D08-QM-3 Prozessbeschreibung zur Durchführung von Code-Reviews und Sicherstellung der Dokumentationsqualität

Editor: Siniša Đukanović (Fraunhofer SIT)

Prüfung: Sven Wohlgemuth (TU Darmstadt/CASED)

Typ: [Technischer Bericht]

Projekt: „PersoApp“

Version: 1.0

Datum: 21. Juni 2013

Status: [FREIGABE]

Klasse: [ÖFFENTLICHKEIT]

Datei: D08-QM-3 Prozessbeschreibung zur Durchführung von Code-Reviews und Sicherstellung der Dokumentationsqualität.docx

Zusammenfassung

Das Dokument erläutert die Rollen und den Reviewprozess von Dokumenten und Quellcode für das Open-Source-Software-Projekt „PersoApp“.

Konsortialleitung:
Prof. Dr. Ahmad-Reza Sadeghi und Dr. Sven Wohlgemuth
System Security Lab, TU Darmstadt/CASED, Mornewegstr. 32, 64293 Darmstadt
Tel.: +49-6151-16-75561 Fax: +49-6151-16-72135
E-Mail: persoapp@trust.cased.de Web: https://www.persoapp.de

Nutzungslizenz

Mitglieder des Konsortiums

1. **AGETO Service GmbH**, Deutschland
2. **Center for Advanced Security Research Darmstadt (CASED)**, Deutschland
3. **Fraunhofer Institut für Sichere Informationstechnologie (SIT)**, Deutschland
4. **Technische Universität (TU) Darmstadt**, Deutschland

Versionen

Version	*Datum*	*Beschreibung (Editor)*
0.1	2013-06-10	Initiale Version mit Beiträgen des Fraunhofer SIT
0.2	2013-06-20	Beiträge von AGETO beigefügt
0.3	2013-06-20	Zusammenfügen der Beiträge und Kapitel 1,2,3,4
0.4	2013-06-21	Korrekturen von AGETO
1.0	2013-06-21	Freigabe durch Siniša Đukanović

Autoren

Autoren	*Beiträge*
Philipp Holzinger (Fraunhofer SIT)	Kapitel 1-3, 4.2-7
Stefan Triller (Fraunhofer SIT)	Kapitel 1-3, 4.2-7
Jens Kubieziel (AGETO)	Kapitel 5.1

[1] http://creativecommons.org/licenses/by-sa/3.0/

Inhaltsverzeichnis

1 Ziel und Zweck des Dokumentes

Das vorliegende Dokument beschreibt den Ablauf von Quellcode- und Dokumentenreviews für die Open-Source-Software „PersoApp“. Zu diesem Zweck werden in den nachfolgenden Kapiteln die involvierten Rollen und der zeitliche Ablauf der Reviewprozesse erläutert. Die Aufteilung in separate Prozesse für Code- und Dokumentenreviews ist begründet durch die Unterschiede in Art und Handhabung des zu prüfenden Materials; Java Quellcode gegenüber Textdokumenten.

2 Anwendungsbereich

Die Prozesse sind im Releasezyklus der Open-Source-Software „PersoApp“ zwingend anzuwenden in der Reviewphase der Major-Releases. Ein vollständiges Review des gesamten Quellcodes ist dabei nicht vorgesehen, jedoch können Reviews zu einzelnen Code-Teilen beantragt werden. Der Prozess für Dokumentenreviews wird auf sämtliche Dokumente, die im Releasezyklus der „PersoApp“-Software entstehen, angewandt.

3 Zuständigkeiten und Verantwortlichkeiten

Dieses Dokument wurde in gemeinschaftlicher Zusammenarbeit vom Fraunhofer-Institut für Sichere Informationstechnologie (SIT) und der AGETO Service GmbH erstellt. Die weiteren Konsortialpartner wurden in Form eines Reviews mit der Möglichkeit für Kommentare und Änderungswünsche indirekt daran beteiligt.

