Handreichung zum SWT-Praktikum 2016/17

N. Arndt, P. Frischmuth, H-G. Gräbe, M. Martin

Version 1.0.1, 11. Januar 2017

Inhaltsverzeichnis

1 Vorbemerkungen und Vorgehensrahmen

Allgemeines

Im Rahmen des Softwaretechnik-Praktikums hat Ihr Team den Auftrag, ein umfangreicheres Software-Projekt von der Anforderungserhebung bis zu einem lauffähigen Prototypen in einem arbeitsteiligen, werkzeuggestützten Prozess selbstständig umzusetzen.

Für viele Teilnehmer wird es das erste Mal sein, dass sie vor einer solchen den Berufsalltag eines Informatikers prägenden Herausforderung stehen. In der Vorlesung *Softwaretechnik* haben Sie die grundlegenden Aspekte der ingenieurmäßigen Erstellung von Software kennengelernt, die es nun praktisch anzuwenden gilt.

Das Praktikumskonzept orientiert sich an einer agilen Vorgehensmethodik, deren Charakteristika in der Vorlesung an verschiedenen Vorgehensmodellen genauer erläutert wurden. Agile Vorgehensmodelle zeichnen sich durch eine enge Verzahnung und *inkrementelle Weiterentwicklung* von Anforderungsanalyse einerseits sowie Design und Implementierung andererseits aus, wobei die *Anforderungen* sowohl die inhaltliche Seite des Projekts (Produktdimension) als auch die Organisation und Qualitätssicherung (Prozessdimension) berücksichtigen müssen.

Weitere Informationen zum Praktikum sowie Erfahrungsberichte und Übersichten zu Praktika vergangener Jahre finden Sie im OO-Portal. Dort wird insbesondere die Bedeutung eines arbeitsteiligen Vorgehens, der Kommunikation untereinander sowie des Einsatzes adäquater technischer Hilfsmittel betont, um den Aufwand für Ihr Projekt in einem überschaubaren Rahmen zu halten.

Etappen des Praktikums

Agile Vorgehensmodelle bestehen aus

- einer *initialen Phase*, in der sich das Team einen generellen Überblick über das Zielsystem verschafft und die Grundlagen für die weitere imkrementelle Entwicklung legt,
- mehreren *Zyklen*, in denen die einzelnen Inkremente des Systems entwickelt werden,
- sowie der *Projektabschlussphase*, in der auch benötigte Dokumentationen und Erfahrungsberichte in der geforderten Qualität fertiggestellt bzw. final überarbeitet werden.

Die Etappen des Praktikums orientieren sich an diesen Phasen. Nach den Abschnitten

1. Team formieren und Projektressourcen einrichten sowie
2. genauere Analyse der Aufgabenstellung und grundlegende Vorstellungen zum inhaltlichen Vorgehen sowie zur Qualitätssicherung

der initialen Phase soll das Projekt zügig in mehreren Iterationen von einem ersten „Proof of Concept“ (Vorprojekt) zum finalen Release entwickelt werden, wobei auch die begleitenden konzeptionellen Dokumente (Arbeitsplan und QS-Plan) sowie die Entwurfsdokumentation angemessen fortgeschrieben werden.

Für jede Iteration sind umzusetzende *Issues* festzulegen, die einem *Meilenstein* zugeordnet werden. Jede Iteration wird mit einem *Releasebündel* abgeschlossen, das die bisherigen Ent-

wicklungen integriert und so den aktuelle Stand der Umsetzung Ihres Projekts dokumentiert. Zum *Projektabschluss* ist ein finales Releasebündel zu erstellen und abzugeben.

Als Zielstellungen für die einzelnen Etappen der initialen Phase ergeben sich folgende Punkte:

1. Stärken und Schwächen sind analysiert, Webseiten sind online, Projektressourcen sind eingerichtet und alle Mitglieder des Teams haben den Zugang zum Team-Repo nachgewiesen.

2a. Im *Recherchebericht* ist das Verständnis der Aufgabenstellung detailliert und mit dem Betreuer abgestimmt.

2b. Im *Arbeitsplan* und im *QS-Plan* sind grundlegende Vorstellungen zum inhaltlichen Vorgehen sowie zur Qualitätssicherung fixiert.

