Übung 12: Softwareprozesse und Konfigurationsmanagement

Teil 1. Softwareprozesse

Aufgabe A: Bestimmung von Entwicklungsprozessen

Schlagen Sie für jede dieser Entwicklungsaufträge ein Entwicklungsprozess vor und begründen Sie kurz ihre Auswahl (warum dieser und nicht die anderen Prozesse; Sie können auf die in Aufgabe A herausgearbeitet Punkte verweisen).

a. Versionssprung der Software für Fahrkartenautomaten
- Auftraggeber ist der RMV
- Ca. 10 -15 EntwicklerInnen, Entwickler der alten Version sind ansprechbar, aber nicht projektbeteiligt
- Für alle Fahrkartenautomaten für den RMV(Frankfurt und Umland)
- Neue Funktionalität: Automatensoftware erhält Aktualisierungen über ein Netzwerk (früher über ein lokales Terminal); Zusätzlich wird ein Aktualisierungsserver benötigt, welcher über ein definiertes Protokoll Daten mit der Automatensoftware austauscht; Statistiken über Fahrkartenverkäufe sollen so später ausgewertet werden können; Benutzerergonomische Richtlinien sollen eingehalten werden; neue Grafiken von einer Designfirma sollen die Erlernbarkeit verbessern;
- Auftragsabwicklung erfolgt gemischt: Anforderungsspezifikation wird durch Vertreter des RMV und der SW-Firma erstellt; Realisierung erfolgt durch SW-Firma
- Herausforderung: hohe Stabilität der Software, hohe Fehlertoleranz bzgl. Benutzereingaben

b. Materialbeschaffungssoftware für Kfz-Werkstatt
- Auftraggeber ist Kfz-Werkstatt, Auftragsabwicklung durch SW-Firma
- Ca. 5-10 EntwicklerInnen
- Materialanforderung erfolgte bisher auf dem Papierweg
- Funktionalität: Einfache Eingabemaske; Bestellung über standardisierte elektronische Schnittstelle direkt beim Lieferanten;
- Herausforderung: einfache Bedienung

c. Spieleentwicklung im „Adventure" –Bereich
- Auftraggeber: keiner, Ca. 20-30 EntwicklerInnen
- Nutzer: Alterszielgruppe ca. 15 – 35
- Herausforderung: Schnelle Entwicklung und schneller Markteintritt; Parallele Entwicklung in Teams(Grafik, Story, Logik); innovative visuelle Effekte; zusammenhängende Geschichte

d. Entwicklung eines Linux-Kernel-Moduls für das automatische Mounten von DVD-Laufwerken
- Auftraggeber: keiner (Toolentwicklung aus „Spass")
- 1 EntwicklerIn ggf. 2 EntwicklerInnen (FreundIn kommt dazu)
- Funktionalität: Beim Einlegen einer DVD soll das Laufwerk automatisch gemountet werden
- Nutzer: selber und ggf. andere
- Herausforderung: Erlernen der Kernelprogrammierung; Operating System bleibt weiterhin stabil und sicher

Teil 2. Konfigurationsmanagement

Aufgabe B: Bug Prozess

Modellieren Sie das grobe Vorgehen zum Melden und Lösen eines Bugs mittels Bugtracking Software (Ticket Systeme) als Sequenz- oder Kollaborationsdiagram (vgl. Vorlesung, Folie 45).

Das Modell soll beschreiben wie die Akteure „Person die den Fehler meldet“, „Tester“, „Projektmanager“ und „Entwickler“ mit dem System interagieren.

Aufgabe C: SVN/Git

Machen Sie sich mit Mercurial oder Git vertraut (Ihre Wahl). Erstellen Sie ein Repository und machen Sie es dem Tutor zugänglich

- Die einfachste Möglichkeit ist einen Account bei github.com (für Git) oder bitbucket.org anzulegen und dort ein Projekt zu erstellen.

Machen Sie jetzt folgende Änderungen an diesem Repository, möglichst jeweils als getrenntes Commit mit sinnvoller Commit-Nachricht:

a) Kopieren Sie eine Version des Lok-Quelltextes in dieses Repository.
b) Legen Sie im Hauptverzeichnis eine Datei README.txt oder README.md (in Markdown Format — http://github.github.com/github-flavored-markdown/) an, die mindestens Ihren Vornamen enthält.
c) Wenn Sie die Aufgabe zusammen mit Kommilitonen abgeben muss jeder seinen Namen selber zu der readme.txt Datei hinzufügen (in der Historie sollten mehrere Änderungen von verschiedenen Entwicklern erkennbar sein).
d) Führen Sie ein Refactoring aus um die Klasse Lokomotive in DieselLokomotive umzubenennen.

