HOCHSCHULE FURTWANGEN UNIVERSITY | HFU

Blatt 2

Software Engineering 2

Luis Staudt

Aufgabe 1

Name	Eigenschaften
Software für die Steuerung einer Kaffeemaschine	▪ **Kunde:** Nicht explizit genannt, vermutlich ein Hersteller von Kaffeemaschinen ▪ **Art der Applikation:** Eingebettete Software zur Steuerung der Hardware einer Kaffeemaschine ▪ **Zusätzliche Komponenten:** – Benötigt: Schnittstellen zu Sensoren und Aktuatoren der Kaffeemaschine – Nicht benötigt: Keine externe Datenbank, keine Netzwerkverbindung ▪ **Besonderheiten:** Fest definierte Anforderungen, stabile Spezifikation
Software “Pizza bestellen“	▪ **Kunde:** Pizza-Kette oder ähnlicher Gastronomiebetrieb mit direktem Kontakt zum Entwicklungsteam ▪ **Art der Applikation:** Web- oder Mobile-Anwendung für Kunden zur Bestellung von Pizza ▪ **Zusätzliche Komponenten:** – Benötigt: Datenbank für Bestellungen und Kundenprofile, Zahlungsschnittstelle, Sicherheitsmaßnahmen für Kundendaten – Nicht explizit erwähnt: Mail-Server für Bestellbestätigungen ▪ **Besonderheiten:** Kundenspezifische Entwicklung, Anforderungen können sich ändern

Tabelle 1: Zusammenfassung der Software-Projekte (Teil 1)

Name	Eigenschaften
Software "Elektronische Patientenakte"	▪ **Kunde:** Öffentlicher Auftraggeber (Bund), gewonnen durch Ausschreibung ▪ **Art der Applikation:** Datenbanksystem mit verteilten Zugriffsrechten für medizinische Einrichtungen ▪ **Zusätzliche Komponenten:** – Benötigt: Hochsichere Datenbank, Authentifizierungssystem, Zugriffskontrolle, Audit-Trail-System – Nicht explizit erwähnt: Schnittstellen zu bestehenden Krankenhausinformationssystemen ▪ **Besonderheiten:** Höchste Sicherheitsstandards, umfangreiche Dokumentationspflicht
Software zur Generierung von Klausuren	▪ **Kunde:** Hochschulen bzw. Bildungseinrichtungen ▪ **Art der Applikation:** Desktop-Software zur lokalen Installation ▪ **Zusätzliche Komponenten:** – Benötigt: Lokale Datenbank für Aufgabensammlung, Druckfunktionalität – Nicht benötigt: Keine zentrale Serverinfrastruktur, keine Netzwerkkomponenten ▪ **Besonderheiten:** Muss lokal installierbar sein, keine Onlineverbindung erforderlich

Tabelle 2: Zusammenfassung der Software-Projekte (Teil 2)

Aufgabe 2

Projekt	**Anwendbarkeit agiler Prinzipien**
Kaffeemaschinen-Steuerung	Nur bedingt anwendbar. Stabile Anforderungen und sequenzielle Funktionsabhängigkeiten passen eher zum Wasserfall-Modell. Hardware-Software-Interaktion erfordert detaillierte Vorabplanung. Fehlender direkter Kundenkontakt während der Entwicklung limitiert agile Kundenzusammenarbeit.
Pizza-Bestellsoftware	Sehr gut anwendbar. Direkter Kundenkontakt ermöglicht Anforderungsanpassungen. Kundenspezifische Natur unterstützt iterative Entwicklung mit frühem Feedback. Die drei Hauptkomponenten können inkrementell entwickelt werden. Nicht-funktionale Anforderungen betonen Individuen über Prozesse.
Elektronische Patientenakte	Teilweise anwendbar mit Anpassungsbedarf. Umfassende Dokumentationspflicht steht im Kontrast zum agilen Prinzip "Funktionierende Software über Dokumentation". Hohe Sicherheitsstandards erfordern gründliche Planung. Inkrementelle Funktionsentwicklung möglich. Komplexe Stakeholder-Landschaft erfordert formellere Abstimmungsprozesse.
Klausurgenerator	Moderat anwendbar. Klare Funktionalitäten eignen sich für iterative Entwicklung mit frühen Prototypen. Desktop-Installation vereinfacht Bereitstellung früher Versionen. Direkte Einbeziehung von Dozierenden möglich. Stabile Anforderungen und lokale Nutzung erschweren kontinuierliche Integration.

