WS 2016/17

Simon Kaiser, 354692
Philipp Hochmann, 356148
Felix Kiunke, 357322
Giacomo Klingen, 356778
Daniel Schleiz, 356092

Softwaretechnik
Aufgabenblatt 1

Aufgabe 1.1

Analyse

- Qualitätssicherung des Pflichtenheftes durchführen
  Das Pflichtenheft entsteht in der Analyse-Phase und dient als ein wichtiges Kommunikationsmittel zwischen dem Kunden und dem Projektleiter. Dessen Qualität sollte stets gesichert sein.

- Gesetzliche Rahmenbedingungen prüfen
  Es ist sinnvoll, schon in der Analyse-Phase die gesetzlichen Rahmenbedingungen zu prüfen, um festzustellen, ob die Wünsche des Kunden überhaupt rechtlich gesehen umsetzbar sind und im Zweifelsfall nach Alternativen zu suchen.

- Entwicklerteam zusammenstellen
  Während der Analyse sollte klar werden, in welche Richtung die Anwendung gehen soll und bei Bedarf kann man sich schon um Fachkräfte für absehbare Spezialgebiete kümmern. Zudem ist man bei einer so frühen Planung des Teams recht flexibel und kann im Notfall später noch Verstärkung suchen.

- Termine und Kosten des Projektes planen
  Es sollten jetzt schon die Termine und Kosten für das Projekt geplant werden, damit das Projekt nicht außer Kontrolle gerät.

- Vorhandene Altlasten des Kunden analysieren
  Anhand von Software Altlasten des Kunden entsteht ein genaueres Bild, was der Kunde tatsächlich haben oder eben nicht haben möchte, was wiederum in die Analyse einfließen sollte.

- Kunden eine Rechnung stellen
  Da man die Kosten bereits anhand der Anforderungen prognostizieren kann, ist ein erster Kostenanschlag in der Analyse-Phase ebenfalls

vernünftig. Zudem erspart man sich so spätere Streitigkeiten mit dem Kunden und kann einen Kompromiss finden. Die letztendliche Rechnung ist aber während der Integration zu stellen.

- Nach bereits vorhandenen, wiederverwendbaren Software-Bibliotheken suchen

Entwurf

- Konzept und Prototyp einer Benutzeroberfläche erstellen
  Erste Konzepte und der Prototyp einer Benutzerfläche sollten beim Entwurf entwickelt werden, da diese für den Kunden eine der wichtigsten Sachen darstellt und man bei Nicht-Gefallen darauf reagieren kann. Außerdem fallen bei der Erstellung des Konzeptes der Benutzeroberfläche noch eventuell weitere Funktionalitäten auf, welche in den Entwurf mit eingebunden werden sollten.

- Datenstrukturen festlegen
  Während man sich bei der Analyse bereits Gedanken über mögliche Datenstrukturen machen kann, sollte man sich beim Entwurf auf geeignete und effiziente Datenstrukturen festlegen.

- Schnittstellen von Programmmodulen definieren
  Im Entwurf sollten Programmmodule des Produkts ersichtlich werden. Mittels Definitionen von Schnittstellen lassen sich diese dann sinnvoll vernetzen.

- Strukturmodell des gesamten Softwaresystems entwerfen
  Der zentrale Aspekt des Entwurfes ist es, sich ein Strukturmodell des Projekts zu überlegen.

- Performance-Prognose des Softwaresystems erstellen
  Da beim Entwurf ein Gesamteindruck der Software entsteht, kann man eine erste Prognose der Performance des Systems erstellen und diese in der Implementierungsphase sinnvoll nutzen, um sich gewisse Ziele in Hinsicht auf Performance zu setzen.

Implementierung

- Code eines Programmmoduls debuggen
  Während der Implementierung bringt ein stumpfes Programmieren ohne Tests natürlich nichts; den Code eines Moduls zu debuggen ist deswegen elementar.

- Leistung der Entwickler bewerten und belohnen
  Schon im Implementierungsprozess sind bei Benutzung einer geeigneten Verwaltungssoftware die individuellen Leistungen der Entwickler ersichtlich. Eine Belohnung guter Leistungen kann dabei die Moral erhöhen und so die Qualität der Implementierung erhöhen.

- Nach bereits vorhandenen, wiederverwendbaren Software-Bibliotheken suchen
  Beim Programmieren fällt vielleicht auf, dass für gewisse Funktionalitäten bereits gute Bibliotheken existieren, wodurch sich Zeit sparen lässt und der Code meist weniger fehleranfällig ist.

- Programmcode kommentieren
  Während der Implementierung sollte der Code natürlich stets kommentiert werden, damit er später sauber gewartet oder erweitert werden kann.

Test-Integration

- Benutzer der Software schulen
  Bei der Integration sollte der Kunde beziehungsweise der letztendliche Nutzer natürlich zur Nutzung geschult werden, da sonst vielleicht einige Features falsch verstanden, falsch genutzt oder als Fehler interpretiert werden können.