Prüfen Sie vor der Abgabe dass das Repository öffentlich zugänglich ist (oder zumindest dem Tutor zugänglich ist). Anschließend reicht es dem Tutor einen Link zum Repository zu schicken. Bei technischen Fragen bitte rechtzeitig an den Tutor wenden.

Ihre Lösung dieses Übungszettels geben Sie bitte wie gewohnt bis zum 10.02.2013 23:59 Uhr per Email an selecture@mathematik.uni-marburg.de ab.

Hinweise: Bei Mehrdeutigkeiten oder fehlenden Details nehmen Sie eine plausible Möglichkeit an und dokumentieren Sie ihre Annahmen.

Tutoriumaufgaben

Die folgenden Aufgaben werden wir gemeinsam im Tutorium besprechen. Bitte bereiten Sie sich selbständig angemessen darauf vor, um sinnvoll an einem Gespräch teilnehmen zu können. Antworten zu den Fragen müssen **nicht** schriftlich bearbeitet werden.

Teil 1. Softwareprozesse

Aufgabe D: Entwicklungsprozesse

Stellen Sie die in der Vorlesung besprochenen Softwareprozesse gegenüber. Arbeiten Sie jeweils Vor- und Nachteile der Prozesse heraus.

Zur Gegenüberstellung können Sie etwa eine Tabelle in der folgenden Form benutzen:

	Wasserfallmodell	V-Modell	…
Vorteile			
Nachteile			

Aufgabe E: Inkrementelles Vorgehen

Welche der Aussagen treffen für ein inkrementelles Vorgehen zu? Begründen Sie kurz Ihre Antworten.

a) Es ist frühzeitig eine benutzbare Version auslieferbar.
b) Der Auftraggeber ist nur ganz am Anfang involviert.
c) Man sollte es vor allem bei unklaren Anforderungen einsetzen.
d) Die Phasen sind an Bausteine/Subsysteme gekoppelt.
e) Man kann auf Modifikationen der Anforderungen gut reagieren.

Teil 2. Konfigurationsmanagement

Aufgabe F: Begriffe

1) Erläutern Sie am Beispiel von Microsoft Windows (oder einer anderen Software) die Begriffe Version, Revision und Variante.
2) Was versteht man in Versionsverwaltungssoftware unter Tags, Branches und Konflikten?

Aufgabe G: Aussagen

Welche der Aussagen treffen zu? Begründen Sie kurz Ihre Antworten.

a) Softwarevarianten werden in Versionsverwaltungssysteme durch Branches gelöst
b) Konfigurationsmanagement ist im Wasserfallprozess nicht nötig
c) Die Revisionsnummern in Versionsverwaltungssystemen sind unabhängig von den Versionsnummern der erstellten Software
d) Ein Versionsverwaltungssystem ersetzt regelmäßige Backups
e) Versionsverwaltungssysteme können alle Konflikte automatisch erkennen, aber nicht alle Konflikte automatisch beheben
f) Konfigurationsmanagement ist nur für Textdateien relevant und funktioniert für Binärdateien (z.b. Excel oder .exe) nicht
g) Mit Bugtracking-/Ticket-Systemen können auch Kundenwünsche für Änderungen und neue Features gesammelt und priorisiert werden.

h) Verteilte Versionsverwaltungssysteme brauchen keinen zentralen Server
i) In verteilten Versionsverwaltungssystemen ist jedes verteilte Repository immer auf dem gleichen Stand

Aufgabe H: Kleine vs. große Commits

Diskutieren Sie wie häufig lokale Änderungen in ein Versionsverwaltungssystem übertragen werden sollen (commit). Welche Vor- und Nachteile bieten häufige kleine Commits nach Änderungen an nur wenigen Zeilen Quelltext und seltenere, aber größere Commits?

Ändern sich Ihre Argumente wenn man Branches oder verteilte Versionsverwaltungssysteme mit einbezieht?