FFHS
Fernfachhochschule Schweiz
Zürich | Basel | Bern | Brig
Mitglied der SUPSI

Arbeitsmappe für das Modul „Projektarbeit“ (PA_5)

(Vorgaben zur Projektarbeit)

Grundsätzliches zum Modul Projektarbeit (PA_5)

Mit der Projektarbeit liegt ein Modul vor, bei dem die Studierenden die Gelegenheit erhalten, die bis dato erworbenen Kenntnisse und Fähigkeiten interdisziplinär, d.h. über Fächergrenzen hinweg zu vertiefen. Der Ansatz: Es wird ein zusammenhängendes, kleineres Projekt selbständig und mit hoher Eigenverantwortung während der Dauer eines Semesters realisiert.

Ausgehend von einer selbst gewählten Projektidee, analysieren die Studierenden die Anforderungen, die der Auftrag stellt, erarbeiten Lösungskonzepte, wählen die angemessenen Entwurfs-Werkzeuge und erarbeiten mit Einsatz von Programmier- und Test-Techniken eine funktionsfähige Software-Lösung. Die Arbeiten sind mit Hilfe bekannter Projektmanagement-Methoden/-modelle zu planen, zu steuern und zu kontrollieren.

Damit erbringen die Studierenden den Nachweis, dass sie in der Lage sind, Lösungswege und Lösungsinhalte eines Projekts funktional richtig und nachvollziehbar umzusetzen sowie verständlich, d.h. in korrekter sprachlicher Form und Darstellungstechnik zu dokumentieren und überzeugend zu präsentieren.

Die Projektteams werden Ende des 4. Semesters gebildet und am Kick-off-Meeting durch weitere Studenten definitiv gebildet und ergänzt. Pro Team einigt man sich bis spätestens am Ende des Kick-off-Meetings auf ein Projekt, welches während der PA_5 bearbeitet wird. Um die Themenwahl ideenmässig zu unterstützen, finden Sie nachstehend eine kleine Themenauswahl, die Sie jedoch nicht nutzen müssen.

Mögliche Projektthemen / Themenbeispiele

- Vereinsbuchhaltung
- Debitorenbewirtschaftung für ein KMU
- Internetprojekt: Bewerbung von Interessenten für Produkt X
- Einfaches Ressourcentool zur Erkennung der Kapazitätssituation
- Online Reservationssystem für einen Sportbetrieb
- Eigenständige Kostenüberwachung für Projekte
- Auswertungs-Software für Sport-Wettkämpfe
- Sportturnierplanung (Mannschaften, Spielfelder, Schiedsrichter, Spielmodi)
- Erfahrungsdatenbank für Projektleiter
- Wissensablage mit hierarchischer Kategorisierung (Schweiz, Europa o. Java, OOP)
- Marktanalyse / Marktinformation für einen spezifischen Markt
- Online-Umfrage über die Mitarbeiter-Zufriedenheit
- Zeiterfassung und -Auswertung für Aufträge mit Arbeitszeiten und Material

- Webbasiertes Anmeldemodul für Kurse und Seminare
- Terminkalender für einfache und wiederholte Termine
- Auswertung und Darstellung von Datenbeständen (z.B. Wetterdaten)
- Verwalten von Sammlungen (z.B. Übungen) mit Katalogisieren und Suchen
- Simulationen nicht komplexer Systeme
- Wertschriften-Portfolio mit Zu- und -Abgängen, Erträgen, Spesen etc. verwalten
- Benachrichtigungen (SMS, Mail) nach Ereignissen (fallende Aktie, verspäteter Zug)
- Online Büchertauschbörse mit virtueller Währung
- Games

Massgebende Dokumente / Unterlagen für die Projektarbeit

- Der Modulplan: Dieses Dokument skizziert „kurz und bündig“ alles Notwendige, alles Wissenswerte zum Modul Projektarbeit (PA_5)

- Die Arbeitsmappe: In der Arbeitsmappe finden Sie sämtliche Dokumente bzw. Resultate aufgelistet, die je Fachbereich (PM-G, SWE, OOP1, OOP2, PVAJ und DBS) erwartet, bewertet und benotet werden. Die Vorgaben der Arbeitsmappe gelten als „Mass“ für die Bewertung der Projektarbeit.