4 Prozessbeschreibung

4.1 Code-Reviews zur Sicherstellung der Quellcodequalität

Code Reviews sind ein wichtiger Bestandteil zur Sicherstellung hoher Codequalität. Sie beinhalten die systematische Überprüfung des Quellcodes nach Fehlern. Damit sollen problematische Teile im Code schnell identifiziert und später behoben werden. Studien zeigen, dass die Kosten für die Behebung von Fehlern umso geringer sind, je früher im Entwicklungsprozess der Fehler gefunden wird. Code Reviews leisten hierzu einen wichtigen Beitrag.

Mit dem Standard IEEE Std. 1028-2008 (IEEE Standard for Software Reviews and Audits) existiert ein Regelwerk zur Durchführung von Audits. Die unten stehenden Ausführungen richten sich zum Teil nach den Vorgaben des Standards, aber auch nach praktischen Erfahrungen.

4.1.1 Rollen beim Code Review

Ein Code Review bezieht verschiedene Gruppen bzw. Personen mit ein. Auf der einen Seite ist der Autor des Codestücks. Dieser hat den zu reviewenden Teil des Quellcodes entworfen. Seine Arbeit wird von einem Reviewer geprüft. Die Aufgabe des Reviewers ist es, den Quellcode auf Fehler oder andere Unzulänglichkeiten zu

prüfen. Dabei hat der Reviewer darauf zu achten, dem Autor eine wertschätzende Rückmeldung über den Code zu geben.

Nach den Planungen rekrutieren sich die Autoren überwiegend aus der Community und leisten freiwillige Arbeit am Projekt. Daher ist eine wertschätzende Rückmeldung durch den Reviewer sehr wichtig. Dies erhält die Motivation, weiter am Projekt mitzuarbeiten und bietet dem Autor einen Lerneffekt. Auf der anderen Seite führen missbilligende Kommentare zum Quellcode der Autoren schnell zu Frustration und letztlich zur Demotivation. Wertschätzende Rückmeldungen sind daher als wichtiger Bestandteil des Community Building anzusehen.

Eine dritte Rolle kann in der Etablierung eines Moderators bestehen. Dessen Aufgabe ist es, den Review-Prozess zu begleiten und eine effiziente, zielgerichtete Arbeitsweise zu gewährleisten. Der Review-Prozess sollte sich an vorher festgelegten Codeteilen oder Patches entlang bewegen. Ein Abgleiten in andere zu dem Zeitpunkt nicht relevante Codeteile schmälert unter Umständen das Ergebnis des Review-Prozesses. Daher hat der Moderator darauf zu achten, dass zeitliche Vorgaben eingehalten werden und dass der Code gründlich einem Review unterzogen wird. Letztlich kann die Aufgabe des Moderators auch sein, eine entsprechende Arbeitsatmosphäre für ein Review zu gewährleisten.

Der Standard IEEE Std. 1028-2008 definiert weitere Rollen. Für die Größe des Projektes „PersoApp“ ist es nicht angebracht, diese Rollen einzuführen. Denn dies führt zu einer Erhöhung des administrativen Aufwandes. Eine wesentliche Verbesserung des eigentliches Code Reviews ist hingegen nicht zu erkennen. Sollte das Projekt über die ursprünglich geplanten Grenzen hinaus wachsen, wäre die Position zu überdenken.

4.1.2 Der Review-Prozess

Der Review-Prozess gliedert sich grob in folgende Teile:

1. Autor erstellt den Quellcode.
2. Autor initiiert eine Code Review Request.
3. Reviewer prüft den Code und dokumentiert die Funde.
4. Reviewer gibt das Dokument an den Autor zurück.

Autor ändert ggf. den Code und initiiert eine neue Runde des Review-Prozesses.

Der im ersten Schritt erstellte Quellcode kann aus verschiedenen Quellen herrühren. Im Bug-Tracking-System sind Fehler in der Software, Feature Requests und anderes vermerkt. Daher werden die meisten Code Reviews eine enge Zusammenarbeit mit dem Bug-Tracking-System bedingen: Im System wird ein Fehler gemeldet und das System weist dieser Meldung eine Bearbeitungsnummer zu. Gleiches geschieht bei Feature Requests und anderen Meldungen im Bug-Tracker.

