



## Bewertungskriterien Projektdokumentation

### Bewertungskriterien

Die Bewertungskriterien werden entsprechend der Gütestufen mit 1 (Minimum) bis 3 (Maximum) bewertet und in Meilensteinen gruppiert und gewichtet. Fehlt ein Kriterium, wird dieses mit 0 gewertet. Es werden Punkte für zu spät eingereichte Artefakte abgezogen.

Zu jedem Meilenstein werden konkrete, messbare Bewertungskriterien gelistet.

### Übersicht

	<i>Meilensteine</i>	<i>Gewichtung</i>
<i>Anwendungsfälle und Anforderungen</i>		20%
<i>Klassen und deren Beziehungen</i>		20%
<i>Architektur-Design</i>		20%
<i>GUI-Design</i>		10%
<i>Testfälle</i>		10%
<i>Anwendung und API</i>		20%
	<i>Total</i>	<i>100%</i>

## Anwendungsfälle und Anforderungen

Der Projektauftrag wird sinngemäss in ein Anwendungsfalldiagramm übersetzt.		
<i>Das Diagramm genügt weder den formalen Anforderungen noch gibt es den Projektauftrag sinngemäss wieder.</i>	<i>Das Diagramm genügt entweder den formalen Anforderungen ODER gibt den Projektauftrag sinngemäss wieder.</i>	<i>Das Diagramm genügt den formalen Anforderungen und gibt den Projektauftrag sinngemäss wieder.</i>
Jeder Anwendungsfall wird in einer Anwendungsfallspezifikation präzisiert.		
<i>Es fehlen wichtige Merkmale der Beschreibung wie auch ein sinnvoller Ablauf.</i>	<i>Es fehlen wichtige Merkmale der Beschreibung. Der beschriebene Ablauf ist sinnvoll.</i>	<i>Die Beschreibung ist vollständig und gibt einen sinnvollen Ablauf wieder.</i>
Die Anforderungen sind aus den Anwendungsfällen sinnvoll abgeleitet.		
<i>Die Anforderungen sind unvollständig, unverständlich UND nicht aus den Anwendungsfällen abgeleitet.</i>	<i>Die Anforderungen sind teilweise unvollständig und unverständlich ODER sind nicht aus den Anwendungsfällen abgeleitet.</i>	<i>Die Anforderungen sind vollständig, verständlich und aus den Anwendungsfällen abgeleitet.</i>

## Klassen und deren Beziehungen

Die fachlichen Klassen liegen als CRC-Karten vor.		
<i>Es fehlen wichtige Klassen. ODER Bei den dargestellten Klassen fehlen wichtige Aspekte.</i>	<i>Das Diagramm weist Mängel in der Beschreibung der dargestellten Klassen auf.</i>	<i>Das Diagramm ist bezüglich der dargestellten Klassen vollständig.</i>
Der Entwurf ist mit den benötigten technischen Klassen und Schnittstellen in einem Klassendiagramm ergänzt.		
<i>Die technischen Klassen sind erwähnt aber im Entwurf nicht dokumentiert.</i>	<i>Die technischen Klassen sind im Entwurf teilweise dokumentiert.</i>	<i>Die technischen Klassen sind vollständig im Entwurf dokumentiert.</i>
Die Beziehungen der Klassen sind vollständig dokumentiert.		
<i>Die Beziehungen der Klassen sind unklar dokumentiert.</i>	<i>Die Beziehungen der Klassen sind teilweise dokumentiert.</i>	<i>Die Beziehungen der Klassen sind vollständig dokumentiert (inkl. Kardinalitäten)</i>
Der Entwurf weist eine passende Paketstruktur auf.		
<i>Der Entwurf basiert auf einem einzigen Paket (nicht default Paket).</i>	<i>Der Entwurf weist eine (eher zufällige) Paketstruktur auf.</i>	<i>Der Entwurf weist eine begründete Paketstruktur auf.</i>

## Architektur-Design

Die Analyse beinhaltet Aktivitätsdiagramme.		
<i>Es werden nur vereinzelte Aktivitäten durch Diagramme dokumentiert.</i>	<i>Die Aktivitäten werden mehrheitlich durch Diagramme dokumentiert</i>	<i>Die Aktivitäten werden vollständig durch Diagramme dokumentiert.</i>
Das Design beinhaltet Zustandsdiagramme.		
<i>Es werden nur vereinzelte Zustände dokumentiert.</i>	<i>Es werden einige wichtige Zustände dokumentiert.</i>	<i>Kritische Zustände sind vollständig dokumentiert.</i>
Das Design beinhaltet Sequenzdiagramme.		
<i>Es werden nur vereinzelte Abläufe dokumentiert.</i>	<i>Es werden einige wichtige Abläufe dokumentiert.</i>	<i>Kritische Abläufe sind vollständig dokumentiert.</i>
Das gewählte Design unterstützt die einfache Realisierung der Anwendung.		
<i>Design ist nicht der Aufgabe angemessen.</i>	<i>Design weist schwächen auf.</i>	<i>Design ist klar und auf Problemstellung bezogen.</i>

