ZEMAN_SOFTWAREPLAN_V7.DOCX

Honorarnotenverwaltungsapp

11.06.2022

SPENGERGASSE
ausbildung mit zukunft

Zeman Laura
ZEM19778@spengergasse.at

ZEMAN Laura
4AHBGM

Prüfungen und Änderungen

Geprüft von:

Datum:

Unterschrift: _____

Freigegeben von:

Datum:

Unterschrift: _____

Gültigkeit

Von:

Bis:

Änderungshistorie

Versionsnummer	Änderungsdatum	Person	Änderung
1.0	18.10.2021	Laura Zeman	Erstversion
2.0	23.11.2021	Laura Zeman	Zweitversion
3.0	06.12.2021	Laura Zeman	Drittversion
4.0	20.01.2022	Laura Zeman	Viertversion
5.0	21.02.2022	Laura Zeman	Erweiterung um GIT-Workflow
6.0	02.05.2022	Laura Zeman	Erweiterung um 5.1.5 und 5.1.6
7.0	23.05.2022	Laura Zeman	Erweiterung um 5.1.7

Zielsetzung und Gültigkeitsbereich

Zielsetzung (Definition des SW-Entwicklungsprozesses)

Zielsetzung:

Der vorliegende Softwareentwicklungsplan beschäftigt sich mit dem Entwicklungsprozess einer Honorarnotenverwaltungs-Software. Es soll den gewünschten Ablauf möglichst genau beschreiben, um den eigentlichen Entwicklungsprozess so klar wie möglich zu strukturieren beziehungsweise zu beschreiben. Um einen gewissen Standard beizubehalten, wurden zur Erstellung dieses Softwareentwicklungsplanes die EN 62304 5.1.1, die EN 62304 5.1.4 sowie die MDR (Medical Device Regulation) miteinbezogen.

Geltungsbereich

Geltungsbereich:

Der vorliegende Softwareentwicklungsplan gilt für die Entwicklung sowie Wartung der Honorarnotenverwaltungsapplikation durch die Entwickler*innen. Es ist wichtig anzumerken, dass diese Software nicht im medizinischen Bereich gilt. Das heißt, dass es nicht als Medizinprodukt oder als in der Medizin angewandtes Produkt zu klassifizieren ist, da nicht die entsprechenden Standards beziehungsweise Anforderungen an die Software erfüllt sind.
Das Dokument umfasst keine Softwarelebenszyklusschritte nach Freigabe des Produktes wie beispielsweise die Implementierung, Konfiguration, Außerbetriebnahme oder Entsorgung.

Prozessbeschreibungen

Prozessnummer	Prozessbeschreibung	Unterlagen	Verantwortlicher
1	Aneignen von Grundkenntnissen in der Programmierung mit Ruby (on Rails)	Codecadamy-Kurs, Youtube-Videos	HAJOSSY, RANKOVIC, ZEMAN
2	Organisatorisches klären (Aufgabenteilung, Verantwortungsbereiche, Zeitrahmen,…)	Wird zum gegebenen Zeitpunkt in einem Word festgehalten	HAJOSSY, RANKOVIC, ZEMAN
3	Die geforderten Anforderungen des Auftraggebers sollen erfüllt werden, indem eine Honorarnotenverwaltungs-App auf Ruby (Entwicklung erfolgt laut SW-Plan) programmiert wird	SW-Entwicklungsplan	ZEMAN
4	Die Entwicklung eines Konzeptes, welches das optische Layout der Honorarnotenverwaltungsapp festlegt. Hier soll ein besonderes Augenmerk auf die Wünsche des Auftraggebers gelegt werden.	Mündliche Absprache	HAJOSSY, RANKOVIC, ZEMAN
5	Nachdem die Software von der Entwicklerin kontinuierlich während der Programmierung getestet wurden und als fehlerfrei erachtet wurden, sollen die anderen Teammitglieder, getrennt voneinander, die SW auf Userfreundlichkeit und Fehlerlosigkeit überprüfen	Festhaltung der Ergebnisse in Form von Screenshots, welche ebenfalls in Github gespeichert werden sollen.	HAJOSSY, RANKOVIC

Noch zu klären ist:

- **Veröffentlichung:**
  Nach Abschluss der Testphase soll die Honorarnotenverwaltungsapp vom Auftraggeber abgenommen und den Endnutzern zur Verfügung gestellt zu werden.