Auf der anderen Seite ist das Versionskontrollsystem eines der wichtigen Werkzeuge für den Entwickler. Dieses bietet die Möglichkeit, dass verschiedene Personen gleichzeitig am Quellcode arbeiten und hat eine Historie über alle eingebrachten Änderungen. Im Laufe der Entwicklung ist es einfach, alte Versionsstände der Software zu betrachten und auch von einem älteren Stand der Software

Entwicklungen vorzunehmen. Diese Eigenschaft ist insbesondere bei verkürzten Release-Zyklen, wie sie im Dokument „D08-QM-2 Operative Planung und Durchführung von Reviews und Release-Updates“ beschrieben sind, wichtig.

Versionskontrollsysteme bieten des Weiteren die Möglichkeit, Entwicklung von neuen Features oder Korrekturen von Fehlern in sogenannten Branches zu erledigen. Dies sind Entwicklungszweige, die parallel zur Hauptentwicklung verlaufen. Auf diese Weise kann eine Neuerung entwickelt werden, ohne ggf. auf gleichzeitige Änderungen im Hauptentwicklungszweig Rücksicht nehmen zu müssen. Bei Fertigstellung des Features bzw. Behebung des Fehlers wird der Branch mit dem Hauptentwicklungszweig zusammengeführt. Auf diese Weise findet parallel die Korrektur mehrerer Fehler oder Entwicklung von Neuerungen statt und die Entwicklung der Software bleibt vergleichsweise übersichtlich und klar.

Daher ist es unbedingt zu empfehlen, neue Features oder Fehlerkorrekturen innerhalb von Branches bzw. dem Äquivalent im jeweiligen Versionsverwaltungssystem zu entwickeln. Die Versionsverwaltung weist jeder Änderung eine eindeutige Nummerierung zu. Ein Branch kann anhand dieser Nummer oder anhand einer vom Entwickler gewählten Bezeichnung („Branch A“) identifiziert werden. Die von der Versionsverwaltungssoftware erzeugte Nummerierung oder der selbst gewählte Name des Branches sollte im Bug-Tracking-System referenziert werden.

- Wenn der Autor im zweiten Schritt einen Code Review Request initiiert. Dieser sollte die folgenden Angaben enthalten:
- Beschreibung des zu prüfenden Teils, d.h. insbesondere Angaben über die betroffenen Klassen und eine kurze Angabe über den Zweck der Änderung
- Referenz auf den Eintrag im Bug-Tracking-System
- Name oder Identifikationsnummer des Branches im Versionsverwaltungssystem. Falls nicht mit Branches gearbeitet wird, so ist die Revision vor Beginn der Änderung und die Revision nach Ende der Änderung anzugeben.
- Gegebenenfalls ist zur Erleichterung des Reviews der konkrete Patch zur Verfügung zu stellen. Dies sollte insbesondere dann geschehen, wenn nicht mit Branches gearbeitet wurde oder zwischen dem eigentlichen Inhalt des Reviews mehrere Änderungen liegen, die keinen Bezug zum Review haben.
- Datumsangabe bis zu dem der Code einem Review unterzogen werden soll

Der Reviewer nimmt den Request entgegen und bearbeitet den Review. Sollte der Review nicht bis zum angegebenen Datum zu realisieren sein, ist der Autor hiervon zu unterrichten.

Es sind keine speziellen Werkzeuge für den Code Review vorgeschrieben. Der Reviewer ist frei in seiner Entscheidung, ob er Code-Review-Werkzeuge einsetzt oder auf anderen Wegen den Code prüft.

Die während des Code Reviews gefundenen Mängel sind zu dokumentieren. Es wird an dieser Stelle dem Reviewer überlassen, in welcher Form die Dokumentation geschieht. Bei Änderungen direkt im Quellcode empfiehlt es sich, direkt an den

entsprechenden Stellen im Code einen Kommentar zu hinterlassen. Bei konzeptionellen Änderungen ist es hingegen eher empfehlenswert, das Bug-Tracking-System oder den Code Review Request zur Dokumentation und Rückmeldung zu nutzen. Sollten direkt im Quellcode Änderungen oder Kommentare vorgenommen werden, so sollte dies wieder in das Versionskontrollsystem eingecheckt werden oder über einen Patch an den Autoren zurückgehen.

