

SOFTWARETECHNIK

Aufgabenblatt 1

Aufgabe 1.1

Analyse

- 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

- Qualitätssicherung des Pflichtenheftes durchführen
Das Pflichtenheft dient als ein wichtiges Kommunikationsmittel zwischen dem Projektleiter und dem Kunden und dessen Qualität sollte gesichert sein, damit der Kunde stets sieht, in welche Richtung das Projekt sich während des Entwurfs bewegt und im Notfall intervenieren kann.
- Konzept und Prototyp einer Benutzeroberfläche erstellen
Erste Konzepte und der Prototyp einer Benutzeroberfläche 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
Da im Entwurf die gewünschten Funktionalitäten klar werden, sollte man sich im gleichen Zuge auch Gedanken um effiziente Datenstrukturen machen und diese festlegen, wodurch in der Implementierungsphase weniger Chaos entsteht.
- Schnittstellen von Programmmodulen definieren
Im Entwurf sollten Programmmodule des Produkts ersichtlich werden. Mittels definitionen von Schnittstellen lassen sich diese dann sinnvoll vernetzen.
- Test-Eingabedaten für ein Programmmodul ermitteln
Bei der Definition von Programmmodulen hat man eigentlich stets deren Sinn und Funktionalität im Kopf, weshalb es sinnvoll ist, sich erste Test-Eingabedaten zu überlegen, mit welchen die Entwickler dann in der Implementierungsphase arbeiten können.
- Strukturmodell des gesamten Softwaresystems entwerfen
Der zentrale Aspekt des Entwurfes ist es, sich ein Strukturmodell des Projekts zu überlegen.
- Nach bereits vorhandenen, wiederverwendbaren Software-Bibliotheken suchen
Beim Entwurf sollte einem auffallen, dass zur Umsetzung mancher Funktionalitäten vorhandene, wiederverwendbare Software-Bibliotheken sinnvoll oder auch notwendig sind, um Zeit einzusparen, weshalb diese nun schon gesucht werden sollten.

- 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.
- Zwei Subsysteme verbinden und testen
Beim Implementieren sollten stets Subsysteme verbunden werden und getestet werden, damit nicht am Ende beim Zusammensetzen grundlegende Fehler auftreten, welche das Neuschreiben großer Teile erfordert. Die Tests erfolgen dabei aber im Rahmen des Möglichen, vollständige Tests finden in der entsprechenden Testphase ab. Dieser Punkt könnte also genauso gut im Testabschnitt auftauchen.
- 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.
- 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.
- Kunden eine Rechnung stellen
Die letztendliche Rechnung kann nun gestellt werden, da einerseits das meiste der Software steht und andererseits der kommende Wartungsaufwand ersichtlich ist.

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
Ein Großteil des Projekts ist in dieser Phase abgeschlossen, weshalb der Ablauf nun bewertet werden 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
- Der Stundenplan soll auch offline abrufbar sein mit dem Stand der letzten Synchronisation

- Der Geolocation Service darf nur ausgeführt werden, wenn der Nutzer explizit dazu zustimmt
- Die Erstellung eines Accounts soll schnell 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