- Im Moodle-Kurs finden Sie weitere bzw. ergänzende Dokumente / Vorlagen, die vor allem anlässlich des Kick-off-Meetings festgelegt wurden. Der Informationsaustausch erfolgt ausschliesslich via Moodle.

Übersicht der Vorgaben zur Projektarbeit

1. Allgemeine Vorgaben

Sämtliche Dokumente sind durch Studiengang, Jahrgang, Ort, Teammitglieder und Projekttitel zu identifizieren.

Die Dokumentation, welche am Schluss der Arbeit zur Bewertung eingereicht wird, muss von allen Teammitgliedern akzeptiert sein. Die diesbezügliche Bestätigung ist mit separatem Formular und durch Unterschrift aller Teammitglieder zu „beurkunden".

Die nachstehenden Vorgaben / Anforderungen der Module Projektmanagement und Software - Engineering sowie die Coderichtlinien sind schriftlich zu dokumentieren, d.h. in der Dokumentation festzuhalten!

2. Vorgaben Projektmanagement

2.1 Projektauftrag mit folgenden Inhalten:

- Formale Angaben
- Ausgangslage (Beschreibung der IST-Situation)
- Projektziele und Inhalte. Was soll erreicht werden? Was NICHT? etc.
- Rahmenbedingungen / Restriktionen / Technische Ressourcen
- Grundlagen; welche Vorarbeiten liegen schon vor?
- Messbare Lieferobjekte:
  - Dokumente (gebunden) gemäss Arbeitsmappe
  - Präsentation, Funktionalitätsnachweis
- Eckwerte:
  - Termine (Meilensteine)
  - Allenfalls Kostenaspekte
  - „Personalplanung / Rollenverteilung"
- Risiken
- Nutzenbetrachtungen (fakultativ)
- Varia

2.2 Begründeter Lösungsansatz

Gewählter Lösungsansatz (gewählte Variante) ist „kurz und bündig" schriftlich zu begründen. Der Lösungsansatz muss aus der Problemsituation bzw. Auftragssituation im Sinne des Problemlösungszyklus (Zielsuche, Lösungssuche, Auswahl) approximativ nachvollziehbar sein. Es muss erkennbar sein, dass die Lösungsvariante(n) aus der Situationsanalyse (IST-Situation) und der Zielformulierung (Projektziele) abgeleitet ist/sind. Zur Darstellungsform werden keine Vorgaben gemacht.

2.3 Projektstrukturplan (PSP)

2.3.1 Aufgabengliederung

Mittels des Projektstrukturplanes (PSP) sind die wichtigsten Aufgaben (was ist zu tun?) des Projektes in plan- und kontrollierbare Teilaufgaben und Arbeitspakete zu gliedern. Die Gliederung kann objekt- und / oder funktionsorientiert sein.

Es werden mindestens diejenigen Teilaufgaben / Arbeitspakete erwartet, die notwendig sind, um die Projektziele zu erreichen.

2.3.2 Darstellungsform

Der Projektstrukturplan kann grafisch, mittels Mind Mapping bzw. als Organigramm oder in tabellarischer Form (einfache, klassifizierte Auflistung der Aufgaben) dargestellt werden.

2.4 Phasenplan / Vorgehensmodell / -prozess

Das Vorgehens- oder Phasenmodell / Phasenprozess gliedert die Projektabwicklung in eine sequentielle Reihenfolge von Phasen und Entscheidungen (Meilensteine), die „iterativ zu durchlaufen“ sind. Das phasenweise Vorgehen ist dahingehend zu dokumentieren, dass die notwendigen Aufgaben / Aktivitäten für die Projektrealisierung ablauflogisch erkennbar sind.

Pro definierte Phase sind also Meilensteine zu setzen und stichwortartig die Teilaufgaben und Arbeitsschritte je Phase zu beschrieben. Da Modellart, Anzahl Projektphasen, Form etc. von Art und Umfang des Projektes abhängig ist, werden diesbezüglich keine weiteren Vorgaben gemacht. Es genügt deshalb, die Phasen und die Arbeitsschritte / Arbeitspakete im Terminplan (siehe Punkt 2.5) umfassend abzubilden bzw. „sichtbar“ zu machen.