Im Arbeitsplan ist eine *Projektvision* zu formulieren sowie *Arbeitspakete* zu spezifizieren und deren Aufwand abzuschätzen als Grundgerüst der Umsetzungsplanung. Im weiteren Projektverlauf werden die Arbeitspakete im *Implementierungsplan* detailliert und in *Issues und Meilensteine* (Issue Tracker) heruntergebrochen.

Der Arbeitsplan ist Gegenstand des *ersten Reviews*, in dem das Team die bisher entwickelten Vorstellungen zur Projektumsetzung präsentiert und zur Diskussion stellt. Bitte beachten Sie die Ausführungen zur Durchführung von Reviews im Abschnitt 2.5.

Parallel dazu kann schon an der Umsetzung gearbeitet werden. Das erste Arbeitspaket (Vorprojekt) soll als „Proof of Concept“ bereits zeigen, dass Sie auf dem richtigen Weg sind.

Der Arbeitsstand nach dem zweiten Releasebündel ist Gegenstand des *zweiten Reviews*, in dem schwerpunktmäßig die Modellierung auf der Basis der aktuellen *Entwurfsbeschreibung* (siehe Abschnitt 3.5) thematisiert wird.

2 Vorgaben zu den einzelnen Phasen des Praktikums

2.1 Team bilden und Projektressourcen einrichten

Ziel dieses Praktikumsabschnitts ist es, die Stärken und Schwächen des Teams zu analysieren, die Arbeitsfähigkeit des Teams herzustellen, Rollen zu definieren und zu verteilen, die Projektressourcen einzurichten und die Webseiten des Projekts einzurichten.

Die Stärken und Schwächen jedes Teams ergeben sich aus den Stärken und Schwächen der einzelnen Teilnehmer. In der gemeinsamen Arbeit ist es wichtig, die individuellen Stärken im Team zur Geltung zu bringen und die individuellen Schwächen zu kompensieren.

> Tragen Sie die Stärken und Schwächen der einzelnen Mitglieder im Markdown-Format in einer Datei **StaerkenUndSchwaechen.md** im git-Repo zusammen. Jedes Teammitglied hat dabei wenigstens einen Commit selbst erstellt und gepusht.

Besetzen Sie auf dieser Basis die Rollen **Projektleiter** und **Stellvertreter**. Besprechen Sie, welche weiteren Rollen im Team besetzt werden sollen.

Die **Besetzung der Rolle des Projektleiters** mit einer Person, welche die Fäden straff in der Hand hält, ist für den Erfolg der Projektarbeit besonders entscheidend. Es geht dabei eher um organisatorisches Geschick und Durchsetzungsfähigkeit als um detaillierte fachliche Kenntnisse. Sie sollten deshalb diese Rolle mit besonderer Sorgfalt und (möglichst) für die gesamte Dauer des Praktikums vergeben.

Der Projektleiter ist erster Ansprechpartner für Betreuer und Tutor.

Vorgehen und Abgabe

Legen Sie für die Arbeiten zu diesem Punkt (Projektressourcen einrichten, Webseiten aufsetzen, Rollen besetzen usw.) Issues an und fassen Sie diese zu einem Meilenstein „Projektumgebung aufgesetzt“ zusammen.

Zum Abgabetermin A1 prüft der Tutor, ob die Issues angemessen und vollständig sind, alle Arbeitsaufträge abgearbeitet wurden und die Materialien wie gefordert im Team-Repo sowie auf den Webseiten vorhanden sind.

Zum Abgabetermin A1 sind noch keine Dokumente abzugeben.

2.2 Recherche zum Projektthema

Ziel dieses Praktikumsabschnitts ist die genauere Erschließung des Themas durch eine **Recherche zum Projektthema**, mit der sich das Team mit dem Umfeld der Themenstellung genauer vertraut macht, um auf dieser Basis mit dem Auftraggeber die *Projektvision* zu verhandeln. Eine solche Klärung ist auch für die gruppeninterne Kommunikation wichtig, denn so wird deutlich, wie weit im Team einheitliche Vorstellungen über die mit der Aufgabenstellung verbundenen Begrifflichkeiten sowie ein detailliertes Bild der Anforderungen entwickelt werden konnten.

Wichtige Grundlage für die Verhandlungen mit dem Auftraggeber (Betreuer) ist die angemessene Durchdringung der Thematik Ihres Projekts und die Identifizierung relevanter Begriffe und Konzepte der Anwendungsdomäne sowie der verschiedenen Aspekte Ihres Themas. Dies soll durch eine thematische Recherche erreicht werden, die in einem **Recherchebericht** zu dokumentieren ist als Grundlage für ein ausführliches **Gespräch mit dem Auftraggeber** zur Klärung offener Fragen und Aspekte des zu bearbeitenden Themas (zeitnah festzulegender Termin).