Tabelle 3: Anwendbarkeit agiler Prinzipien auf verschiedene Softwareprojekte

1. Software für die Steuerung einer Kaffeemaschine

Die Entwicklung der Kaffeemaschinen-Steuerungssoftware **eignet sich weniger** für agile Methoden:

- Technisches System mit klaren physikalischen Einschränkungen
- Dokumentation wichtig für Sicherheit, Regulierung und Wartung
- Stabile, vordefinierte Anforderungen
- Deterministische Prozessfolge mit klaren Abhängigkeiten
- Geringer Raum für Änderungen

2. Software „Pizza bestellen

Die Pizza-Bestellsoftware **eignet sich sehr gut** für agile Entwicklungsmethoden:

- Kundenorientierte Software erfordert tiefes Verständnis der Benutzerinteraktionen
- Schnelle Markteinführung wichtiger als umfassende Dokumentation
- Direkte Zusammenarbeit mit dem Auftraggeber möglich
- Anforderungen anpassbar während der Entwicklung
- Iterative Verbesserung der Benutzererfahrung und Sicherheit

3. Software „Elektronische Patientenakte

Die Patientenakten-Software **eignet sich teilweise** für agile Methoden:

- Enge Zusammenarbeit zwischen Fachexperten und Entwicklern nötig
- Umfassende Dokumentation für Regulierung und Patientensicherheit unerlässlich
- Öffentliche Ausschreibung mit weniger flexiblen Vertragsbedingungen
- Hohe Sicherheitsanforderungen erfordern stabile Architektur
- Hybrider Ansatz empfehlenswert

4. Software zur Generierung von Klausuren

Die Klausurgenerierungs-Software **eignet sich gut** für agile Entwicklungsmethoden:

- Enge Interaktion mit Dozierenden als Endnutzer
- Frühes Feedback wichtiger als umfassende Vorabdokumentation
- Kontinuierliche Abstimmung zu spezifischen Anforderungen nötig
- Anforderungen können sich während der Implementierung ändern
- Überschaubare Komplexität, Fokus auf Benutzerfreundlichkeit

Aufgabe 3

1. Grundidee/Grundprinzip

Scrum ist ein agiles Framework für komplexe Produktentwicklung, das auf empirischer Prozesskontrolle basiert. Es folgt drei Grundprinzipien:

- **Transparenz:** Alle wichtigen Aspekte des Prozesses müssen für alle Beteiligten sichtbar sein
- **Überprüfung:** Regelmäßige Überprüfung der Artefakte und des Fortschritts
- **Anpassung:** Schnelle Anpassung bei Abweichungen vom Ziel

Scrum organisiert die Arbeit in kurzen, zeitlich begrenzten Entwicklungszyklen (Sprints) und betont selbstorganisierte, cross-funktionale Teams.

2. Aktivitäten

- **Sprint:** Zeitlich begrenzte Entwicklungsphase (1-4 Wochen) mit festem Ziel
- **Sprint Planning:** Meeting zur Festlegung der Sprintziele und Aufgabenauswahl
- **Daily Scrum:** Tägliches 15-minütiges Statusmeeting des Teams
- **Sprint Review:** Präsentation des fertigen Inkrements am Ende des Sprints
- **Sprint Retrospective:** Reflexion über den abgeschlossenen Sprint mit Fokus auf Verbesserungspotential
- **Product Backlog Refinement:** Kontinuierliche Pflege und Priorisierung des Backlogs

3. Rollen und Verantwortlichkeiten

- **Product Owner:**
  - Verantwortlich für Produktvision und Maximierung des Wertes
  - Verwaltet das Product Backlog und priorisiert Anforderungen
  - Entscheidet über Produktfunktionalitäten
- **Scrum Master:**

  - Fördert und unterstützt Scrum-Praktiken
  - Beseitigt Hindernisse
  - Schützt das Team vor externen Störungen
  - Coaching des Teams und der Organisation
- **Entwicklungsteam:**
  - Selbstorganisiert und cross-funktional
  - Verantwortlich für die Umsetzung der Anforderungen
  - Gemeinsame Verantwortung für das Ergebnis

4. Produkte/Dokumente (Deliverables)

- **Product Backlog:** Priorisierte Liste aller gewünschten Produktfeatures
- **Sprint Backlog:** Auswahl von Product Backlog-Einträgen für den aktuellen Sprint
- **Inkrement:** Summe aller abgeschlossenen Product Backlog-Einträge, die "Done" sind
- **Definition of Done:** Gemeinsames Verständnis darüber, wann ein Inkrement als fertig gilt
- **Burndown Chart:** Visualisierung des verbleibenden Arbeitsaufwands