2.5 Terminplan

Die Terminplanung ist in Form eines Balkenplanes / Balkendiagramms (Gantt-Diagramm) darzustellen, der die Arbeitspakete, ihre ablauflogischen Zusammenhänge, die Bearbeitungsdauer und weitere Details je Aktivität aufzuzeigen vermag. Folgende Kriterien müssen aus dem Balkendiagramm mindestens „lesbar“ sein:

- Alle Aktivitäten (Teilaufgaben / Arbeitspakete) phasenbezogen dokumentiert (siehe auch Punkt 2.4)
- Start und Ende (Dauer) je Aktivität und Phase
- Meilensteine / Fixtermine
- Verknüpfungen zwischen den Aktivitäten
- Kritischer Pfad
- Personenzuteilung und Funktion (Ressourcen- / Rollenplanung)

Anmerkung:
Selbstverständlich kann der Terminplan auch mittels Netzplan dargestellt werden.

3. Vorgaben Software – Engineering

3.1 Analysedokumente

3.1.1 Systemidee

Die wichtigsten Eigenschaften, Leistungsmerkmale, Rahmenbedingungen, Voraussetzungen sowie auch eine grobe Abgrenzung des Systems sind in 5 bis 20 Sätzen zu beschreiben.

3.1.2 Anforderungen dokumentieren

Die Dokumentation von Anforderungen spielt eine zentrale Rolle im Requirements Engineering. Bei einer grossen Menge a Anforderungen ist es enorm wichtig, diese übersichtlich zu strukturieren. Ergänzend bildet ein Use-Case Diagramm eine Übersicht der Anwendungsfälle. Siehe [Pohl15] Kapitel 4, 5 und 6.

3.1.3 Fachklassenmodell (Domänenmodell)

In einem UML-Klassendiagramm sind alle fachlichen Klassen (z.B. Kunde, Buchung, Rechnung, Auto, Abteilung etc.), deren Beziehungen und wichtigsten Attribute zu dokumentieren.

3.1.4 Schnittstellenbeschreibung

Die Schnittstellen zu Umsystemen sind zu dokumentieren. Bei Zugriffen auf Umsysteme sind die genauen Aufrufe und die Reaktionen auf mögliche Fehler zu beschreiben. Pro Anwendungsfall sind die ein- und ausgehenden Daten, Objekte und Ereignisse festzuhalten.

Siehe [Ludewig13] Kapitel 17.2.3 und [Starke15] Kapitel 6.2.3 „Schnittstellen“

3.2 Softwareentwurfsdokumente

3.2.1 Sichten der Softwarearchitektur

Die Architektur des Systems ist zu beschreiben, Subsysteme und Module zu bestimmen sowie die Verteilung der Subsysteme und Module bezüglich ihrer Installation auf evtl. unterschiedlicher Hardware festzulegen. Die benötigten Fremdsysteme sowie die Systemgrenzen sind zu veranschaulichen.

Siehe [Starke15] Kapitel 5.4ff „Sichten“

3.2.2 Klassenmodelle

Für alle Komponenten (Module, Subsysteme) sind die Klassenmodelle als UML-Klassendiagramme zu entwerfen. Die Komponenten-Schnittstellen, die Methoden und Attribute der Klassen sowie die Vererbungsbeziehungen, Assoziationen etc. sind darin zu beschreiben.

Das Fachklassenmodell der Analyse bietet eine gute Grundlage für die Klassenmodelle. Doch die Klassenmodelle können davon abweichen und beinhalten weitere zur Implementierung notwendige (technische) Klassen wie z.B. Controller etc.

Siehe [Seidl12] Kapitel 4 „Klassendiagramm“

3.2.3 Datenmodell

Soll eine nicht objektorientierte Technologie zur persistenten Datenhaltung eingesetzt werden (z.B. eine relationale Datenbank), so ist zusätzlich das Datenmodell der Datenhaltung zu entwerfen (bei der Haltung der Daten in einer relationalen Datenbank ist ein Entity-Relationship-Diagramm zu verwenden).