Es wird empfohlen, die Vorschläge so konkret wie möglich zu gestalten. Dies hilft dem Autoren schnell darauf einzugehen und entsprechende Korrekturen einzubringen.

Im vorletzten Schritt gibt der Reviewer seine Dokumentation über die Funde zurück an den Autor. Dabei sollte das Bug-Tracking-System genutzt werden.

Der Autor prüft letztlich die Kommentare und versucht, die Vorschläge aus dem Review umzusetzen. Sobald die Umsetzung abgeschlossen ist, ist ggf. ein neuer Review-Prozess einzuleiten. Sollten im Review-Prozess nur triviale Änderungen festgestellt worden sein, so kann eine erneute Initiierung unterbleiben.

4.1.3 Empfehlungen für den Reviewer

Die Hauptaufgabe des Reviewers ist es, den Quellcode auf Defekte, Fehler und andere Unzulänglichkeiten zu prüfen. Dies soll die Qualität des Codes erhöhen und das Auffinden von Fehlern früh im Entwicklungsprozess hilft, die Kosten für die Entwicklung einer Software gering zu halten.

Um den Review-Prozess effektiv zu gestalten, sind untenstehende Hinweise gegeben. Die Hinweise sind einer Studie von entnommen, die die Firma Smart Bear Systems im Jahr 2006 bei Cisco Systems durchgeführt hat.[2]

Die Menge an Code, der zu prüfen ist, sollte 200 Zeilen nicht überschreiten. Die obige Studie fand heraus, dass ein mehr an Code zu proportional weniger gefundenen Defekten führt. Es wird vermutet, dass die Konzentration nachlässt und daher nicht jeder Fehler gefunden wird.

Der Zeitraum für einen Code Review sollte etwa eine Stunde betragen. Bei den untersuchten Reviews, die länger als eine Stunde dauerten, nahm die Effektivität stark ab. Dies ist mit Ermüdungseffekten zu begründen.

Der Autor kann seinen Quellcode annotieren. Eine Annotierung soll den Reviewer durch die Änderungen leiten und erklären, warum diese durchgeführt wurden. In der Studie wurde eine Verbesserung der Fehlerrate festgestellt. Jedoch sind die Gründe im Unklaren. Auf der positiven Seite kann es sein, dass der Autor seinen Code durch die Annotierung selbst reviewt und Fehler findet. Auf der negativen Seite kann es auch sein, dass der Reviewer sich zu sehr auf die Annotation verlässt und daher die Aufgabe nicht gut genug erledigt. Daher kann die Annotation eine Hilfe sein. Der Reviewer sollte den Code dennoch mit einem kritischen Blick bewerten.

---

[2] Smart Bear Systems: Best Kept Secrets of Peer Code Review. http://smartbear.com/resources/whitepapers/best-kept-secrets-of-peer-code-review

Es sollte eine angenehme Atmosphäre etabliert werden, wo das Finden von Fehlern positiv gesehen und gehandhabt wird. Dies wurde bereits mit den obigen Bemerkungen über Rückmeldungen versucht zu initiieren.

4.2 Dokumentations-Reviews zur Sicherstellung der Dokumentationsqualität

Die Erstellung und das Review von Dokumentationen erfolgt – mit Ausnahme von „Javadoc“ – gemäß definierter Vorgehensweisen. Anhand von Abbildung 1 wird zunächst ein Überblick über den Erstellungs- und Änderungsprozess gegeben.

