7.1 Checkliste Anwendungsklasse 1

Dieser Abschnitt fasst alle Empfehlungen der Anwendungsklasse 1 als Checkliste zusammen.

Empfehlung	Bemerkungen	Umgesetzt?
Qualifizierung		
EQA.1: Der Software-Verantwortliche kennt die verschiedenen Anwendungsklassen und weiß, welche für seine Software anzustreben ist.	Software-Verantwortliche: • Marc Garbade • Falk Heinecke	MG // 10.04.18
EQA.2: Der Software-Verantwortliche weiß, wie er gezielt Unterstützung zu Beginn und im Verlauf der Entwicklung anfordern und sich mit anderen Kollegen zum Thema Software-Entwicklung austauschen kann.	Umgesetzt.	MG // 10.04.18
EQA.3: Die an der Entwicklung Beteiligten ermitteln den Qualifikationsbedarf in Bezug auf ihre Rolle und die angestrebte Anwendungsklasse. Sie kommunizieren diesen Bedarf an den Vorgesetzten.	Umgesetzt. Für dieses Projekt ist kein unverzichtbarer externer Bedarf vorhanden.	MG // 10.04.18
EQA.4: Den an der Entwicklung Beteiligten stehen die für ihre Aufgaben benötigten Werkzeuge zur Verfügung und sie sind geschult in deren Benutzung.	Umgesetzt.	MG // 10.04.18
Anforderungsmanagement		
EAM.1: Die Aufgabenstellung ist mit allen Beteiligten abgestimmt und dokumentiert. Sie beschreibt in knapper, verständlicher Form die Ziele, den Zweck der Software, die wesentlichen Anforderungen und die angestrebte Anwendungsklasse.	Umgesetzt. Alle erforderlichen Dokumente befinden sich im Repository und werden (falls erforderlich) zusätzlich auf eine globale Teamsite gespiegelt.	MG // 10.04.18
EAM.3: Die Randbedingungen sind erfasst.	Umgesetzt. Die Aufgaben der Software sind klar definiert und vollumfänglich umgesetzt.	MG // 10.04.18

Empfehlung	Bemerkungen	Umgesetzt?
Software-Architektur		
ESA.2: Wesentliche Architekturkonzepte und damit zusammenhängende Entscheidungen sind zumindest in knapper Form dokumentiert.	Umgesetzt.	MG // 10.04.18
Änderungsmanagement		
EÄM.2: Die wichtigsten Informationen, um zur Entwicklung beitragen zu können, sind an einer zentralen Stelle abgelegt.	Umgesetzt. Die Software ist komplett versioniert.	MG // 10.04.18
EÄM.5: Bekannte Fehler, wichtige ausstehende Aufgaben und Ideen sind zumindest stichpunktartig in einer Liste festgehalten und zentral abgelegt.	Umgesetzt. Jedes Modul verfügt über eine eigene Änderungshistorie.	MG // 10.04.18
EÄM.7: Ein Repository ist in einem Versionskontrollsystem eingerichtet. Das Repository ist angemessen strukturiert und enthält möglichst alle Artefakte, die zum Erstellen einer nutzbaren Version der Software und deren Test erforderlich sind.	Umgesetzt.	MG // 10.04.18
EÄM.8: Jede Änderung des Repository dient möglichst einem spezifischen Zweck, enthält eine verständliche Beschreibung und hinterlässt die Software möglichst in einem konsistenten, funktionierenden Zustand.	Umgesetzt. Software wird täglich für Builds von MCODAC und BoxBeam verwendet.	MG // 10.04.18
Design und Implementierung		
EDI.1: Es werden die üblichen Konstrukte und Lösungsansätze der gewählten Programmiersprache eingesetzt sowie ein Regelsatz hinsichtlich des Programmierstils konsequent angewendet. Der Regelsatz bezieht sich zumindest auf die Formatierung und Kommentierung.	Umgesetzt.	MG // 18.05.18

Empfehlung	Bemerkungen	Umgesetzt?
EDI.2: Die Software ist möglichst modular strukturiert. Die Module sind lose gekoppelt, d.h., ein einzelnes Modul hängt möglichst gering von anderen Modulen ab.	Umgesetzt.	MG // 10.04.18
EDI.9: Im Quelltext und in den Kommentaren sind möglichst wenig duplizierte Informationen enthalten. („Don`t repeat yourself.“)	Umgesetzt.	MG // 10.04.18
EDI.10: Es werden einfache, verständliche Lösungen bevorzugt eingesetzt. („Keep it simple and stupid.“).	Umgesetzt.	MG // 10.04.18
Software-Test		
EST.4: Die grundlegenden Funktionen und Eigenschaften der Software werden in einer möglichst betriebsnahen Umgebung getestet.	Umgesetzt, siehe EÄM.8.	MG // 10.04.18
EST.10: Das Repository enthält möglichst alle für den Test der Software erforderlichen Artefakte.	Umgesetzt, sofern möglich.	MG // 10.04.18
Release-Management		
ERM.1: Jedes Release besitzt eine eindeutige Release-Nummer. Anhand der Release-Nummer lässt sich der zugrunde liegende Softwarestand im Repository ermitteln.	Internes Projekt; keine Veröffentlichung geplant. Punkt redundant.	MG // 20.03.19

Empfehlung	Bemerkungen	Umgesetzt?
ERM.2: Das Release-Paket enthält oder verweist auf die Nutzer-Dokumentation. Sie besteht zumindest aus Installations-, Nutzungs- und Kontaktinformationen sowie den Release Notes. Im Fall der Weitergabe des Release-Pakets an Dritte außerhalb des DLR, sind die Lizenzbedingungen unbedingt beizulegen.	Internes Projekt; keine Veröffentlichung geplant. Punkt redundant.	MG // 20.03.19
ERM.6: Während der Release-Durchführung werden alle vorgesehenen Testaktivitäten ausgeführt.	Internes Projekt; keine Veröffentlichung geplant. Punkt redundant.	MG // 20.03.19
ERM.9: Vor der Weitergabe des Release-Pakets an Dritte außerhalb des DLR ist sicherzustellen, dass eine Lizenz festgelegt ist, die Lizenzbestimmungen verwendeter Fremdsoftware eingehalten werden und alle erforderlichen Lizenzinformationen dem Release-Paket beigelegt sind.	Internes Projekt; keine Veröffentlichung geplant. Punkt redundant.	MG // 20.03.19
ERM.10: Vor der Weitergabe des Release-Pakets an Dritte außerhalb des DLR ist sicherzustellen, dass die Regelungen zur Exportkontrolle eingehalten werden.	Internes Projekt; keine Veröffentlichung geplant. Punkt redundant.	MG // 20.03.19
Automatisierung und Abhängigkeitsmanagement		
EAA.1: Der einfache Build-Prozess läuft grundlegend automatisiert ab und notwendige manuelle Schritte sind beschrieben. Zudem sind ausreichend Informationen zur Betriebs- und Entwicklungsumgebung vorhanden.	Umgesetzt.	MG // 10.04.18
EAA.2: Die Abhängigkeiten zum Erstellen der Software sind zumindest mit dem Namen, der Versionsnummer, dem Zweck, den Lizenzbestimmungen und der Bezugsquelle beschrieben.	Umgesetzt.	MG // 20.03.19
EAA.10: Das Repository enthält möglichst alle Bestandteile, um den Build-Prozess durchführen zu können.	Umgesetzt, sofern möglich.	MG // 10.04.18