Bei einer objektbasierten Datenhaltung heben Sie die Persistenzklassen im Klassenmodell hervor und beschreiben die Attribute detailliert.

3.2.4 Dynamische Modelle

Mindestens ein Ablauf und alle komplexeren Abläufe sind für den Entwurf mit einem dynamischen UML-Diagramm zu dokumentieren.

Siehe [Seidl12] Kapitel 5-7 „Zustands-“, „Sequenz-“ und „Aktivitätsdiagramm“

3.2.5 Testkonzept

Das Vorgehen zur Einhaltung und Erreichung der geforderten Qualität des zu implementierenden Systems ist festzulegen. Zu den Anwendungsfällen werden jeweils die nötigen Testfälle beschreiben und die erwarteten Resultate definiert.

Siehe [Ludewig13] Kapitel 19 „Programmtest“

3.2.6 Spezifikation der Bedienoberflächen

Die Gestaltung der Bedienoberflächen zu spezifizieren und liegt als Skizze mit der erforderlichen Beschreibung vor. Bedienoberflächen sind mit Bildschirmausdrucken von Prototypen darzustellen. Die Abläufe innerhalb der Bildschirmmasken und die Aktionen, die durch Betätigung von Oberflächenelementen ausgelöst werden, sind zu veranschaulichen.

3.3 Inbetriebnahme

3.3.1 Lieferobjekte und Installationsanleitung

Die Zusammensetzung des gelieferten Resultats der Projektarbeit ist zu dokumentieren. Die Installation dieser Lieferobjekte sowie die Konfiguration von Drittsoftware sind in einer Installationsanleitung festzuhalten.

3.3.2 Benutzerhandbuch

In einer kurzen Übersicht werden die Funktionen aus Sicht des Benutzers beschrieben. Dabei werden die Anwendungsfälle direkt mit Bildern aus dem laufenden Programm dargestellt und genauer erläutert.

4. Vorgaben Objektorientierte Programmierung

4.1 Programmiersprachen

4.1.1 Java

Als Programmiersprache ist hauptsächlich Java zu verwenden. Beachten Sie die Regeln und nutzen Sie die Sprachmittel der objektorientierten Programmierung

4.1.2 Weitere Sprachen

Neben Java dürfen Sie selbstverständlich auch andere Sprachen und Technologien wie SQL, (X)HTML usw. verwenden.

4.2 Software – Implementationsdokumente

4.2.1 Coderichtlinien

Dokumentieren Sie die in Ihrer Gruppe verwendeten Coderichtlinien. Begründen Sie die Wahl der Richtlinien kurz.

4.2.2 Quellcode

Jeder selbst geschriebene Quellcode ist in unkompilierter und in einer mit ASCII-Texteditoren lesbaren Form auf CD oder USB-Stick mitzuliefern.
Quellcode in einem allfälligen Image erfüllt diese Anforderung nicht.
Die Ablagestruktur des Quellcodes ist in Ihrer schriftlichen Arbeit zu dokumentieren.

Der Quellcode muss fehlerfrei ausführbar sowie objektorientiert und wartbar (als Basis für die Erweiter- und Wiederverwendbarkeit) sein.

Folgende Kriterien werden zur Bewertung Programmcodes herangezogen:
Korrektheit:

- Implementation stimmt mit dem SW-Entwurf überein resp. Abweichungen sind dokumentiert
- Fehlerfrei ausführbar

Wartbarkeit:

- Coderichtlinien sind eingehalten
- Strukturiert in Packages
- Klassen und mind. alle öffentlichen Methoden sind kommentiert (siehe 4.2.3)
- Klassen-, Methode-, Variablennamen sind verständlich
- JUnit-Test vorhanden, relevante Codeteile mit den Tests abgedeckt (siehe 4.2.4)
- Relationale DB: Sinnvoll normalisiert

Objektorientiert:

- Vernünftige Grössen der Klassen und Methoden (hohe Kohäsion, Trennung der Zuständigkeiten, eine Klasse resp. Methode hat genau eine Aufgabe/einen Zweck)
- Kapselung (keine öffentlichen Attribute)
- Lose Kopplung, sinnvolle Verwendung von Interfaces
- Vererbung eingesetzt, massvoller Umgang damit
- Statische Methoden möglichst vermieden

