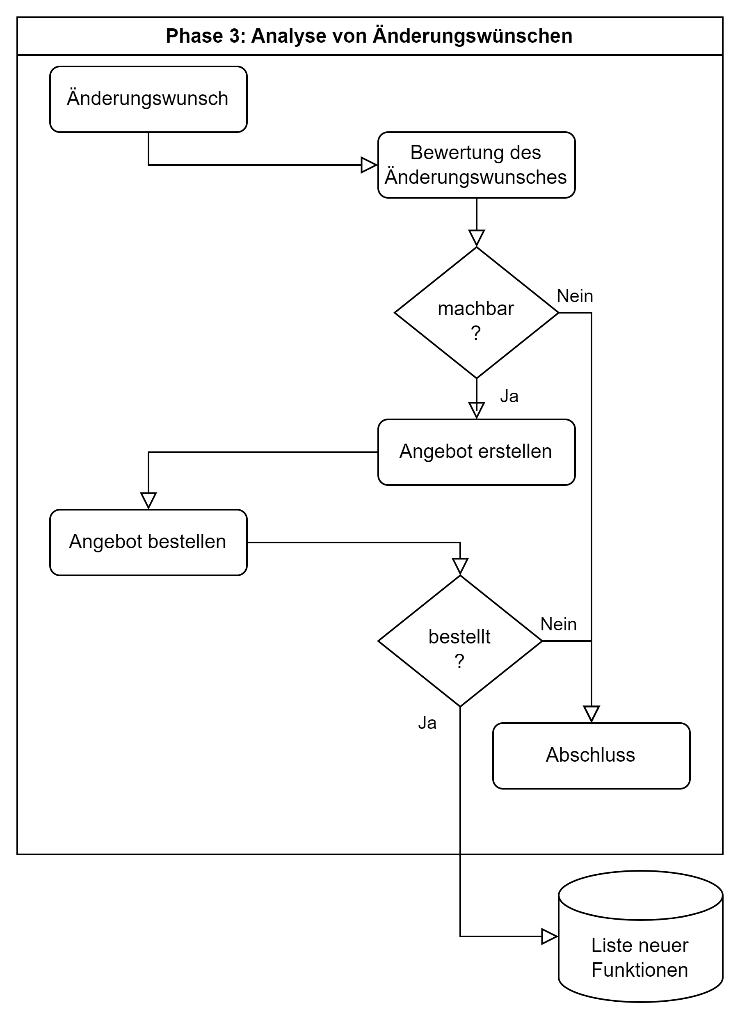
**Trigger:** Kundenanfragen nach neuen Funktionen oder Softwaremodifikationen   
 (dokumentiert von TPM in Jira)   
**Eingang:**

Abbildung 5 – Phase 3: Analyse von Änderungswünschen

* TPM: Sammelt und dokumentiert die Änderungswünsche des Kunden
* DES: Bewertet die Machbarkeit der Implementierung
* PO/ DES: Bewertet den erforderlichen Implementierungsaufwand

**Aktivitäten:**

* TPM/ DES/ PO: Entscheidung über die Machbarkeit
* DES/ PO/ QAE: Aufwandsabschätzung
* TPM: Diskutiert die Kostenfreigabe mit dem Kunden und definiert die vorläufige Release-Planung
* CPM: Gibt eine offizielle Bestellung für die Änderung auf

**Ergebnis:**

* Liste der angeforderten und bewerteten Änderungen
* Änderungsangebot von TPM und CPM an den Kunden