Dieser Prozess umfasst die folgenden Schritte:

1. Der Bearbeiter editiert das Dokument. Handelt es sich um eine initiale Erstellung, so ist der Status des Dokumentes als „Entwurf“ zu kennzeichnen. Handelt es sich hingegen um eine Überarbeitung, dann ist der Status „Revision“ anzuzeigen.
2. Im Anschluss an die Bearbeitung erstellt der Bearbeiter ein „Review“-Ticket im Issuetracker des „PersoApp“-Projektes, wodurch der Reviewprozess gestartet wird. Das Dokument nimmt hierdurch den Status „Prüfung“ an.
3. Im Zuge des Reviews könnten die Gutachter Mängel feststellen, für deren Behebung der Bearbeiter jeweils ein Issue-Ticket zugewiesen bekommen würde. In diesem Fall wäre das Review zunächst abgeschlossen und das Dokument würde für eine Überarbeitung in den Status „Revision“ zurückfallen. Sollte das Review hingegen keine Mängel aufzeigen, so geht das Dokument in die Freigabeprüfung
4. Das Dokument wird daraufhin geprüft, ob seiner Freigabe zugestimmt werden kann. Solange nimmt es den Status „Freigabe“ an. Sollte es für eine Veröffentlichung einer Überarbeitung bedürfen, so geht das Dokument erneut in die „Revision“. Andernfalls kann das Dokument veröffentlich werden.
5. Das Dokument wird gemäß seiner Vertraulichkeitsstufe veröffentlicht. Entsprechend erhält es den Status „Veröffentlicht“.

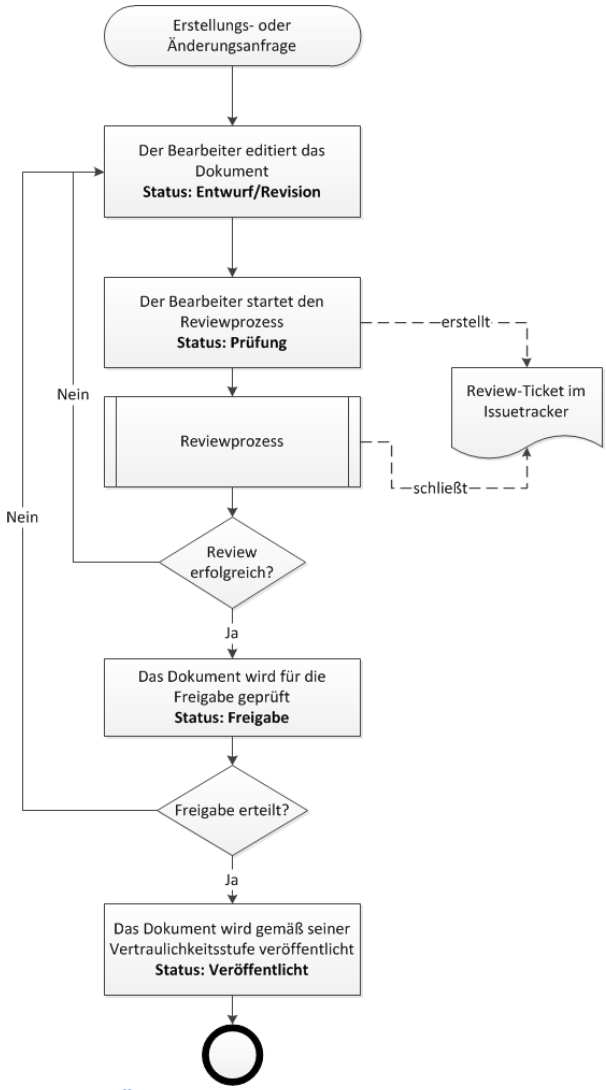

Abbildung 1- Der Erstellungs- und Änderungsprozess von Dokumentationen (ausgenommen Javadoc)

Der Reviewprozess, auf den der Erstellungs- und Änderungsprozess in Abbildung 1 verweist, wird im Folgenden anhand von Abbildung 2 erläutert.