Robustheit:
- Validierung von Inputparametern (analog JavaDoc - siehe auch 4.2.3)
- Behandlung der Ausnahmefälle (Exceptions)

Effizienz:
- eingesetzte Algorithmen sind effizient und ohne Seiteneffekte

4.2.3 Dokumentation des Codes

Dokumentieren Sie in Ihrem Code alle Java-Klassen und öffentlichen Methoden mit Javadoc. Achten Sie darauf, dem Leser Ihrer Methoden-Dokumentation Informationen über erlaubte und nicht erlaubte Inputparameter, die Varianten der Ausnahmebehand-lung und die zu erwartenden Rückgabewerte anzugeben. Quellcode in anderen Sprachen ist ebenfalls so zu dokumentieren, dass Dritte den Code verwenden und warten könnten.

4.2.4 Automatisierte Unit-Tests

Stellen Sie mit JUnit-Tests die korrekte Funktion und die Dokumentation der Funktionsweise Ihrer Java-Klassen sicher.

4.3 Testprotokolle

Erstellen Sie Protokolle des Nachweises der Funktionalität und dokumentieren Sie die Verifikation des in der Entwurfsphase festgelegten Testkonzeptes.

5. Vorgaben Statusberichte

5.1 Projektstatus (Status-Meeting)

Während der Projektrealisierung muss der jeweilige Projektleiter dafür besorgt sein, dass alle im Projekt involvierten Personen über den Stand und Fortschritt des Projektes informiert sind / informiert werden. Dies erfolgt u.a. an Sitzungen, mittels Präsentationen, Berichte etc. An den beiden vorterminierten Statusmeetings (siehe Modulplan und Einsatzplanung) ist je Termin ein schriftlicher Projektstatus im Umfang von 1-3 Seiten abzugeben; beide Originalberichte sind in der Abschlussdokumentation aufzunehmen. Die Berichte sind erst am Status-Meeting (nicht früher!!) den Dozenten abzugeben.

5.2 Inhalte Statusberichte

Folgende Beurteilungsaspekte müssen im „Projektstatus-Bericht“ aufgenommen werden:

- Kurzbeschreibung des Gesamtstatus inkl. Bewertung (kritisch, teilweise kritisch, plan-mässig) und allfällige Massnahmen
- Status Termine inkl. Bewertung / allfällige Massnahmen
- Status Lieferobjekte inkl. Bewertung / allfällige Massnahmen
- Status Qualität inkl. Bewertung / allfällige Massnahmen
- Andere Probleme / Risiken und allfällige Massnahmen
- Nächste Schritte / Änderungsanträge
- Aktualisierte Rollenorganisation
- Nur beim 1. Statusmeeting: Nennung des gewählten Vorgehensmodells im Rahmen der Softwareentwicklung
- Varia

6. Vorgaben Abschlussbericht

6.1 Evaluierungsbericht

Mit der Projektdokumentation und der Schlusspräsentation zu Ende des Semesters sind die wesentlichen Planungs- und Realisierungsschritte im Sinne eines Projekt-Abschlussberichtes erfüllt. Um jedoch die Idee des Wissensmanagements (Nutzung von Erfahrungswerten) auch im Projektmanagement einzubringen, ist die Abschlussdokumentation noch mit einem Evaluierungsbericht zu ergänzen.

6.2 Inhalte Evaluierungsbericht (Projektverlauf)

- Wie ist das Projekt verlaufen?
- Wurden die Projekt- und Planungsziele erreicht?
- Gab es Schwierigkeiten? Wenn ja, welcher Art waren sie?
- Was lief gut, was lief weniger gut?
- Varia

7. Referenzen

[Ludewig13] Ludewig, Jochen und Lichter, Horst: „Software Engineering“, 3. Auflage, 2013

[Pohl15] Pohl, Klaus und Rupp, Chris: „Basiswissen Requirements Engineering“, 4. Auflage, 2015

[Starke15] Starke, Gernot: „Effektive Software-Architekturen“, 7. Auflage, 2015

[Seidl12] Seidl, Martina und Brandsteidl, Marion et al.: „UML@Classroom“, 1. Auflage, 2012