Der **Recherchebericht** (5 bis 8 Seiten im pdf-Format) soll unter deutlicher Bezugnahme auf Ihr Thema einerseits die Ergebnisse Ihrer konzeptionellen Analyse wiedergeben und andererseits wichtige themenrelevante Begriffe und Zusammenhänge erläutern.

Teilen Sie den Recherchebericht in drei Kapitel *Begriffe* (als Grundlage für das Glossar), *Konzepte* (domänenspezifisches Was und Wie – Rahmenbedingungen) und *Aspekte* (projektspezifische Momente – Gestaltungsspielräume).

Vorgehen und Abgabe

Legen Sie für die Arbeiten zu diesem Punkt Issues an und fassen Sie diese zu einem Meilenstein „Recherchebericht“ zusammen. Zum Abgabetermin A2 prüft der Tutor, ob die Issues vollständig sind, alle Arbeitsaufträge abgearbeitet wurden und die Materialien wie gefordert im Team-Repo sowie auf den Webseiten vorhanden sind.

Weiterhin sind zum Abgabetermin A2 der **Recherchebericht** sowie der **Aufwandsbericht** Ihres Teams über die ersten Wochen des Praktikums abzugeben. Packen Sie beides in einer zip-Datei `<Datum>.zip` zusammen und checken Sie diese im Verzeichnis `Submit` des master-Zweigs Ihres Team-Repos ein.

Bitte beachten Sie die Formatvorgaben für die Abgabe von Textdokumenten auf den Webseiten des Praktikums.

2.3 Arbeitsplan und Qualitätssicherungsplan

Ziel dieses Praktikumsabschnitts ist die Erstellung eine ersten Version von Arbeitsplan und Qualitätssicherungsplan, die als Arbeitsprodukte die inhaltliche und organisatorische Grundlage für die Implementierung bilden. Die Vorgaben zu Arbeitsplan und Qualitätssicherung sind in den Abschnitten 3.3 und 3.4 genauer ausgeführt.

Vorgehen und Abgabe

Legen Sie für die Arbeiten zu diesem Punkt Issues an und fassen diese zu einem Meilenstein zusammen.

Zum Abgabetermin A3 sind abzugeben:

- Arbeitsplan (6 bis 8 Seiten im pdf-Format).
- Qualitätssicherungskonzept (3 bis 5 Seiten im pdf-Format)
- Aufwandsbericht

Packen Sie alles in einer zip-Datei `<Datum>.zip` zusammen und checken Sie diese im Verzeichnis `Submit` des master-Zweigs Ihres Team-Repos ein.

Zu diesem Abgabetermin werden Aufwandsbericht und Qualitätssicherungskonzept sowie **die Führung der Issues durch das Team** bewertet. Die Bewertung des Arbeitsplans geht in die Gesamtbewertung des ersten Reviews ein.

Erstes Review

Im ersten Review wird der Arbeitsplan begutachtet sowie die Vorstellungen Ihrer Gruppe mit denen des Auftraggebers abgeglichen und zur *Projektvision* verdichtet, die für das weitere Vorgehen verbindlich ist. Dazu ist vom Team eine Präsentation vorzubereiten.

Die Review-Sitzungen finden nach dem im Web veröffentlichten Plan statt. Damit sich alle Seiten auf diese Treffen gut vorbereiten können, ist die **rechtzeitige Vorlage** des Arbeitsplans Zulassungsvoraussetzung. Beachten Sie die **Hinweise zum Ablauf eines Reviews** im Abschnitt 2.5.

2.4 Umsetzungsphase

Parallel zur Erstellung von *Arbeitsplan* und *Qualitätssicherungsplan*, spätestens aber nach deren Verabschiedung, beginnt die Umsetzung des Projekts. Hierzu sind in regelmäßigen Abständen (siehe dazu die zeitlichen Vorgaben im Web) *Releasebündel* bereitzustellen, in denen der Projektfortschritt sichtbar wird.