1. Durch Erstellung eines „Review“-Tickets wird ein Gutachter angewiesen, das Dokument zu prüfen. Die Zuordnung des Tickets zu einem Gutachter erfolgt entweder automatisch durch den Issuetracker, oder wird vom Qualitätsbeauftragten durchgeführt.
2. Nach der Zuweisung prüft der Gutachter das Dokument anhand der Dokumenten-Review-Checkliste. Falls alle Prüfkriterien erfüllt werden können, gilt das Review als erfolgreich. Sollten jedoch Mängel gefunden worden sein, so erstellt der Gutachter entsprechende Issue-Tickets und weist sie dem Bearbeiter der Dokumentation zu, um eine Nachbearbeitung anzustoßen.
3. Das „Review“-Ticket wird geschlossen.

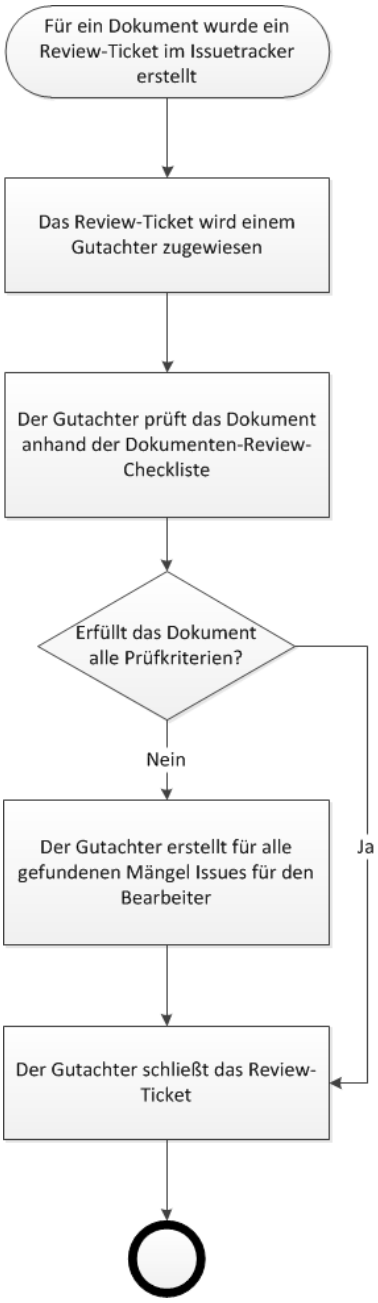

Abbildung 2 - Der Reviewprozess für Dokumente

Kopplung an die Softwaremodul-Entwicklung

Zur Dokumentation des Quelltextes wird „Javadoc“ verwendet. Mit Hilfe von „Javadoc“ dokumentieren die Entwickler ihren Quelltext, indem sie ihre Methoden annotieren. Diese Annotation ist ein spezieller Kommentar vor jeder als „public“ deklarierten Methode der die Funktion der Methode, ihre Eingabeparameter, sowie ihr Ergebnis beschreibt. Entwicklungsumgebungen, wie z.B. Eclipse, bereiten diese „Javadoc“-Annotationen für die Entwickler auf. Jedes Mal wenn ein Entwickler eine annotierte Methode verwenden will, bekommt er diese Beschreibungen angezeigt.

Im Zuge der Qualitätssicherung ist es nötig, dass die Annotationen stets aktuell gehalten werden. Die grundsätzliche Funktion von erstellten Methoden ändert sich

selten. Nimmt ein Entwickler dennoch Änderungen vor, so ist er angehalten die „Javadoc“- Annotation ebenfalls zu aktualisieren. Da der Quelltext mit einem Versionskontrollsystem verwaltet wird, werden Änderungen auch sofort an die anderen Entwickler propagiert. Die IDE der anderen Entwickler zeigt automatisch die neuste Dokumentation nachdem der Quelltext aus dem Versionskontrollsystem aktualisiert wurde.

Ein separates Review der „Javadoc“- Annotationen ist daher nicht erforderlich.

Checkliste zur Reviewkontrolle

Um den Review Prozess transparent zu gestalten, befindet sich im Anhang A eine Checkliste, die im Rahmen eines Reviews einer Dokumentation ausgefüllt wird. Diese Checkliste umfasst die folgenden Punkte:

Gibt es offene Dokumentations-Issues im Issue Tracker?

