Managementplan

Gruppe SE2-04

Für das Projekt „Sharebox Ultimate“ entscheiden wir uns für ein agiles und iteratives Vorgehen. Es handelt sich bei diesem Projekt um keine hochsicherheitskritische Anwendung und bedarf daher keiner enorm exakten und vollständigen Planung und Fehleranalyse. Das iterative Vorgehen hingegen hilft uns dabei auf während der Entwicklung auftretende Probleme schneller und einfacher zu reagieren. Vor allem da unser Team noch relativ jung und dynamisch ist, werden Fehler auf Grund von Fehleinschätzungen passieren, welche aber durch das iterative und agile Vorgehen besser aufgefangen werden können.

Natürlich benötigt man auch bei einem agilen, iterativen Vorgehen eine gewisse Planung bzw. einen „Masterplan“. Deshalb werden wir in der ersten Projektphase zunächst ein Architekturmodell bzw. entsprechende Struktur, Ablauf- und Interaktionsmodelle erstellen. Die entsprechenden Modelle werden in gemeinsamen Meetings erarbeitet und dann von den Teammitgliedern separat verfeinert.

Auf Basis dieser Modelle können wir dann zu Beginn der zweiten Projektphase - der Implementierungsphase - entsprechende User-Stories bzw. Tasks erstellen, welche die einzelnen Teammitglieder dann nacheinander umsetzen. Eine User-Story bzw. ein Task ist erst dann abgeschlossen, wenn alle vorher definierten Akzeptanzkriterien erfüllt sind und die entsprechenden Unit-Tests erstellt wurden und funktionieren. Ob die Unit-Tests dabei nach Implementierung der Funktionalität oder nach dem Test-Driven-Development Prinzip erstellt werden ist dem jeweiligen Entwickler überlassen. Angestrebt wird jedoch, sofern später nicht anders vom Auftraggeber gewünscht, eine Testabdeckung des Codes von mehr als 80%. Dies ist ein aus Erfahrung ganz guter Wert, da 100% zu erreichen oftmals nicht sinnvoll ist bzw. der Nutzen den Aufwand nicht rechtfertigt oder es auch Teile des Codes gibt die nicht sinnvoll zu testen sind.
Wir werden hier stark mit GitHub als Versionierungssystem und Issue-Tracker-System arbeiten. Da unser Projekt nicht übermäßig komplex ist sollte das Issue-Tracker-System das GitHub zur Verwaltung von Stories, Tasks und Bugs bereitstellt vollkommen ausreichen.
Zur Versionierung verwenden wir an GitFlow angelehnte Konventionen. Es gibt einen Master-Branch auf dem nur als funktionierender Prototyp eingestufte Versionen aus dem develop-Branch eingecheckt werden. Vom develop-Branch aus erstellt sich jeder Entwickler für jede neue Story bzw. Task einen neuen Feature-Branch auf dem er dann die entsprechenden Änderungen vornimmt. Diese Feature-Branches werden dann in gemeinsamen Teammeetings zusammengeführt um Konflikte sofort mit allem Teammitgliederen lösen zu können und den Code gemeinsam zu reviewn. Somit ist jeder für den gesamten Code verantwortlich und sollte sich auch überall auskennen. Dies gibt uns mehr Flexibilität falls einzelne Teammitglieder länger mit bestimmten Aufgaben beschäftigt sind als geplant oder aufgrund von zB. Krankheit ausfallen. Zudem werden wir vor allem in der ersten Implementierungsphase einige Bereiche in Pair Programming umsetzen um eine höhere Codequalität zu erreichen und Wissenstransfer zwischen den Teammitgliedern zu fördern.
Auf den wöchtenlichen Teammeetings planen und verteilen wir zudem jeweils die Aufgaben für die nächste Woche. Wir legen Wert auf eine dezentrale Organisation und treffen die Entscheidungen als

Team.
Ein weiterer wichtiger Grundsatz unseres Teams, der allerdings fast selbstverständlich ist, ist „Program to an interface not to an implementation!“.

Neben GitHub werden wir zur Kommunikation viel auf Skype und natürlich E-Mails setzen. Der Datenaustausch mit dem Auftraggeber wird weiterhin über Dropbox erfolgen. Zur Verwaltung von Dokumenten verwenden wir, aufgrund der guten Cross-System Unterstützung, Google Docs.
Die Wahl der IDE überlassen wir den einzelnen Teammitgliedern, sie muss jedoch wie vom Auftraggeber gefordert das PMD-Plugin unterstützen. Voraussichtlich werden wir daher Eclipse, Netbeans und IntelliJ Idea verwenden.
Um unseren Code besser verständlich und weniger Fehleranfällig zu machen werden wir für die GUI das SWIxml Framework verwenden und bei den Unit-Test zum erstellen von Mock-Objekten das Mockito-Framework.

Bei der Umsetzungsreihenfolge der einzelnen Tasks und Stories orientieren wir uns grob an der im Analysedokument definierten Reihenfolge. Einige Abweichungen sind nötig, da viele Features aufeinander aufbauen bzw. technisch zusammenhängen und es keinen Sinn macht diese getrennt voneinander zu implementieren, bzw. es gar nicht möglich diese unäbhängig funktionsfähig zu machen. Ein gewisses technisches Grundgerüst ist zudem als Basis notwendig, was im Analysedokument nicht berücksichtigt wird. Die tatsächliche Reihenfolge der der Impelemntierung ergibt sich dann aus den erstellen Stories und Tasks, nachdem wir in der ersten Projektphase die zugrundeliegenden Modelle erstellt haben.

Aufgrund unseres oben beschriebenen Vorgehens können wir Ihnen daher leider keinen detaillierten Plan vorlegen, wann welches Teammitglied welche Features umsetzen wird, aber in folgendem Gantt-Diagram haben wir Ihnen grob aufgeführt wann wir welche Phase des Projekts durchlaufen werden.