Als Abgabetermin für Releasebündel ist jeweils ein Montag 24 Uhr vorgesehen, um etwaige Arbeiten übers Wochenende noch im Team besprechen und konsolidieren zu können. Die Abgabetermine für die einzelnen Releasebündel sind empfohlene Richtwerte, Abweichungen von diesen Terminen sind mit dem Betreuer explizit zu vereinbaren. Die Releasaebündel werden nach dem Abgabetermin zentral eingesammelt und vom Tutor bewertet.

Nach Abgabe des zweiten Releasebündels ist eine *weitere Präsentation* des Arbeitsstands durch das Team vorgesehen, die mit Betreuer und Tutor besprochen wird. Im Mittelpunkt dieses **zweiten Reviews** steht der Stand der Modellierung an Hand der *Entwurfsbeschreibung* sowie die bisherige Umsetzung des Projekts.

Methodik

Wir orientieren darauf, das Projekt in sechs Iterationsphasen umzusetzen, in denen die Funktionalität Schritt für Schritt erweitert und zu deren Ende der aktuelle Arbeitsstand jeweils in einem **Releasebündel** integriert und dokumentiert wird. Damit soll gewährleistet werden, dass zum Abschluss jeder Iteration die Umsetzung der funktionalen Anforderungen, Dokumentation und Tests zueinander passen, um die Vollständigkeit und praktische Realisierbarkeit der entwickelten Konzepte zu sichern.

Ein solches Vorgehen erleichtert es, Entwurfsentscheidungen schon in einer frühen Phase auf praktische Realisierbarkeit hin zu überprüfen und gegebenenfalls in einer weiteren Iteration mit nicht zu hohem Aufwand zu korrigieren.

In den Iterationsphasen müssen die personellen Projektressourcen klug eingesetzt werden. Es ist unbedingt zu vermeiden, dass nur einzelne Personen zum Projekterfolg beitragen. Das erfordert vorausschauende Planung, Herunterbrechen von größeren, grob grabularen Epics und Stories auf handhabbare Issues und deren parallele Umsetzung.

Ein Issue ist so zu dimensionieren, dass es vom Assignee in einer Woche realisiert werden kann. Der Assignee ist neben dem zu leistenden Beitrag zum Quellcode des Systems auch für die Zuarbeiten verantwortlich, die zur Ergänzung des Entwurfs, der Dokumentation sowie der Suite der Komponententests erforderlich sind. Zum Ende jeder Iteration wird Bilanz gezogen, Code, Entwurf, Dokumentation und Testsuite zu einem neuen Release integriert und die Implementierungsplanung fortgeschrieben.

Implementierungsplan

Um die strategische Dimension der zeitlichen Abläufe der Projektrealisierung im Auge zu behalten, ist im Team-Wiki auf der Basis der Arbeitspakete aus dem Arbeitsplan und der dort enthaltenen Aufwandsschätzungen ein *Implementierungsplan* zu erstellen und fortzuschreiben, in dem ausgewiesen ist, welche Funktionalität in welchen Iterationen umgesetzt werden soll. Der Implementierungsplan detailliert damit die Arbeitspakete und ordnet die Aufgaben einzelnen Iterationen zu. Der Implementierungsplan ist die *Roadmap* des Projekts und zeigt, wie Sie die gestellte Aufgabe mit den verfügbaren Ressourcen in der verfügbaren Zeit erfüllen wollen.

Erstellen Sie eine erste Version des Implementierungsplans als grobgranulare Epics auf der Basis der Arbeitspakete aus dem Arbeitsplan und schreiben Sie diese Planung durch Verfeinerung, Ergänzung und Rescheduling regelmäßig fort.

Die *wöchentlichen Treffen* des Teams mit dem Betreuer und/oder Tutor dienen der inhaltlichen Strukturierung von Epics, Stories, dem Herunterbrechen auf Issues, der Zuordnung von Issues zum aktuellen Meilenstein, der Zuweisung von Issues zu Assignees und der *inhaltlichen* Fortschrittskontrolle.

Aufwandsberichte

In größeren Projekten werden oft strukturiertere Methoden des Zeitmanagements angewendet. Für Teams der Größe unserer Praktikumsgruppen hat sich ein leichtgewichtiges Zeitmanagement einschließlich der Zeitaufschreibung in den geforderten *Aufwandsberichten* bewährt.

