1	2	3	Sum
3,5	2	4	9,5

Houman Biglari

Aufgabe 1.1

- Benutzer der Software schulen – **Keine Zuordnung** eigentl. Test-Integration oder Wartung
  - Benutzer können erst geschult werden nach dem die Software geschrieben wurde. Das Training von Benutzern ist nicht Teil des Entwicklungsprozesses.
- Qualitätssicherung des Pflichtenheftes durchführen – **Analyse** √
  - Das Pflichtenheft entsteht in der Analysephase
- Gesetzliche Rahmenbedingungen prüfen – **Analyse** √
  - Juristische Anforderungen in der Analyse überprüfen.
- Konzept und Prototyp einer Benutzeroberfläche erstellen – **Entwurf** √
  - Schnittstelle zwischen Mensch und Maschine wird entworfen und Anforderung an die Funktion des Programms modelliert.
- Entwicklerteam zusammenstellen – **Nach der Analyse** √
  - Es muss zuerst ein Team für die Programmstruktur zusammengestellt werden und später muss es für die Implementierung erweitert werden.
- Code eines Programmmoduls debuggen – **Implementierung** √
  - Einzelne Fehler müssen während der Implementierung behoben werden, nachdem sie beim Test entdeckt wurden.
- Zwei Subsysteme verbinden und testen – **Test-Integration** √
  - Systemintegration.
- Termine und Kosten des Projektes planen – **Analyse** √
  - Kostenplan und Terminplan werden während der Analyse erstellt.
- Datenstrukturen festlegen – **Analyse** √
  - Die Datenstruktur muss für die Entwicklung bekannt sein.
- Vorhandene Altlasten des Kunden analysieren – **Analyse** √
  - Auf die Bedürfnisse des Kunden umgehen.
- Schnittstellen von Programmmodulen definieren – **Analyse** eigentl. Entwurf
  - Schnittstellen werden während der Modellierung analysiert und definiert.
- Leistung der Entwickler bewerten und belohnen –**keine Zuordnung** √
  - Die Leistung ist nicht ab schätzbar, da die Wünsche und Bedürfnisse des Kunden sich ändern können. Die Testphase kann unvorhergesehene Schwierigkeiten bereiten.
- Software an neue Umgebung anpassen – **Wartung** √
  - Neu Wunsche des Kunden führen zu der Evolution des Systems.
- Kunden eine Rechnung stellen – **Nach der Testphase** eigentl. keine Zuordnung möglich
  - Der Kunde unterschreibt ein Pflichtenheft als Auftrag nach der Analysephase in dem sich die Kosten für das Programm befinden, die Rechnung kann erst nach der Auslieferung der Software gestellt werden (Vertragsabhängig).
- Test-Eingabedaten für ein Programmmodul ermitteln – **Entwurf** eigentl. Test-Integration
  - Beim Entwurf wird festgelegt was die einzelnen Funktionen als Wert bekommen und zurückliefern sollen.
- Strukturmodell des gesamten Softwaresystems entwerfen – **Analyse** (Systemmodellierung) eigentl. Entwurf
  - Teil der Systemmodellierung.
- Dokumentation des Projektablaufes bewerten und archivieren – **Keine Zuordnung, in der Regel nach Projektabschluss** √
  - Nach dem das Produkt ausgelifert wurde kann die Bewertung und Archivierung der Dateien erfolgen.
- Nach bereits vorhandenen, wiederverwendbaren Software-Bibliotheken suchen – **Implementierung** √

    - Spart Arbeit und führt zu geringerer Fehleranfälligkeit, hat aber keine Auswirkungen auf Modellierung in der Entwurfsphase.
- Performance-Prognose des Softwaresystems erstellen – **Entwurf** √
    - Sofern „Performance" produktkritisch, Teil des Entwurfs
- Programmcode kommentieren – **Implementierung** √
    - Zu besserem Verständnis vom Code sollte er bei der Entwicklung direkt kommentiert werden.

Warum ist eine starre Zuordnung im Wasserfallmodell problematisch?

Während des Entwicklungsprozesses muss häufig ein Schritt in eine „frühere" Phase gemacht werden, z.B. wenn sich die ursprünglichen Anforderungen als unvollständig erweisen oder während einer „späteren" Phase verändert wurden. Außerdem sind die Phasen nicht eindeutig ab grenzbar, da es z.B. für den Entwurf bereits erforderlich sein kann, eine modellhafte Implementierung zu machen - es kommt also unweigerlich zu Überschneidungen. Aus diesem Grund ist eine feste Zuordnung der Tätigkeiten schwer möglich. √

3,5 / 4 Punkte

Der Prozess der Entwicklung hat keinen linearen Weg.

Aufgabe 1.2

SCRUM-Entwicklung	eXtreme Programming
1. Sprint: zeitlich begrenzte Phase, um das Produkt zu entwickeln 2. Sprint Backlog: Liste der zu realisierenden User Stories √	1. evolutionäre Entwicklung in sehr kleinen Inkrements 2. Diszipliniertes und automatisiertes Testen als Qualitätssicherung √

2 / 2 Punkte

Aufgabe 1.3

Funktionale Anforderungen
F200: Ein Profil erstellen
F500: Chatroom erstellen
F510: Chatroom löschen
F520: User in Chatroom aufnehmen
F521: User zu Moderator eines Chatrooms ernennen
F700: Übungspartner in örtlicher Umgebung vorschlagen

√

Nicht-funktionale Anforderungen
Das System sollte mit 500.000 angemeldeten Studenten zurechtkommen
Die Anwendung soll in C++ geschrieben werden (Richtlinie des Universität-Rechenzentrums)
Die Anwendung soll auch auf AWS einsetzbar sein
Die Weboberfläche soll „Web 2.0" sein – durch Einsatz von nachladenden Elementen
Austausch mit Bestandsystem über JSON API
Schlankes, modernes Design

√

4 / 4 Punkte