## Phase 4: Planung & Vorbereitung

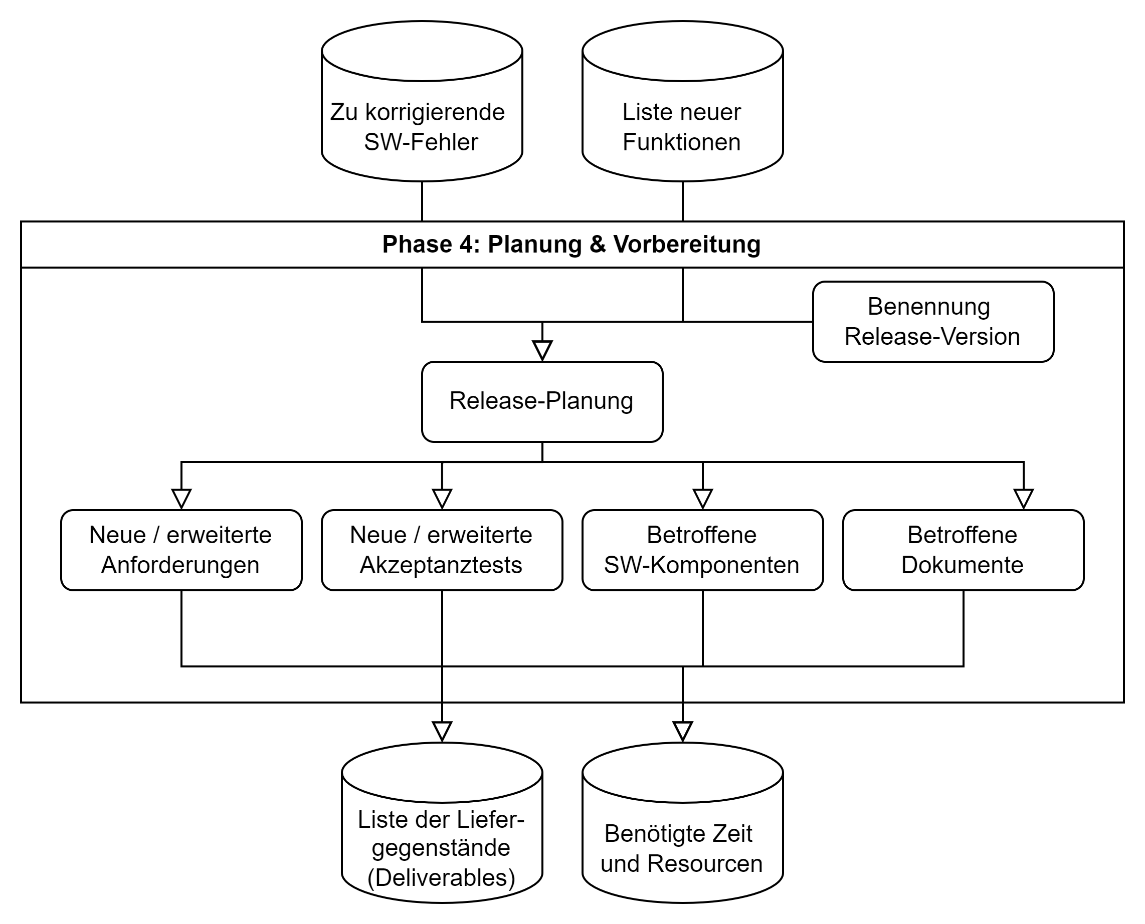
**Trigger:** Genehmigte Fehlerbehebungen und bestellte Änderungsanfragen   
 (CPM/TPM in Jira)

Abbildung 6 – Phase 4 Planung & Vorbereitung

**Eingang:**

* Ergebnisse aus Phase 2
* Ergebnisse aus Phase 3

**Aktivitäten:**

* Definition des Release-Typs (Fehlerbehebung, Funktionsupgrade, Hauptversion)
* TPM/ PO/ QAE/ DES: Release-Planung
* Auswirkungsanalyse: Bewertung der erforderlichen V-Modell-Schritte
  + DES: Welche Anforderungen und Architekturbeschreibungen sind betroffen
  + QAE: Welche Akzeptanztests sind betroffenen
  + TW: Welche Dokumente sind betroffen
  + IMP: Welche SW-Komponenten sind betroffen

**Ergebnis:**

* Detaillierte Liste der Liefergegenstände, die sich für das Release ändern
* Aktivitäten und Aufwände, die für das Release notwendig sind.