Proof of Concept

Zur Darstellung der Leistungsfähigkeit Ihres Teams soll mit dem zweiten Releasebündel eine erste Softwareskizze als *Proof of Concept* (Vorprojekt) fertig gestellt sein, der zeigt, dass Sie mit den wichtigsten Architekturprinzipien Ihres Themas vertraut und bei der Umsetzung des Projekts auf dem richtigen Weg sind. Das Vorprojekt soll direkten Bezug zum Hauptprojekt haben und ist mit dem Betreuer abzustimmen.

Zweites Review

Die Bewertung der Modellierungsstands auf der Basis der aktuellen *Entwurfsbeschreibung* sowie die Darstellung des Arbeitsstands erfolgt im Rahmen des **zweiten Reviews**. Die relevanten Dokumente werden wieder mit Betreuer, Tutor und dem Team durchgesprochen. Das Team bereitet dazu eine Präsentation vor. Damit sich alle Seiten auf diese Treffen gut vorbereiten können, ist die **rechtzeitige Vorlage** der Unterlagen Zulassungsvoraussetzung. Die Review-Sitzungen finden nach dem im Web veröffentlichten Plan statt.

Vorgehen, Abgabe und Bewertung

Jedes Release schließt eine weitere Iteration der Umsetzungsphase ab. Eine solche *Iteration* beginnt mit der Identifizierung und ggf. Verfeinerung von *Issues* entsprechend der *Implementierungsplanung*, deren Zuordnung zu einem (neuen) *Meilenstein* im git-Repo des Teams und der Zuordnung der einzelnen Issues zur Bearbeitung.

Releasebündel sind zu den im Web veröffentlichten Terminen als zip-Datei `<Datum>.zip` entsprechend den Vorgaben im Abschnitt 3.6 zusammenzupacken und im Verzeichnis `Submit` des master-Zweigs Ihres Team-Repos einzuchecken. Jedes Releasebündel schließt eine Iterationsetappe im Projekt ab – die Issues des zugehörigen Meilensteins sollten abgearbeitet oder für das Rescheduling vorgemerkt sein. Zugleich ist die Demoversion zu aktualisieren, so dass deren Funktionalität vom Tutor bzw. Betreuer ohne weitere Probleme getestet werden kann.

Abzugeben bzw. bereitzustellen: Eine zip-Datei des aktuellen Release sowie eine lauffähige neue Demoversion. Weiterhin ist die *Implementierungsplanung* im Team-Wiki zu aktualisieren.

Bewertung: Bewertet wird für jedes Release die organisatorische Umsetzung (Pünktlichkeit der Einreichung, Konsistenz der Unterlagen, Aufwandsbericht, Funktionalität der Demoversion – zusammen max. 7 Punkte), die Fortschreibung des Implementierungsplans sowie das Issue Management (2 Punkte) und die Aktualität der Webseiten des Projekts (1 Punkt).

2.5 Zur Durchführung eines Reviews

Ein Review ist eine manuelle Qualitätssischerungsmethode, in der Arbeitsprodukte begutachtet werden, die in einem fortgeschrittenen, aber noch nicht finalen Stadium vorliegen. Zentrales Ziel eines Reviews ist die Verbesserung der Qualität der untersuchten Arbeitsprodukte.

Mit dem Review sollen deshalb Stärken und Schwächen der vorgelegten Arbeitsprodukte identifiziert werden. Dabei steht der Qualitätsverbesserungsaspekt durch Hinzuziehen externen Sachverstands im Mittelpunkt. Das erfordert natürlich, dass die vorgelegten Arbeitsprodukte bereits einen ausgereiften Stand erreicht haben und interne Diskussionsprozesse weitgehend abgeschlossen sind.

Ein Review gliedert sich in die Phasen Vorbereitung, Review-Sitzung und Nachbereitung. Die wichtigsten Rollen im Review sind die des Moderators (Betreuer), der Autoren (hier: ein Mitglied des Teams als „Wortführer“), der Gutachter (Betreuer und Tutor) und des Protokollführers (Tutor). Die Gutachter arbeiten die vorgelegten Unterlagen vorab durch und notieren entsprechende Fragen. Es geht dabei und auch in der Review-Sitzung um die Identifizierung von Defekten, nicht um die Diskussion, wie diese Defekte beseitigt werden können.

