Bewertungsrichtlinien

Projektarbeit im Rahmen von Anwendungsorientierte Programmierung für Ingenieure

3. April 2018

Die Bewertung eines Projekts gliedert sich in die vier Teilbereiche Planung, Sprintdokumentation, Implementierung und Anleitung. Für das Bestehen des Projekts ist eine Bewertung von mindestens 50 % erforderlich. Die Gesamtbewertung gliedert sich wie folgt:

Planung

- Backlog mit User-Stories

  Eine User-Story beinhaltet eine Funktion der zu entwickelnden Software. Sie ist definiert über die Aktion einer Rolle und dem Nutzen dieser Aktion. Eine User-Story soll kurz und prägnant formuliert werden. Eine User-Story gilt als verhandelbar und soll in dem Entwicklungsprozess als Diskussionsgrundlage dienen. Die Formulierung einer User-Story kann nach dem folgenden Schema erfolgen:
  Als <*Rolle*> möchte ich eine <*Aktion*> ausführen, weil sie diesen <*Nutzen*> bringt.
  Ein ausformuliertes Beispiel kann wie folgt aussehen:
  Ich als Autofahrer möchte beim Parkvorgang den Abstand zu den angrenzenden Fahrzeugen messen, um eine Kollision mit anderen Fahrzeugen zu vermeiden.
  Damit im Entwicklungsprozess überprüft werden kann, ob eine User-Story erfolgreich abgeschlossen wurde, besitzt jede User-Story Zielkriterien. Die Zielkriterien sollen voneinander unabhängig und messbar sein. Für das angegebene Beispiel können die Zielkriterien wie folgt aussehen:

  - Parkvorgang wird automatisch erkannt
  - Der Abstand zu den Fahrzeugen vor und hinter dem eigenen Fahrzeug wird gemessen
  - Der Abstand zu den Fahrzeugen wird als Piepton wiedergegeben, der sich in seiner Frequenz reziprok proportional zu dem Abstand verhält
  - Bei der Unterschreitung eines fest definierten Abstandes ertönt ein durchgängiger Ton

  Jedes Gruppenmitglied soll für die Projektmappe **eine User-Story** mit jeweils **vier Zielkriterien** erstellen. Die User-Stories sollen voneinander abgrenzbar sein. Legt beispielsweise ein Studierender in seiner User-Story die Abstandsmessung nach vorne aus und ein anderes Gruppenmitglied legt die Abstandsmessung nach hinten aus, werden diese als nicht voneinander abgegrenzt gewertet. Die Punkte für eine User-Story werden auf beide Studierende verteilt.

  - User-Story enthält Rolle — Individual
  - User-Story enthält eine nachvollziehbare Aktion — Individual
  - User-Story enthält Nutzen — Individual
  - Zielkriterien vorhanden — Individual
  - Zielkriterien messbar — Individual
  - User-Story wurde umgesetzt — Gruppe

- Projektschema

Das Projektschema soll die Struktur der Software und - falls vorhanden - der Hardware sowie deren Interaktion darstellen. In dem Schema sollen die Funktionalitäten der Software möglichst modular abgebildet werden. Gleichzeitig sollen die Zielkriterien aller User-Stories in dem Schema auffindbar sein (zum Beispiel über Systemgrenzen, Beschriftung, etc.).

- Vollständigkeit
- Konsistenz und adäquate Darstellung Gruppe
- Verknüpfung mit Zielkriterien der User-Stories Gruppe
- Aktualität und Übereinstimmung mit der finalen Version des Projekts Gruppe

Sprintdokumentation

Ein Sprint ist ein begrenzter Entwicklungszeitraum, in dem ein Inkrement der Software entwickelt werden soll. Das Ergebnis eines Sprints soll eine lauffähige Software sein, die Teile der User-Stories oder einer User-Story enthält. In der Projektmappe sollen **4 Sprints** dokumentiert werden.