## Phase 5: Implementierung & Validierung

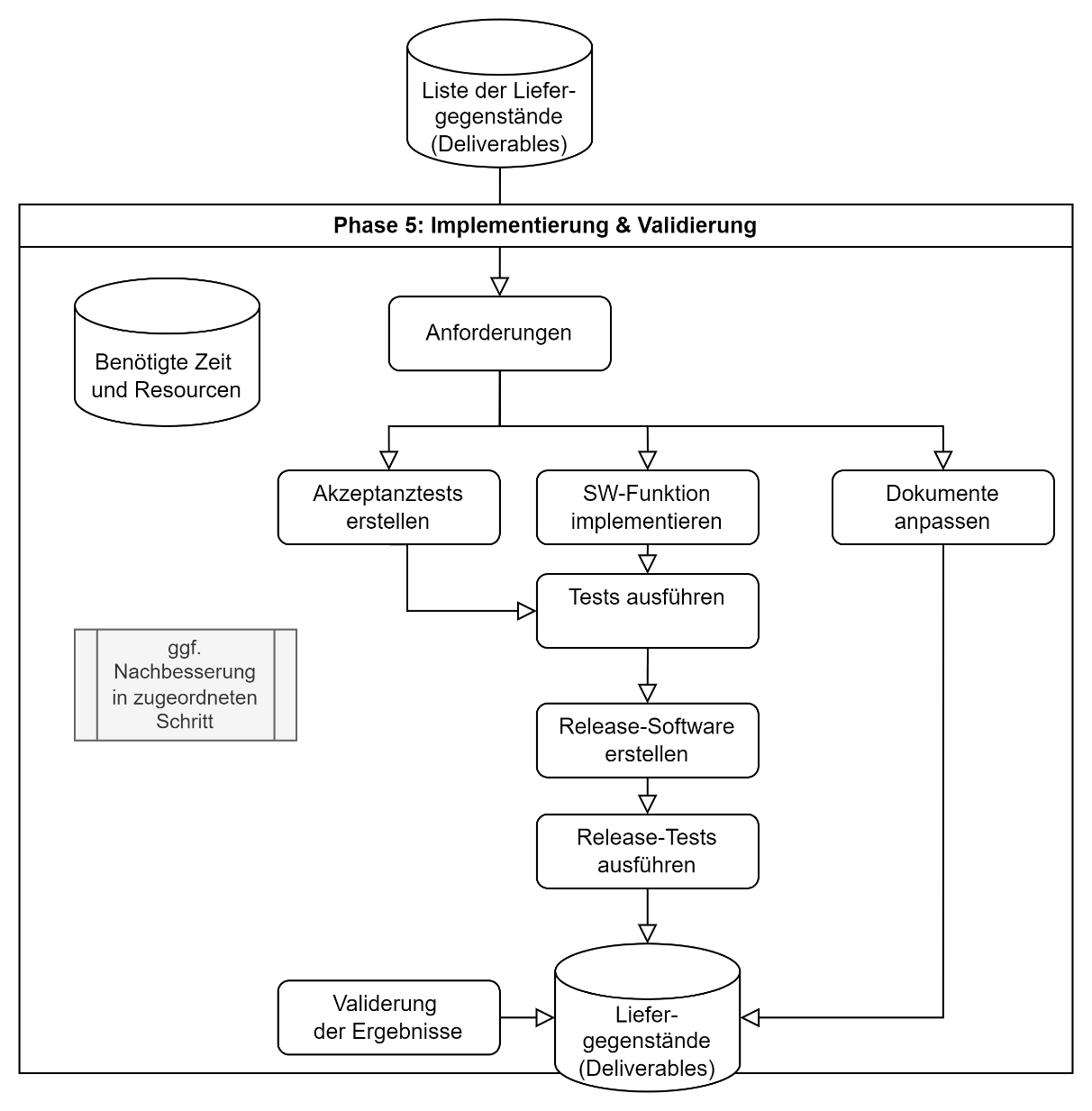
**Trigger:** Freigabe der Planung und des Budgets

Abbildung 7 – Phase 5: Implementierung & Validierung

**Eingang:**

* Ergebnisse aus Phase 4

**Aktivitäten: Implementierung von Funktionen und Tests**

* QAE (VER): Erweitert / aktualisiert Akzeptanztests
* IMP: Implementiert die SW-Änderungen
* QAE (VER): Führt automatisierte und manuelle Tests am Release-Kandidaten durch
* PO: Generiert Release-Kandidat
* QAE (VER): Führt vollständige Testsuite einschließlich Regressionstests durch
* PO/QAE(VAL): Validierung / Überprüfung der Dokumente und Testberichte

**Nachbesserung**

* IMP/ QAE: Anforderung unvollständig
  + Entwurfsdetaillierung durch DES (Software-Designer)
* DES/ IMP: Implementierung nicht machbar
  + Erneute Analyse der Änderungsanfrage (TPM)
* QAE: Implementierung unvollständig
  + Softwarekorrektur durch IMP (Software-Entwickler)
* TW/ VAL: Dokumentation unvollständig
  + Dokumentenkorrektur durch TW (Technischer Redakteur)
* TPM/ VAL: Tests unvollständig
  + Testkorrektur durch VER (Softwareprüfer / Tester)