Die im Review zu begutachtenden Arbeitsprodukte müssen allen Beteiligten rechtzeitig vorliegen. Die Beteiligten bereiten sich auf die Review-Sitzung individuell vor und arbeiten dazu die vorgelegten Unterlagen durch. Zur Review-Sitzung stellt das Team in einer Präsentation die Schwerpunkte der vorgelegten Arbeitsprodukte noch einmal vor. Es ist sinnvoll, die Präsentation in einem gruppeninternen Meeting vorzubereiten und dabei genaue Absprachen zu treffen. Zu diesem Meeting können Sie Ihren Tutor hinzuziehen oder sich mit ihm im Vorfeld verständigen.

Für die Review-Sitzung ist eine Zeitspanne von 45 Minuten vorgesehen. Davon stehen 20 Minuten für die Präsentation zur Verfügung, 20 Minuten für die Diskussion und 5 Minuten für die Aus- und Bewertung. Die Review-Sitzung wird vom Betreuer geleitet, der streng auf die Einhaltung des Zeitplans achtet. An der Review-Sitzung sollen alle Teammitglieder teilnehmen. Der Tutor führt ein (inhaltliches) Protokoll. Im Ergebnis werden Hinweise gegeben, welche in der Regel eine Nachbearbeitung der vorgelegten Arbeitsprodukte erforderlich machen. Diese ist bis zu einem festgesetzten Termin auszuführen und beim Tutor abzurechnen.

Als Teil des SWT-Praktikums wird ein Review ebenfalls bewertet, wobei sich die Bewertung aus der Qualität der Dokumente (80 %), der Präsentation selbst (20 %) und der Qualität der evtl. erforderlichen Nacharbeiten zusammensetzt. Zum Ende der Review-Sitzung gibt der Betreuer eine Einschätzung sowie eine Bewertung in Form einer Punktspanne (z.B. 5. . . 7). Der genaue Punktwert ergibt sich aus der Qualität der Nacharbeiten, die vom Tutor durchgesehen werden.

In Einzelfällen kann bei schwerwiegenden Bedenken eine Wiederholung des Reviews angesetzt werden.

2.6 Endabnahme

Bis zur Deadline für die Arbeit an Ihrem Softwareprojekt soll das Release Ihres Teams so weit entwickelt sein, dass alle vorgesehenen Funktionalitäten implementiert ist, sich Designbeschreibung und Testmaterial auf dem aktuellen Stand befinden und auch die Anwender- und Installationsdokumentation, so weit erforderlich, fertiggestellt ist. Dazu empfiehlt es sich, die letzte Iteration als *Projektabschlussphase* komplett für Bug fixing, Tests und Konsolidierung der Dokumentation vorzusehen.

Die relevanten Materialien sind als finales Release in Form eines weiteren *Releasebündels* abzugeben und bilden die Grundlage für die Endabnahme. Die Endabnahme läuft ähnlich wie ein Review ab, nimmt aber etwa 90 Minuten pro Gruppe in Anspruch, davon 40 Minuten Präsentation der inhaltlichen Ergebnisse sowie der Erfahrungen in der Organisation Ihrer Projektarbeit.

3 Beschreibungen der einzelnen Artefakte

3.1 Projektressourcen

ifi-gitlab

Für jedes Team ist im ifi-gitlab `https://git.informatik.uni-leipzig.de` ein git-Repo als Klon unseres Demo-Repos eingerichtet, das der Tutor an das Team durch Eintragen weiterer Mitglieder an das Team übergibt. Dazu müssen Sie Ihren gitlab-Account aktiviert haben, d.h. sich einmal am ifi-gitlab mit Ihren studserv-Daten angemeldet haben.

Das gitlab-System bietet neben der eigentlichen Verwaltung der Quellen über `git` die Möglichkeit,

- *Issues* anzulegen und zu *Meilensteinen* zusammenzufassen,
- ein *Team-Wiki* zu nutzen und
- Prozesse des Continuous Integration einzurichten.

Von all diesen Möglichkeiten soll im Praktikum Gebrauch gemacht werden.

Über diese Repos werden auch die Abgaben eingesammelt, die zu den vorgegebenen Terminen als zip-Datei `<Datum>.zip` zu packen und im Verzeichnis `Submit` des master-Zweigs einzuchecken sind.

Praktikumsserver

Weiterhin steht auf dem Praktikumsserver `pcai042.informatik.uni-leipzig.de` für Ihr Team ein eigener **Projekt-Account** zur Verfügung (mit Datenbank und Port-Range), über den Sie die Fortschritte Ihrer Arbeiten auf den Webseiten des Teams dokumentieren und später die Funktionsfähigkeit Ihres Prototyps demonstrieren.