- Sprintplanung

  In der Sprintplanung sollen aus einer oder mehreren User-Stories Aufgaben generiert werden. Die Aufgaben sollen so gewählt werden, dass am Ende des Sprints ein lauffähiges Programm entsteht. In jedem Sprint sollte von jedem Studierenden mindestens ein Zielkriterium einer User-Story erfüllt werden. Nach der Generierung der Aufgaben sollen von den Gruppenmitgliedern die Abhängigkeiten dieser Aufgaben identifiziert werden. Parallele Vorgänge sollen erkannt werden, damit diese gleichzeitig von unterschiedlichen Gruppenmitgliedern bearbeitet werden können. Die im folgenden Abschnitt erwähnten Bug-Reports müssen spätestens im letzten Sprint als Aufgabe adressiert werden.

  - Es wurde für jedes Gruppenmitglied mindestens eine Aufgabe generiert. Etwaige Abhängigkeiten der Aufgaben untereinander sind erfasst. Gruppe
  - Zielkriterien wurden adressiert und einem Gruppenmitglied zugewiesen. Bepunktung einmalig bei Erfüllung des Zielkriteriums. Individual
  - Ein Bug-Report wurde einem Gruppenmitglied zugewiesen. Bepunktung einmalig bei Behebung des Fehlers Individual

- Sprintmeeting

  Sprintmeetings sollen wöchentlich in der API-Übung (Dienstags 16:45-18:15 Uhr PK15.1) stattfinden. In diesen Meetings wird festgehalten, welches Gruppenmitglied welche Aufgabe bearbeitet hat, welche Aufgaben in der kommenden Woche bearbeitet werden sollen und welche Probleme aufgetreten sind. Dieses Meeting soll hierbei zwingend auf 15 Minuten begrenzt werden. Die Punkte dieses Meetings sollen in einem Protokoll zusammengefasst werden.

  - Bearbeitete Aufgaben und deren Zwischenstand wurden niedergeschrieben Individual
  - Geplante Aufgaben für die Folgewoche wurden niedergeschrieben Individual

- Sprintabschluss

  Im Sprintabschluss sollen die Aufgaben für den abzuschließenden Sprint geprüft werden. Es soll dokumentiert werden, ob und unter welcher Revisionsnummer die Aufgaben abgeschlossen wurden. Falls die Aufgaben nicht abgeschlossen wurden, sind die Gründe zu nennen. Des Weiteren soll ein kurzer Überblick über den aktuellen Stand des Projekts erstellt werden. Welche User-Stories und Zielkriterien sind abgeschlossen? Falls sich Änderungen am Backlog ergeben, sollen diese in dem Sprintabschluss festgehalten werden. Neben der Aufgabenprüfung soll auch eine Bewertung der Sprintplanung erfolgen, hieraus können sich Änderungen für den Folgesprint ergeben.

  - Projektstatus und aktuelle Revisionsnummer — Gruppe
  - Status der Aufgaben inklusive Revisionsnummer — Gruppe
  - Verständlichkeit und Nachvollziehbarkeit von der Bewertung der Sprintplanung — Gruppe

Implementierung

Die API-Projektmappe enthält zum Zeitpunkt der Abgabe (10. Juli 2018 23:59 Uhr) die zu bewertende Version der Software und Dokumentation. Für die Bewertung wird ausschließlich der Stand des master-Zweigs zum Zeitpunkt der Abgabe verwendet. Im Speziellen sind die folgenden Repositories Teil der Abgabe: Das Software-Repository und das Wiki-Repository.

- Quellcode

  Das Software-Repository soll die fertige Software als Quelltext enthalten. Alle User-Stories sollen in dieser Software umgesetzt werden (eine ausführbare Datei pro Projekt). Eine Ausnahme bilden Projekte, in denen die Software zur Kommunikation zwischen verschiedenen Geräten verwendet wird. In diesem Fall dürfen mehrere ausführbare Dateien Bestandteil der Projektmappe sein. Die Software soll kompilierbar sein. Der Quellcode soll von anderen Dateien (Medien, Dokumente und weitere Ressourcen) abgrenzbar sein und in einem dedizierten Ordner (REPO-URL/src) abgespeichert werden.