**Ergebnis:**

* Getestete und überprüfte Liefergegenstände

## Phase 6: Software-Veröffentlichung & Übergabe

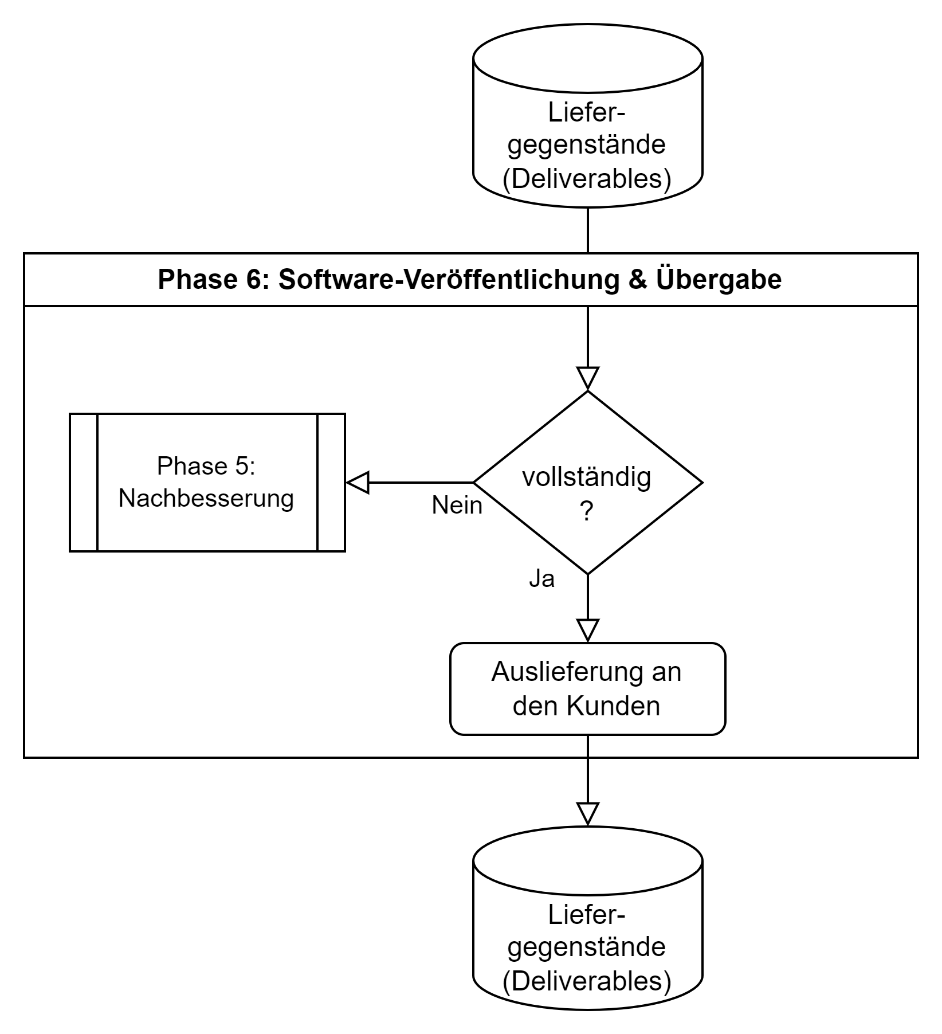
**Trigger:** Erfolgreiche Validierung und Verifizierung

Abbildung 8 – Phase 6: SW-Veröffentlichung & Übergabe

**Eingang:**

* Ergebnisse aus Phase 5

**Aktivitäten: Release**

* TPM: Überprüft die Ergebnisse und Liefergegenstände aus Phase 5
* TPM: Liefert die Software und Dokumentation an den Kunden

**Ergebnis: Release-Paket**

* Definierte und aktualisierte Liefergegenstände
* unveränderte Liefergegenstände aus früheren Releases

# Richtlinien, Prozeduren und Werkzeuge

## Richtlinien und Prozeduren

Während des Entwicklungsprojekts wird nach den in der folgenden Tabelle genannten Richtlinien und Prozeduren, zuzüglich der jeweils verwendeten Werkzeuge und Methoden gearbeitet.