- Zwei Subsysteme verbinden und testen
  Zwar sollten beim Implementieren zumindest rudimentäre Tests über die Kompatibilität von Subsystemen durchgeführt werden, ausführliche Tests finden aber in dieser Phase statt.

- Kunden eine Rechnung stellen
  Die letztendliche Rechnung kann nun gestellt werden, da am Ende dieser Phase einerseits das meiste der Software steht und andererseits der kommende Wartungsaufwand ersichtlich ist.

- Test-Eingabedaten für ein Programmmodul ermitteln
  Zum ausgiebigen Testen ist es notwendig, sinnvolle und realitätsnahe Testdaten für ein Programmmodul zu ermitteln.

Wartung

- Software an neue Umgebung anpassen
  Falls sich in der Umgebung des Kunden fundamentale Sachen ändern,

muss die Software gewartet und abgeändert werden, um den Anforderungen zu entsprechen.

- Dokumentation des Projektablaufes bewerten und archivieren
  Das Projekt sollte nun bis auf die Wartung abgeschlossen sein, weshalb nun eine Bewertung und Archivierung der Dokumentation erfolgen kann.

Eine solche starre Einteilung von Aktivitäten in Phasen wie im Wasserfallmodell ist problematisch, da dieses System sehr statisch ist und manche Aktivitäten wie zum Beispiel die Qualitätssicherung stets stattfinden sollten, um später größere Probleme zu vermeiden. Tritt beispielsweise in einer frühen Phase ein großer Fehler auf, welcher im Wasserfallmodell erst in einer viel späteren Phase durch eine Aktivität aufgedeckt wird, so wird unter Umständen geleistete Arbeit obsolet und man muss zu einer früheren Phase zurückspringen. Außerdem ist im Wasserfallmodell die Kommunikation mit dem Kunden in der Entwurfs- und der Implementierungsphase eher nicht vorgesehen, was aber prinzipiell eine schlechte Idee ist, da der Kunden in den Entwicklungsprozess involviert sein sollte, da dieser letzten Endes das Produkt akzeptieren soll.

Aufgabe 1.2

Ein zentraler Gesichtspunkt der SCRUM-Entwicklung ist das Sprint-basierte Arbeitsverhalten, wodurch man zeitlich begrenzte Phase für bestimmte Aufgaben hat und dann die Entwicklung für den nächsten Sprint erneut organisiert wird. Man arbeitet also vor allem nicht ergebnisorientiert, sondern zeitorientiert.
Zudem weist SCRUM einer inkrementelle Arbeitsweise auf, welche auf den Sprint-Backlogs beruht. Es werden für einen Sprint von der Gesamtheit der zu erledigenden Aufgaben ein gewisser Teil herausgenommen und zu Sprint-Backlogs organisiert, welche dann abgearbeitet werden können. Ist der Sprint vorbei, so werden die Backlogs wieder zurückgesteckt und es geschieht eine neue Organisation in in Backlogs aus der Gesamtheit der Aufgaben für den nächsten Sprint.
Zuletzt lassen sich durch diese Art der Softwareentwicklung die geleisteten Ergebnisse transparent darstellen, der Dialog zum Kunden ist also in sämtlichen Phasen sehr wichtig.

Die wohl wichtigste Eigenschaft des extreme Programming ist die inkrementelle, evolutionäre Entwicklung eines permanent lauffähigen Codes. In diesem sind dann die Ergebnisse stets sichtbar.
Daraus folgt eine zweite grundlegende, ein wenig kuriose Eigenschaft: Der Code selber und die ständig darauf ausgeführten Tests sind dann de facto

das Entwurfsdokument und die Dokumentation, welche im klassischen Sinne so nicht existieren. Insbesondere das disziplinierte Testen des Codes und die daraus entstehenden Dokumente sorgen für eine Qualitätssicherung.
Nichtsdestotrotz sollte beachtet werden, dass mit dieser Art der Entwicklung in gewissem Maße Chaos entstehen kann bei zu großen Teams, es ist also empfehlenswert eher kleinere Projekte damit zu realisieren.

Aufgabe 1.3

Funktionale Anforderungen

- Die App muss sich mit dem Stundenplan des vorhandenen Management Systems der Uni synchronisieren können
- Es sollte eine integrierte Accountverwaltung implementiert werden
- Als Identifikation soll ein verbinden mit social-network Profilen ermöglicht werden
- Eine Geolocationfunktion zum Finden von Übungspartnern
- Nachrichten verschicken und empfangen von anderen Personen
- Ein Live-Chatsystem mit Gruppenchats

Funktionale Anforderungen

- Der Geolocation Service soll nur im Umkreis von 5km nach Übungspartnern suchen
- Die Anwendung soll in C++ geschrieben werden
- Die Erstellung eines Accounts soll schnell und unkompliziert sein und nicht zu viele Informationen abfragen, mehr optionale Felder
- Die grafische Oberfläche soll der des Management Systems der Universität nachempfunden sein, damit der Umstieg nicht allzu schwer ist

- Die Anwendung soll gut skalierbar sein und mit vielen angemeldeten Usern zurechtkommen