## GUI-Design

Die Mockups geben den Anwendungsaufbau und deren Verwendung wieder.		
<i>Es werden nur einzelne Aspekte abgedeckt.</i>	<i>Es wird eine Mehrheit der Interaktionsmöglichkeiten dargestellt.</i>	<i>Die grafische Oberfläche wird vollständig und nachvollziehbar beschrieben.</i>

## Testfälle

Die gewählten Testfälle geben die Funktionalität der Anwendungsfälle wieder.		
<i>Es werden nur einige Anwendungsfälle abgedeckt.</i>	<i>Es wird eine Mehrheit der Anwendungsfälle abgedeckt.</i>	<i>Es werden alle Anwendungsfälle abgedeckt.</i>
Die gewählten Testfälle geben die Anforderungen wieder.		
<i>Es werden nur einige Anforderungen abgedeckt.</i>	<i>Es wird eine Mehrheit der Anforderungen abgedeckt.</i>	<i>Es werden alle Anforderungen abgedeckt.</i>
Die Testfälle sind durch JUnit abgedeckt.		
<i>Einige wenige Testfälle können durch JUnit nachgewiesen werden.</i>	<i>Eine Mehrheit der geeigneten Testfälle können durch JUnit nachgewiesen werden.</i>	<i>Alle geeigneten Testfälle können mit JUnit-Tests nachgewiesen werden.</i>
Das Testprotokoll ist vollständig.		
<i>Das Protokoll ist bezüglich Soll-Ist-Vergleichs unvollständig.</i>	<i>Das Protokoll enthält korrekte Soll-Ist-Vergleiche. Es fehlen Hinweise betreffend notwendige Massnahmen.</i>	<i>Das Testprotokoll enthält korrekte Soll-Ist-Vergleiche und wo notwendig zu ergreifenden Massnahmen.</i>

## Anwendung und API

Die Klassen sind entsprechend den UML-Diagrammen implementiert und paketierte.		
<i>Im Code finden sich zusätzliche Elemente (Attribute, Methoden, Klassen, usw.), die im Entwurf fehlen.</i>	<i>Die Klassen sind korrekt und vollständig implementiert. Es fehlt die Paketierung.</i>	<i>Alle Vorgaben des Entwurfs sind im Code umgesetzt.</i>
Die Dokumentation jeder Klasse ist vollständig (Autor, Beschreibungen) und die API erstellt.		
<i>Die Beschreibung der Klasse und deren Methoden erklärt deren Anwendung nicht oder zu wenig klar.</i>	<i>Die Beschreibung ist nicht vollständig. Es fehlen Elemente wie Version, usw.</i>	<i>Zu jeder Klasse liegt eine vollständige und aussagekräftige API vor.</i>
Der Code ist verständlich und wartbar.		
<i>Die verwendeten Programmstrukturen wirken zufällig, so dass Änderungen nur schwer zu implementieren sind.</i>	<i>Der Code ist mehrheitlich gut strukturiert. Für Änderungen bedarf es einer vertieften Analyse, um den Code umfassend zu verstehen.</i>	<i>Der Code ist gut strukturiert (funktionale Blöcke) und wo nötig mit erklärendem Kommentar unterlegt. Änderungen können einfach vorgenommen werden.</i>
Die für die Aufgabe typischen Entwurfsmuster sind begründet und dokumentiert.		
<i>Entwurfsmuster werden im Code angewendet, aber nicht dokumentiert.</i>	<i>Die Entwurfsmuster sind im UML-Diagramm ersichtlich.</i>	<i>In der Dokumentation werden die verwendeten Entwurfsmuster begründet. Im UML-Diagramm sind sie ersichtlich.</i>
Die Applikation erfüllt die geforderten Funktionalitäten.		
<i>Es werden nur wenige der Anforderungen erfüllt.</i>	<i>Die Anforderungen werden zu einem grossen Teil erfüllt.</i>	<i>Es werden alle Anforderungen des Auftrags erfüllt.</i>