|  |  |  |  |
| --- | --- | --- | --- |
| **Ref.** | **Phase** | **Prozeduren und Richtlinien** | **Werkzeuge und Methoden** |
| [1] | SW-Planung | Gemäß diesem Dokument und den hieraus referenzierten weiteren Planungsdokumenten | MS Office |
| [2] | SW-Anforderungen | Gemäß diesem Dokument | MS Office  <A> Redmine Ticketsystem <B>Jira Ticketsystem GitLab |
| [3] | SW-Architektur- und  Entwurf |
| [4] | SW-Komponentenentwurf |
| [5] | SW-Implementierung | Gemäß Programmierrichtlinien | GitLab |
| [6] | SW-Komponententest | Gemäß SW-Verifikationsplan | entfällt |
| [7.1] | SW-Integration | Gemäß diesem Dokument und den zu erstellendenden Integrationstestspezifikationen | Siehe unten [8] |
| [7.2] | HW/SW-Integration | Siehe unten [8] |
| [8] | SW-Validierung | Gemäß SW-Validierungsplan | FIS-Testautomatisierungsumgebung |
| [9] | SW-Bereitstellung und Wartung | Gemäß diesem Dokument (Kap. SW-Bereitstellungsplan und Kap. SW-Wartungsplan) | GitLab (Versionsmanagement)  <A> Redmine Ticketsystem <B>Jira Ticketsystem,  Nexus |

Tabelle 13: Lebenszyklusphasen, Prozeduren und Richtlinien

## Werkzeuge (Tools)

Die Werkzeuge werden vom SW-Projektmanager bereitgestellt.

Alle verwendeten Werkzeuge sind unter Stadler-Versionskontrolle gehalten, zum einen zentral von der Stadler-IT verwaltete Werkzeuge (z.B. MS Office), zum anderen projekt-spezifische Werkzeuge, bereitgestellt in der Projektablage.

| **Tool** | **Purpose** | **Klasse** | **Eigentümer** | **Version** |
| --- | --- | --- | --- | --- |
| MS Office | Dokumentenerstellung | T1 | Microsoft | Verwaltung durch Stadler IT |
| WebStorm | Frontend Entwicklung | T1 | intelliJ | Lizensiert  durch STAMA |
| Visual Studio | Backend Entwicklung  und Testdesign | T3 | Microsoft | Open Source, verwaltet  durch STAMA |
| C++ und Cmake. Clang | System Programmiersprache und Meta-Build-Generator und Compiler für C++ | T3 | Diverse Projekte | Open Source, verwaltet  durch STAMA |
| Rust Compiler und Werkzeuge | System Programmiersprache mit Compiler und anderen Werkzeugen | T3 | Open Source | Open Source, verwaltet  durch STAMA |
| Docker und Docker Compose | Virtualisierung von Systemhardware, hier genutzt für Testsystem und Applikation. | T2 | Docker.io | Open Source, verwaltet  durch STAMA |
| AsyncAPI | Dokumentationsstandard für Schnittstellen | T1 | Open Source | Open Source, verwaltet  durch STAMA |
| Quicksilver | Generator für AsyncAPI Dokumente zu Validierungsschemata. | T2 | Open Source | Open Source, verwaltet  durch STAMA |
| Redmine [2a] | Ticket-System: <https://stap-software-redmine.stadlerrail.com/projects> | T1 | Redmine.org | Verwaltung durch Stadler IT |
| Jira [2b] XRay Testing | Ticket-System: <https://stadlerrailag.atlassian.net> | T1 | Atlassian | Verwaltung durch Stadler IT (Cloud Anwendung) |
| Confluence | Dokumentationsplattform (Protokolle, Wiki, Planung) | T1 |  |  |
| GitLab | Versionsmanagement, Konfigurationsmanagement | T1 | gitlab.com | Verwaltung durch gitlab.com (Cloud Anwendung) |
| Python und FisTS | Ausführung von Komponententest und Systemtests | T1 | Python.org  FisTs | 3.7.3 Veraltung durch STAMA 2025.1.0 entwickelt durch STAMA |
| GitLab Nexus | Sicherung und Verteilung von Releases | T1 |  |  |
| XRay Generator | Dokumentationswerkzeug für technische Dokumentation | T1 | User Lizenzen | Verwaltung über Jira und Stadler IT |
| Asciidoctor Asciidoctor-pdf | Dokumentationswerkzeug für technische Dokumentation | T1 | Open Source | Verwaltung durch STAMA |