EN 62304(a) PROCESSES

Ein Prozess soll einen Ablauf mit der höchstmöglichen Genauigkeit beschreiben. Dies kann textuell oder grafisch stattfinden. Wichtig ist, dass man im Endeffekt so, auf der ganzen Welt diese Abläufe nachvollziehen kann. “Ein Prozess beschreibt einen Ablauf möglichst eindeutig (kann mehrere Seiten lang sein) Entweder grafisch oder textuell. Welche Prozesse wird es geben? => Liste in SW-Entwicklungsplan aufnehmen“

EN 62304(b) DELIVERABLES

Hier beschreibt man, was die fertiggestellte und gelieferte Software beinhalten soll.“ Was entsteht bei dem Prozess? Dokumente, ausgefüllte Formulare, auslieferbare Software als… oder eine Liste an Anforderungen als UserStories? Gehört zu jedem Prozess definiert.“

EN 62304(c) TRACEABILITY

Traceability (zu Deutsch Rückverfolgbarkeit) heißt, dass sämtliche Ergebnisse eines SW-Tests nachvollzogen werden können, beziehungsweise einer Anforderung zugewiesen werden können. „Die Rückverfolgbarkeit bedeutet, dass jedes Ergebnis eines Softwaretests einer Anforderung zugeordnet werden kann. Z.B. zeigt ein Pulsmesser maximal einen Puls von 200, wobei der Puls bei 220 liegt. Es muss bei dem Fehler erkennbar sein, zu welcher Anforderung das Feature gehört. Praktisch wäre eine Kette: Anforderung – SW-Artefakte - Test Wie kann man das erreichen? => Antwort in den SW- Entwicklungsplan“

EN 62304(d) CONFIGURATION-Management

Das Konfigurationsmanagement befasst sich mit diversen die Konfiguration betreffenden Parametern, wie der Versionsnummer der Developmenttools, Speicherorten der einzelnen Softwareersionen und so weiter. “Konfigurationsmanagement – Was verwenden wir in welcher Version, wo ist es zu finden, wo wird etwas gespeichert und zu welchem Zeitpunkt? Z.B. Entwicklungstools aber auch Libraries, die wir verwenden und auch die Softwareversionen, die wir ausliefern bzw. testen. Welche Tools helfen uns dabei? Wie ist der Prozess => Definieren im SW-Entwicklungsplan“

EN 62304 5.1.4 Standards, Methods and Tools

Standards:
Standards, die bei der Erstellung der medizinischen Software beachtet werden müssen, sind: EN 62304 und Medical Device Regulation (insbesondere Regel 11, sowie die Klassifizierung von Medizinprodukten und deren Anforderungen) sowie die EN 60601-1-8 für allgemeine Festlegungen einschließlich medizinischen Systemen:

- **Kommunikationsstandard:**
  Bei Schnittstellen zwischen den Systemen werden Kommunikationsstandards, wie HL7 (Health Level Seven) verwendet, einem Standard speziell für das Gesundheitswesen. Wenn die Software im Krankenhaus in bereits bestehenden Systemen eingesetzt wird, so muss HL7 Version 2 verwendet werden. Werden aber mit sektorübergreifenden XML-basierten Nachrichten ausgetauscht, so verwendet man HL7 Version 3, welche auf RIM (Referenz-Informations-Modell) basiert und CDA (Clinical Document Architecture) für den Inhalt medizinischer Dokumente. Besonders wichtig sind bei RIM die vier Basisklassen (Entität – entity, Rolle – role, Partizipation – participation, Aktivität – activity). Verbindet werden diese mit Beziehungen von Rollen untereinander (role relationship) und Beziehungen von Aktivitäten (act relationship). Für die Internetkommunikation wird HTTPS (HyperText Transfer Protocol Secure) verwendet.
- **Begriffsstandard:**
  Betreffend der Terminologie werden Standards, wie Snomed-CT, LOINC und ICD verwendet, damit die Begriffe einheitlich sind und auch von allen verstanden werden.

Methoden:
Unser Team hat sich dafür entschieden, sich an dem Wasserfallmodell zu orientieren, da so alle Abläufe sowie Deadlines klar definiert sind. Außerdem soll vor dem Ende dieser Softwareentwicklung der Code durch eine 3. Person reviewed werden. Aus vergangenen Erfahrungen hat das Team gelernt, sich an folgende Verhaltensgrundregeln zu halten:

- Regelmäßige Kommunikation bezüglich der Softwareentwicklung (min. 1x pro Woche)
- Regelmäßige Statusupdates (1x alle zwei Wochen)
- Gegenseitiger Respekt gegenübereinander und gegenüber der Arbeit der anderen Teammitglieder
- Gemeinsames Arbeiten an Problemen
- Wertschätzung der anderen Teammitglieder

Tools:
Um mögliche Fehlerquellen zu minimieren, hat sich das Team darauf geeinigt, dass jedes Mitglied folgende technische Voraussetzungen erfüllen soll:

- Entwicklungsumgebung: IntelliJ IDEA 2021.1.1
- Betriebssystem: Windows
- Dokumentationstools:
    - Zeiterfassung: MS-Excel
    - Gesprächsprotokolle: MS Word
- Datenbankbearbeitungssystem: MySQL-Workbench
- Speicherung des Codes: GitHub
- Versionen: Jeweils die aktuellsten Versionen zum Zeitpunkt des Beginns der Softwareentwicklung

Anleitung für neue Mitarbeiter

Für neue Mitarbeiter des Projekts, sind sämtliche technischen Anforderungen unter dem Punkt „EN 62304 5.1.4 Standards, Methods and Tools“

Glossar

Deliverables:
Dokumente / Resultate aus einem Prozess

Traceability:
Rückverfolgbarkeit

5.1.5 GIT Workflow

Nach längerer Überlegung wurde entschieden, dass sich das Team beziehungsweise der Projekt-Workflow am Feature Branch Workflow orientieren soll. Die Entscheidung wird wie folgt begründet:

- Es gibt einen main branch, auf welchen erst gepushed wird, wenn der geänderte Code absolut keine Fehler produziert (nicht bei anderen Teilen des Projekts, nicht in sich selbst)
- Jedes Feature des Projekts wird auf einem eigenen branch programmiert
- Daraus resultiert der Vorteil, dass man nicht akribisch Acht geben muss, dass man den restlichen Code beschädigt
- Außerdem ist dann im Workflow-Diagramm ersichtlicher, was in welchem wann von wem geändert wurde

Ergänzung (11.06.2022):
Nachdem die SW fertiggestellt wurde, haben wir festgestellt, dass es bei einer SW dieser Größe kosten-nutzentechnisch nicht effizient ist, wenn für jedes Feature ein separater Branch erstellt wird. Daher wurde zunächst alles lokal programmiert und getestet und anschließend wurden die Änderungen auf Github gepushed.

Das Repository soll zentral liegen, damit jede*r Entwickler*in darauf zugreifen kann. Allerdings sollen Berechtigungen so verteilt werden, dass sämtliche push-requests auf den main branch vom Projektleiter abgesegnet beziehungsweise genehmigt werden müssen, um das Risiko, der Hochladung eines Fehlers mit Auswirkungen auf das gesamte Projekt zu minimieren.

Es sollen keine lokalen branches für jede Anforderung erstellt werden, da hierdurch die Möglichkeit geschaffen wird, dass Änderungen falsch beziehungsweise an eine andere Stelle als gewünscht hochgeladen werden, was zur Konsequenz hat, dass diese Daten dann verloren gehen. Zwar kann man dies umgehen, indem lokale branches erst nach einer gewissen Zeit gelöscht werden, allerdings ist schlichtweg das Aufwand-Nutzen Verhältnis besser ist, wenn keine lokalen branches erstellt werden.

Nun stellt sich die Frage wie oft gepushed und gepulled werden?

- Gepushed:
    - Nach jeder **FUNKTIONIERENDEN!** Änderung. Das ist der Fall wenn:
        - Der neue Code-Part die alte Version nicht zum abstürzen bringt
        - Das Programm nach wie vor lauffähig ist
- Gepulled:
    - Zu Beginn eines jeden Weiterarbeitsprozesses

Erweiterung 5.1.6

Verifizierung von Deliverables + Meilensteine

Deliverables wurden unter dem Punkt EN62304(b) definiert. Um die Wahrscheinlichkeit des Auftretens eines Fehlers während der Nutzung unserer Software möglichst gering zu halten, ist es essentiell jede größere Änderung zu verifizieren. Das bedeutet nach dem hinzufügen einer neuen Funktion, ist jedes Mal eine Verifizierung vorzunehmen.

Vorgehensweise bei Verifizierung

Vor der endgültigen Abnahme unseres Produkts durch den Kunden, haben drei Schritte zu erfolgen:

- **Reviews durch unser Team (1. Runde)**
  - Nachdem die SW kompilierbar ist
  - Von jedem fachkundigen Entwickler durchzuführen
  - Funktionalität aller Aspekte des Programms wird überprüft
- **Reviews durch unser Team (2. Runde)**
  - Nachdem die SW die ersten Tests erfolgreich durchlaufen ist
  - Von jedem nicht-fachkundigen Mitarbeiter durchzuführen
  - Funktionalität aller Aspekte des Programms wird überprüft
  - Userfreundlichkeit wird überprüft

Akzeptanzkriterien

- Verifizierung erfolgreich durchlaufen
- Kundenanforderungen im realistischen Rahmen erfüllt

Erweiterung 5.1.7

Testfälle enthalten eine Reihe von Aktionen, die wir durchführen, um eine bestimmte Funktion oder Funktionalität unserer Applikation zu bestimmen. Sie sind ein essentieller Teil der SW-Entwicklung und sichern die Funktionalität wichtiger Komponenten. Wichtige Bestandteile der Beschreibung eines Testfalls sind:

1. die Vorbedingungen, die vor der Testausführung hergestellt werden müssen,
2. die Benennung des Testobjekts und der Spezifikationen, auf die sich der Testfall bezieht,
3. die Eingabedaten für die Durchführung des Tests,
4. die Handlungen, die zur Durchführung des Testfalls notwendig sind,
5. die erwarteten Ergebnisse und/oder Reaktionen des Testobjektes auf die Eingaben,
6. die erwarteten Nachbedingungen, die als Ergebnis der Durchführung des Testfalls erzielt werden.
7. die Prüfanweisungen, d. h. wie Eingaben an das Testobjekt zu übergeben sind und wie Sollwerte abzulesen sind.

Testfälle lassen sich in *Positiv-Testfälle* und *Negativ-Testfälle* unterteilen:

1. In Positivtests wird das Verhalten des Testobjekts infolge gültiger Vorbedingungen und Eingaben überprüft.
2. In Negativtests (auch Robustheitstest genannt) wird das Verhalten des Testobjekts infolge ungültiger Vorbedingungen und/oder Eingaben überprüft.