  - Oben genannte Projektstruktur eingehalten — Gruppe
  - Kompilierbarkeit des Projekts und aller etwaigen Unterprojekte — Gruppe

- Kommentare im Quellcode

  Der Quelltext der Software soll vollständig mittels Code-Kommentaren dokumentiert werden. Als vollständig dokumentiert gilt der Quelltext, wenn die folgenden Punkte dokumentiert sind:

  - Code-Blöcke, die eine eigene Logik enthalten
  - Jede Funktion mit ihrer Aufgabe, den Ein- und Ausgabeparametern und entsprechend ihres Umfangs der Vorgehensweise
  - Klassen mit ihrer Aufgabe und Besonderheiten bei der Initialisierung und Anbindung

- Dateien mit den beteiligten Autoren und den behandelten User-Stories

  Gruppe

- Jedes Gruppenmitglied soll 2 Funktionen, die Ein- und Ausgabeparameter enthalten, wie oben beschrieben, dokumentieren Individual

- Commits

  Jedes Gruppenmitglied soll in dem Software-Repository mindestens 10 Commits erstellt haben. Mit diesen Commits wird sichergestellt, dass jedes Gruppenmitglied ein Versionsverwaltungssystem verwendet hat. Die Commits sollen mit Nachrichten versehen werden, die die Änderung an der Software beschreiben (Einzeilige Zusammenfassung der Änderung und ggf. Angabe von: User-Story, Bug und/oder Feature). Commits sollen möglichst kleinteilig erfolgen und nicht zu viele Änderungen auf einmal enthalten (gut: Bug 121 in Funktion X behoben; schlecht: Implementierung von User-Story A; schlecht: Fortschritt am Projekt; schlecht: Datei xy hinzugefügt). Jedes Gruppenmitglied soll die 10 zu bewertenden Commits auf der Release-Wikiseite unter seinen Namen angeben.

  - Nachvollziehbarkeit der Änderung über Commit-Nachricht Individual
  - Sinnvolle Größe der Commits Individual

- Bug-Report

  Im Laufe der Softwareentwicklung treten unweigerlich Fehler auf. Diese können sich bereits zur Kompilierung oder auch während der Programmlaufzeit manifestieren. Ein Bug-Report adressiert solche Fehler und soll den Programmierer in die Lage versetzen, diese Fehler zu reproduzieren und so bei der Fehlerbehebung helfen. In der Projektmappe soll jedes Gruppenmitglied im Entwicklungszeitraum mindestens einen Bug-Report erstellen, das einen Fehler zur Programmlaufzeit adressiert. Fehler sind Abweichungen von dem vorgesehenen Programmverhalten oder ein Programmabsturz. Bestandteil eines Bug-Reports sind

  - die Revisionsnummer des Commits, das den unveränderten Quellcode der getesteten Software enthält. Individual
  - die Fehlerbeschreibung (Fehlermeldung beim Absturz, Beschreibung des abnormalen Programmverhaltens) Individual
  - die Schritte zum Reproduzieren des Fehlers angefangen bei dem Programmstart Individual

  Der Bug-Report kann als issue auf Github erstellt werden oder direkt auf der vorgesehenen Seite im Wiki. Sollte der Bug-Report als issue erstellt werden, muss dieser auf der Wikiseite verlinkt werden.

Anleitung

Mit der Anleitung soll die Inbetriebnahme der Software und - falls vorhanden - Hardware für einen Projektunbeteiligten ermöglicht werden. Jede User-Story des finalen Backlogs soll durch ein

eigenes Kapitel beschrieben werden. Es ist darauf zu achten, dass Schnittstellen (zum Beispiel der Anschluss der Hardware) und Systemvorraussetzungen in der Anleitung enthalten sind. Funktionalitäten und Konfigurationsmöglichkeiten der User-Stories sollten in der Anleitung gleichermaßen behandelt werden. Grafische Benutzerschnittstellen sollen anhand von Screenshots erläutert werden. Die Anleitung kann entweder als Wikiseite oder als pdf-Dokument erstellt werden. Das pdf-Dokument muss auf der entsprechenden Wiki-Seite verlinkt werden.

- Vollständigkeit
- Adäquate Darstellung und Bezug zu User-Stories und Zielkriterien — Individual
- Verständlichkeit — Individual
- Konsistenz der Darstellung — Gruppe
- Aktualität (Bezug zur finalen Version des Projekts) — Gruppe