Tabelle 14: Werkzeuge (Tools)

# Anforderungsverfolgung

Anforderungsverfolgung findet von der System-Anforderung zur Software-Anforderung statt. Es wird ebenfalls eine Verknüpfung zu den Testfällen hergestellt.  
Optional werden Verknüpfungen in die Architekturbeschreibung und die Komponentenbeschreibungen hergestellt. Diese sind aber primär für die interne Referenz.  
Die Verknüpfung von System-ASP zu SW-ASP ist in der Regel 1:n, also genau eine Quelle. Für SW-Tests können n:m Verknüpfungen eingesetzt werden, wobei auch hier eine 1:m Beziehung angestrebt werden sollte.

Im Ticketstatus kann auf einfache Weise geprüft werden:

* Welche SASP-Tickets noch nicht auf SW-Anforderungstickets verlinkt sind
* Welche SASP-Tickets noch nicht abgeschlossen (vollständig umgesetzt) sind
* Welche SW-Anforderungstickets noch nicht abgeschlossen sind

In jedem SWASP-Ticket ist das Feld *Status* vorhanden und wird wie folgt verwendet:

|  |  |
| --- | --- |
| **Status** | **Bedeutung** |
| Neu oder  Klärung | Diese Anforderung ist noch nicht vollständig spezifiziert. Ist die Anforderung klar spezifiziert und geprüft, wird das Ticket auf „Implementierung“ gesetzt. |
| Implementierung | Die Anforderung ist bereit zur Implementierung. Der Testfall kann entworfen werden, wenn die Anforderung in den Status Implementierung gesetzt wird.   Ausgeführt wird der Test am Ende der Implementierung. Schlägt der Test fehlt, wird das Ticket vom Entwickler nicht auf den nächsten Status gesetzt. Zum Abschluss einer Implementierung wird auch ein manuelles Code-Review von einer anderen Person als dem Entwickler durchgeführt (Merge-Requests). |
| Abgeschlossen | Der Test für diese Anforderung wurde implementiert, ein Code-Review wurde durchgeführt und der Testfall ist fehlerfrei abgeschlossen. |

Tabelle 15: Redmine-Ticketstatus

# Spezifikationen für SW externer Lieferanten

## Firmware

Auf den FIS-Hardwarekomponenten sind COTS-Softwarekomponenten vorhanden und wird im PQSP beschrieben.

| **Tool** | **Purpose** | **Klasse** | **Eigentümer** | **Version** |
| --- | --- | --- | --- | --- |
| **BIOS** | Firmware | COTS | Sintrones | Siehe PQSP |
| **Debian Linux** | Betriebssystem | COTS | Open Source | >= Debian 12 (bookworm) |
| **Mosquitto** | MQTT Broker | COTS | Open Source | 1.6-open-ssl |
| **Open SSL** | Security | COTS | Open Source | >= 3.0 |
| **Audio TTS** | Funktion „Text To Speech“ | COTS | SemVox | >= 2.3.0 |
| **Webbrowser** | Anzeigen des MMI und der Fahrgastanzeiger, sowie der Diagnoseschnittstelle. | COTS | Open Source / Google | latest |
| **Docker** | Hauptsächlich als HW Virtualisierung, teilweise aber als virtuales Betriebssystem (z.B. Metro JJK) | COTS | Docker Inc. | >= 28.0.0 |

Tabelle 16: COTS-Softwarekomponenten

Die Softwarequalitätssicherung dieser Softwarekomponenten obliegt den jeweiligen Lieferanten. Die Auswahl der Versionen der COTS-SW-Komponenten sind in der SW-Architektur- und Entwurfsspezifikation definiert.

## HW-Komponenten von externen Lieferanten

SINTRONES vertreten durch:

SPHINX Computer Vertriebs-GmbH  
Heinrich-Lanz-Straße 14  
69514 Laudenbach  
Deutschland  
  
Tel: +49 (0) 6201 75437  
Fax: +49 (0) 6201 74246  
Web: [www.sphinxcomputer.de](https://www.sphinxcomputer.de/)

# Umsetzung des SW-Qualitätssicherungsplans

Der Leiter der Abteilung SW-Engineering und der SW-Engineering Teamleiter überprüfen die Umsetzung der Entwicklungsvorgaben der vorangegangenen Kapitel.