Der Reviewer durchsucht den Issue-Tracker des Projektes nach Einträgen die die zu prüfende Dokumentation betreffen. Nur wenn dort keine offenen, d.h. unbearbeiteten Issues mehr zu finden sind wird dieser Punkt abgehakt.

Ist ein Bezugspunkt angegeben?

Bezugspunkte in Dokumentationen dienen dazu, dass der Reviewer auf einen Blick sehen kann auf welches Modul die Dokumentation sich bezieht. Der Reviewer kann so ggf. prüfen ob die Dokumentation mit dem Bezugspunkt in den Punkten Aktualität und Integrität übereinstimmt.

Ist ein Verwendungszweck angegeben?

Ohne einen Verwendungszweck kann der Reviewer nicht feststellen ob die Dokumentation in ihrem Umfang diesem gerecht wird.

Ist eine Zielgruppe angegeben?

Um die Verständlichkeit der Dokumentation prüfen zu können, muss der Reviewer sich in die Lage eines Mitglieds der Zielgruppe versetzen. Ohne diese Angabe ist es schwierig zu beurteilen ob die Dokumentation der Zielgruppe angemessen geschrieben wurde.

Enthält die Dokumentation Rechtschreib- bzw. Grammatikfehler?

Dokumentationen sollten stets frei von Rechtschreib- bzw. Grammatikfehlern sein. Hierzu kann der Reviewer ggf. auch automatische Prüfprogramme verwenden.

Entspricht die äußere Form den Richtlinien aus „D05-QM Dokumentationsanforderungen“?

Um ein einheitliches Layout für alle Dokumente des Projektes gewährleisten zu können wurden in dem o.g. Dokument Richtlinien zur äußeren Form von Dokumenten festgelegt. Die Dokumentation hat die ihrem angegebenen Typ entsprechende Form einzuhalten.

Vollständigkeits- und Verständlichkeits-Test durch ausprobieren der Dokumentation

Um festzustellen ob die Dokumentation verständlich und vollständig ist, wird sie mind. einer Testperson vorgelegt, die sie ausprobiert. Die Dokumentation könnte z.B. eine Installationsanleitung für Windows 7 Anwender sein. Wenn die Testperson mit Hilfe der Anleitung die Anwendung installieren kann, ist sie vollständig und verständlich. Andernfalls muss nachgebessert werden.

5 Interne und Externe Anforderungen

Interne Anforderungen betreffen die Veröffentlichung von Quelltext und Dokumenten, die ausschließlich gemäß ihrer Vertraulichkeitsstufe veröffentlicht und weitergegeben werden dürfen. Dies ist bereits bei der Erstellung zu beachten.

Externe Anforderungen betreffen im Besonderen die Verwendung und Einbeziehung von externen Softwareprodukten, die im Rahmen dieses Projekts ausschließlich gemäß der vorliegenden Lizenz verwendet werden dürfen.

6 Abläufe

Durch eine sorgfältige Prüfung der hier getroffenen Festlegungen wurde sichergestellt, dass diese Prozessbeschreibungen praktikabel und angemessen sind. Sollte sich dennoch im Projektverlauf die Notwendigkeit für eine Anpassung ergeben, dann ist dies einerseits an alle Projektteilnehmer explizit und verständlich zu kommunizieren, zum anderen gilt es die entsprechende Änderung detailliert festzuhalten und nachvollziehbar zu dokumentieren.

7 Mitgeltende Dokumente

- „D05-QM Dokumentationsanforderungen“

Anhang A

Dokumenten-Review Checkliste

	Erledigt
Gibt es offene Dokumentations-Issues im Issue Tracker?	
Ist ein Bezugspunkt angegeben?	
Ist ein Verwendungszweck angegeben?	
Ist eine Zielgruppe angegeben?	
Enthält die Dokumentation Rechtschreib- bzw. Grammatikfehler?	
Entspricht die äußere Form den Richtlinien aus D05-QM Dokumentationsanforderungen?	
Vollständigkeits- und Verständlichkeits-Test durch ausprobieren der Dokumentation	

Notizen:	