5. Methoden, Richtlinien, Standards und Werkzeuge

- **User Stories:** Format zur Beschreibung von Anforderungen aus Benutzersicht
- **Planning Poker:** Methode zur Aufwandsschätzung
- **Task Board:** Visualisierung des Arbeitsfortschritts (oft mit Kanban-Boards)
- **Velocity:** Metrik zur Messung der Teamleistung
- **Werkzeuge:** JIRA, Trello, Azure DevOps, physical boards, etc.
- **Timeboxing:** Strenge zeitliche Begrenzung aller Meetings und Aktivitäten
- **Continuous Integration/Continuous Delivery:** Technische Praktiken, die oft mit Scrum kombiniert werden

6. Vor- und Nachteile

Vorteile:

- Hohe Flexibilität und Anpassungsfähigkeit bei sich ändernden Anforderungen
- Frühe und regelmäßige Lieferung von funktionsfähiger Software
- Hohe Transparenz über Fortschritt und Hindernisse
- Starker Fokus auf Kundenzufriedenheit
- Verbesserung der Teamkommunikation und -zusammenarbeit
- Reduzierte Risiken durch regelmäßiges Feedback

Nachteile:

- Weniger geeignet für sehr große oder verteilte Teams ohne Anpassungen
- Erfordert erfahrene und engagierte Teammitglieder
- Kann bei mangelndem Commitment des Managements scheitern
- Herausfordernd bei festen Lieferterminen und festem Budget
- Dokumentation kann vernachlässigt werden
- Schwieriger bei sicherheitskritischen Systemen ohne zusätzliche Maßnahmen

Bewertungstabelle der Modelleigenschaften für Scrum

Eigenschaften des Modells		**Note**
Projektgröße und Komplexität	Klein	1
	Mittel	1
	Groß	3
Qualität von Anforderungen	Klar	2
	Vage	1
Änderungen an Anforderungen	Keine	4
	Moderat	1
	Häufig	1
Sicherheit	Sicherheitskritisch	3
	Hoch	2
	Mittel	1

Tabelle 4: Bewertung der Modelleigenschaften für Scrum (1 = sehr gut geeignet, 6 = ungeeignet)

Aufgabe 4

1. Grundidee/Grundprinzip

Das V-Modell XT ist ein Vorgehensmodell für die Planung und Durchführung von Systementwicklungsprojekten. Es erweitert das klassische V-Modell, wobei XT für "Extreme Tailoring" steht. Folgende Grundprinzipien charakterisieren das Modell:

- **V-förmiger Entwicklungsprozess:** Entwicklungs- und Testaktivitäten bilden die Form eines V
- **Validierung und Verifikation:** Jeder Entwicklungsphase steht eine entsprechende Testphase gegenüber
- **Produktzentrierung:** Fokus auf Ergebnisprodukten statt auf Aktivitäten
- **Anpassbarkeit:** Projekte können das Modell durch Tailoring an ihre Bedürfnisse anpassen
- **Qualitätssicherung:** Durch geregelte Überprüfung und Abnahme der Projektergebnisse
- **Einheitliche Struktur:** Standardisierte Vorlagen und Prozesse für bessere Vergleichbarkeit

2. Aktivitäten

- **Projektmanagement:** Projektplanung, -steuerung und -überwachung
- **Systemanforderungsanalyse:** Erhebung und Dokumentation von Anforderungen
- **Feinentwurf:** Detaillierte Spezifikation einzelner Systemkomponenten
- **Implementierung:** Umsetzung des Entwurfs in Code
- **Integration:** Zusammenführung der Komponenten
- **Systemtest:** Überprüfung des Gesamtsystems
- **Abnahmetest:** Validierung gegen ursprüngliche Anforderungen
- **Konfigurationsmanagement:** Verwaltung der Systemversionen und Änderungen

- **Qualitätssicherung:** Kontinuierliche Überwachung der Qualität
- **Problemmanagement:** Systematische Bearbeitung von Problemen
- **Änderungsmanagement:** Kontrolle und Durchführung von Änderungen

3. Rollen und Verantwortlichkeiten

- **Projektleiter:**
  - Verantwortlich für Projektplanung und -steuerung
  - Führung des Projektteams
- **Entwickler:**
  - Detailentwurf und Implementierung
  - Komponententests
  - Fehlerbehebung
- **Tester:**
  - Erstellung und Durchführung von Testplänen
  - Dokumentation von Testergebnissen
  - Überprüfung der Qualitätsstandards
- **QS-Beauftragter:**
  - Planung und Überwachung der Qualitätssicherungsmaßnahmen
  - Durchführung von Reviews und Audits
  - Überwachung der Prozesskonformität
- **Auftraggeber/Anwendervertreter:**
  - Abnahme der Lieferungen
  - Bereitstellung fachlicher Anforderungen
  - Entscheidung über Änderungsanträge

4. Produkte/Dokumente (Deliverables)

- **Projekthandbuch:** Zentrale Festlegungen zum Projektablauf
- **Projektplan:** Zeitliche und ressourcenbezogene Planung
- **Anforderungsspezifikation:** Funktionale und nicht-funktionale Anforderungen