# Freigabe der Entwicklungsergebnisse

Die Entwicklungsergebnisse für das Produkt FIS und seine Erweiterungskomponenten (Reservierungssystem, Fahrgastzählung und Fahrerassistenz) werden vom technischen Projektmanager (TPM) freigegeben. Das Produkt FIS-Control-Center wird als externes Werkzeug gewertet und nicht auf dem Fahrzeug ausgeführt. Es benötigt damit keinen Nachweis nach [N\_1]

# Werkzeuge und Techniken

Die folgende Tabelle zeigt die gemäß zu erreichendem Sicherheitsintegritätsniveau **Basisintegrität** die im Projekt anzuwendenden Methoden und Maßnahmen. Die Erfüllungsprüfung findet im Rahmen der phasenspezifischen Verifikation statt. Die folgenden Informationen stellen eine Zusammenfassung der in [N\_1] Appendix A definierten Techniken dar. Für die Bewertung steht eine kommentierte Übersicht unter [PIS\_5] zur Verfügung.

| **Nr** | **Titel** | **Referenz** | **Bemerkung** |
| --- | --- | --- | --- |
| 1 | Vollständig definierte (externe) Schnittstellen | A.3/19 | Die Systemschnittstellen werden in der Architekturspezifikation gezeigt. Externe Schnittstellen werden als Async-API Dokumente mit Change-Log geführt. |
| 2 | Modularer Ansatz | A.4/4 | Die Funktionen werden nach VDV301 strukturiert und in Microservices implementiert, siehe Architekturspezifikation. |
| 3 | Komponenten | A.4/5 | Microservices |
| 4 | Vollständig definierte Schnittstellen | - A.20/4 | Async-API mit ausreichender Dokumentation |
| 5 | Software-Codierstandards | A.4/6 - A.12/1  - A.12/2  - A.12/8  - A.12/11 | Programmierrichtlinien [4002] |
| 6 | Programme, die sind analysierbare | A.4/7 | Siehe auch 12 |
| 7 | Streng typisierte Programmiersprache | A.4/8 | C++, Rust, TypeScript, Python (mit type-handles) |
| 8 | Funktionstests | A.5/8 | Unittests, Software Integrationstests |
| 9 | Schnittstellentests | A.5/10 | Zu den Systemen über die externen Schnittstellen |
| 10 | Integrationstests / Funktionstests | A.6/1  - A.14/5 | HW/ SW-Integrationstests  Fahrzeugsimulation |
| 11 | Gesamtsoftwaretests / Funktionstests | A.7/2 | Anforderungstests (Acceptance-Tests) |
| 12 | Statische Softwareanalyse | A.8/1 | TypeScript: ESLint  C++: clang-tidy Rust: clippy Python: Ruff |
| 12 | Walkthroughs / Entwurfsüberprüfungen | - A.19/6 | Merge-Requests mit Vier-Augen-Prinzip  (siehe auch Kapitel 10) |
| 13 | Übereinstimmung mit EN ISO 9001 | A.9/2 | Stadler Mannheim ist zertifiziert |
| 14 | Qualitätssicherungssystem der Firma | A.9/4 | Stadler Mannheim hat ein QM-System eingeführt |
| 15 | Software-Konfigurationsmanagement | A.9/5 | Siehe Kapitel 4 |
| 16 | Checklisten | A.9/6 | Verifikationschecklisten |
| 17 | Datenaufzeichnung und -analyse | A.10/2 | Siehe Kapitel 7 |

Tabelle 17: Werkzeuge und Techniken

# Change-Log (Änderungsverzeichnis)

|  |  |  |  |
| --- | --- | --- | --- |
| **Rev.** | **Änderungsgrund** | **Referenzen** | **Bemerkungen (Autor)** |
| Wann | Warum | Wo | Was (Wem) |
| A | Erstellt | Alle Kapitel | Neuer Prozess FLIRT\_BEMU\_PFZ mit Abwicklung/Entwicklung. |
| B  2025-08-20 | Überarbeitung | Alle Kapitel Klassifizierung Kapitel 1.1 Kapitel 3.1 | Internes Review Intern statt Kommerziell Abschnitt zu FIS-SW entfernt Vollständige Anforderungsverfolgung |
|  |  |  |  |