- **Feinentwurfsdokumente:** Detaillierte Spezifikation der Komponenten
- **Quellcode und ausführbare Programme:** Implementierung
- **Testpläne und Testfälle:** Grundlage für die systematische Überprüfung
- **Testberichte:** Dokumentation der Testergebnisse
- **Benutzerdokumentation:** Anleitungen für Anwender
- **Abnahmeprotokolle:** Bestätigung der Lieferungen
- **Statusberichte:** Regelmäßige Projektfortschrittsberichte

5. Methoden, Richtlinien, Standards und Werkzeuge

- **Projektmanagement-Richtlinien:** Standardisierte Verfahren für Projektplanung und -steuerung
- **Review- und Inspektionsverfahren:** Systematische Überprüfung der Dokumente
- **Konfigurationsmanagement-Richtlinien:** Verfahren zur Versionsverwaltung
- **Tailoring-Konzept:** Anpassung des Modells an Projektbedürfnisse
- **Risikomanagementsystem:** Systematische Identifikation und Behandlung von Risiken
- **Änderungsmanagement-Prozess:** Kontrollierte Durchführung von Änderungen
- **Werkzeuge:** CASE-Tools, Projektmanagement-Software, Versionskontrollsysteme, Requirements-Management-Tools

6. Vor- und Nachteile

Vorteile:

- Umfassende Dokumentation und Nachvollziehbarkeit
- Frühzeitige Fehlererkennung durch Verifikation
- Klare Struktur und definierte Prozesse
- Hohe Qualitätssicherung durch systematische Überprüfungen
- Sehr gut geeignet für sicherheitskritische und regulierte Umgebungen

Nachteile:

- Hoher Dokumentationsaufwand
- Weniger flexibel bei sich ändernden Anforderungen
- Kann bürokratisch und starr wirken
- Zeitaufwändiger Tailoring-Prozess
- Verzögerte Rückmeldung zu Funktionalität durch späte Testphasen

Bewertungstabelle der Modelleigenschaften für V-Modell XT

Eigenschaften des Modells		**Note**
Projektgröße und Komplexität	Klein	4
	Mittel	2
	Groß	1
Qualität von Anforderungen	Klar	1
	Vage	5
Änderungen an Anforderungen	Keine	1
	Moderat	3
	Häufig	5
Sicherheit	Sicherheitskritisch	1
	Hoch	1
	Mittel	2

Tabelle 5: Bewertung der Modelleigenschaften für V-Modell XT (1 = sehr gut geeignet, 6 = ungeeignet)

Aufgabe 5

1. Software für die Steuerung einer Kaffeemaschine

Für die Entwicklung der Kaffeemaschinen-Steuerungssoftware empfehle ich das **V-Modell XT**.

Begründung:

1. Anforderungen sind klar definiert und stabil, optimal für V-Modell XT (Note 1)
2. Funktionales Flussdiagramm entspricht V-förmiger Struktur des Modells
3. Für Sicherheitsaspekte (Elektrizität, heißes Wasser) bietet V-Modell XT systematische Validierung
4. Besonders geeignet für sicherheitskritische Systeme (Note 1)

2. Software PPizza bestellen

Für die Entwicklung der Pizza-Bestellsoftware empfehle ich **Scrum**.

Begründung:

1. Kundenspezifische Software mit anpassbaren Anforderungen, ideal für Scrum
2. Exzelliert bei häufigen oder moderaten Änderungen (Note 1)
3. Direkter Kontakt mit Auftraggeber ermöglicht regelmäßiges Feedback
4. Besonders geeignet für Projekte mittlerer Größe (Note 1)

3. Software Ëlektronische Patientenakte

Für die Entwicklung der elektronischen Patientenakte empfehle ich eindeutig das **V-Modell XT**.

Begründung:

1. Muss höchsten Sicherheitsstandards entsprechen, V-Modell XT ideal für sicherheitskritische Systeme (Note 1)
2. Regierungsauftrag fordert umfassende Dokumentation über gesamten Lebenszyklus
3. V-Modell XT bietet diese für große Projekte (Note 1)

4. Software zur Generierung von Klausuren

Für die Entwicklung der Klausurgenerierungs-Software empfehle ich das **V-Modell XT**, wenn auch mit etwas Tailoring für Effizienz.

Begründung:

1. Stärke des V-Modell XT bei stabilen Anforderungen (Note 1)
2. Zuverlässigkeit und Korrektheit haben oberste Priorität, sichergestellt durch systematische Testphasen
3. Lokale Installation erfordert gründliche Systemarchitektur und Kompatibilitätstests