Die Einrichtung und Verwaltung dieses Accounts liegt in den Händen des technischen Assistenten Ihres Teams. Die Zugangsinformationen erfragt der technische Assistent beim Tutor.

Webseiten des Projekts

Auf den Webseiten des Projekts

`http://pcai042.informatik.uni-leipzig.de/~<gruppenname>`

ist der Fortgang der Arbeit Ihres Teams darzustellen und dort sind auch die erstellten Materialien nach dem Abgabetermin zu veröffentlichen.

Auf der Webseite des Teams sind (wenigstens) folgende Materialien zu veröffentlichen:

- die Teamzusammensetzung und Rollenverteilung,
- allgemeine Informationen über das von Ihnen bearbeitete Projekt,
- die erstellten Dokument-Artefakte als pdf sowie die Releasebündel als zip-Dateien,
- Aufzeichnungen zu den wöchentlichen Treffen mit dem Betreuer/Tutor sowie
- zu gegebener Zeit der Link zu einer Demo der Applikation.

Deploy der Webseiten mit Jekyll (optional)

Es ist möglich, die Webseiten in einem eigenen Ordner im Team-Repo zu verwalten und mit Jekyll zu deployen. Um die diesen Builtservice zu nutzen müssen drei Schritte durchgeführt werden:

1. Es müssen dem Nutzer `wwwrun` Schreib- und Leserechte auf ihrem `~/public_html` Ordner eingeräumt werden. Dies erfolgt mit dem Befehl

   ```
   setfacl -m u:wwwrun:rwx ~/public_html
   ```

2. In ihrem Gruppen-Repository muss ein Ordner `/Webseiten` angelegt werden, in den die auszuliefernden Seiten eingecheckt werden.

3. Ein *push event webhook* muss für den Service mit der URL

   ```
   http://pcai042.informatik.uni-leipzig.de/cgi-bin/buildsite.sh
   ```

   eingerichtet werden.

   Dies geschieht im ifi-gitlab über das Menü mit dem Zahnrad am rechten Rand der Seite unter dem Menüpunkt *Webhooks*. Es muss lediglich die URL eingetragen werden und sichergestellt werden, dass der Haken vor *Push events* gesetzt ist.

Der Buildservice ist vergleichbar zu GitHub pages und erlaubt Ihnen sowohl einfache statisch HTML Seiten zu deployen als auch mit Jekyll[1] Seiten zu erstellen. Wenn Sie ihre Seite mit Jekyll erstellen, muss in der Datei `_config.yml` der Wert `baseurl:` auf `/~<gruppenname>/jekyll/` gesetzt werden. Die durch den Service deployten Seiten sind unter der URL

```
http://pcai042.informatik.uni-leipzig.de/~<gruppenname>/jekyll
```

erreichbar.

Möchten Sie Ihre komplette Projektseite mit dem Deployservice verwalten, können Sie eine einfache Weiterleitungsdatei in `~/public_html` einrichten.

Bemerkung: Es empfiehlt sich, die neue Jekyll Seite im Repository mit den zwei folgenden Befehlen zu erstellen:

```
jekyll new --skip-bundle Webseiten
cd Webseiten && bundle install --path vendor/bundle
```

Bitte checken Sie in diesem Fall den Ordner `vendor/` nicht in das Repository ein. Sie können zusätzlich einen Eintrag für `vendor/` in die Datei `.gitignore` eintragen.

[1] `https://jekyllrb.com/`

3.2 Aufwandserfassung

Für die qualifizierte Führung eines Projekts spielt auch die Aufwandsschätzung, -analyse und -erfassung eine wichtige Rolle, um die personellen Projektressourcen effizient und den jeweiligen Fähigkeiten angemessen einzusetzen. Dazu orientieren wir an folgendem Vorgehen:

- Die auszuführenden Arbeiten werden in Teilaufgaben (Issues) zerlegt und zugeordnet (V: Projektleiter).
- Jedes Teammitglied führt selbst Buch über die aufgewendete Zeit und bewertet den Aufwand (A) sowie die Schwierigkeit (S) der Teilaufgabe auf einer Skala $1 \ldots 5$ (1 = viel zu niedrig, 2 = zu niedrig, 3 = angemessen, 4 = zu hoch, 5 = viel zu hoch). Dazu gehört natürlich auch die Zeit für Projekttreffen und ähnliche Abstimmungs- und Diskussionsrunden.
- Der Projektleiter sammelt diese Informationen regelmäßig ein, prüft sie im Team auf Plausibilität und erstellt den Aufwandsbericht, aus dem ersichtlich wird, welche Teammitglieder für welche Teilaufgaben wie viel Zeit verwendet haben und wie die Arbeit eingeschätzt wurde.
- Die Analysen der abzugebenden Aufwandsberichte sind so zu erstellen, dass sich die Berichtszeiträume *nicht* überschneiden.
- Der Bericht muss in einem standardisierten XML-Format erstellt sein, welches durch das XSchema `Aufwand.xsd` beschrieben ist. Die eingereichten Dokumente müssen valide bzgl. dieser Syntax sein.
- Die Berichte sind im Submit-Verzeichnis Ihrer Gruppe als Datei `Aufwand.xml` abzulegen.

Beispiel für einen Aufwandsbericht:

```
<Analyse von="2016-12-10" bis="2016-12-16" gruppe="uhu17" createdBy="CK"
      createdAt="2016-12-16">
  <done who="alle" A="3" S="2" Zeit="3"> Aufgabenverteilung</done>
  <done who="DB,JM,SS,CK" A="3" S="3" Zeit="1">
     Präsentation Vorprojekt
  </done>
  <done who="SS" A="4" S="4" Zeit="6">
     Ergaenzungen Entwurfsbeschreibung Vorprojekt
  </done>
  ...
  <done who="GK" A="3" S="3" Zeit="2">
     Weitere Goldstandard-Testsätze suchen
  </done>
</Analyse>
```

Im Ordner des OO-Kurses finden Sie die Datei `Aufwand.xsd` sowie als Beispiel die Datei `Aufwand-Beispiel.xml`.

3.3 Arbeitsplan

Der Arbeitsplan ist das zentrale Dokument der Initialphase, an dem sich die weitere Umsetzung des Projekts orientiert. Im Gegensatz zur Projektvision ist dieser Arbeitsplan später noch in Teilen modifizierbar, um Planung und reale Entwicklung im weiteren Verlauf des Projekts aufeinander abzustimmen.

Mit dem Arbeitsplan soll allerdings bereits deutlich werden, wie Sie sich die Umsetzung Ihres Projektes denken, in welche größeren Arbeitspakete sich die Umsetzung untergliedern lässt, wie der Projektfortschritt überwacht werden soll und mit welchem Aufwand Sie für jedes der Pakete rechnen.

Der Zuschnitt der Arbeitspakete soll sich logisch aus der Darstellung von Hintergrund und Einbindung Ihres Projekts (Abschnitt *Voraussetzungen*) ergeben und sich an der Umsetzung der *Projektvision* orientieren. Der Arbeitsplan schreibt damit Ihre thematische Recherche fort.

Vorgaben

Gliedern Sie den Arbeitsplan in die Abschnitte

1. Projektvision
2. Voraussetzungen
3. Designübersicht und Funktionalität
4. Arbeitspakete
5. Vorprojekt
6. Glossar

Stellen Sie im Abschnitt *Designübersicht und Funktionalität* die wichtigsten Rollen und Nutzerszenarien dar.

Entwickeln Sie im Abschnitt *Arbeitspakete* eine Grobgliederung des Projekts. Teilen Sie Ihr Projekt dazu in wenigstens vier Arbeitspakete und geben Sie für jedes Arbeitspaket an, mit welchem Aufwand in Prozent des Gesamtprojekts Sie rechnen. Die geschätzten zeitlichen Aufwendungen sollten sich als inhaltliche Proportionen der Ausführungen zu den einzelnen Arbeitspaketen wiederfinden.

Der Arbeitsplan soll mehr Funktionalität beschreiben als im Rahmen des Projekts implementiert werden kann. Gliedern Sie dazu die Arbeitspakete in Muss- und Kann-Ziele und zeigen Sie, welche Arbeiten evtl. von Nachfolgeprojekten in Angriff genommen werden können. Die Aufwandsrechnung kann also in der Summe 100 % überschreiten.

Erläutern Sie im Abschnitt *Vorprojekt*, was Sie im Rahmen des Vorprojekts umsetzen wollen und welche Arbeitsprodukte zum Ende des Vorprojekts vorliegen sollen.

Übernehmen Sie im Abschnitt *Glossar* eine konsolidierte Version des Glossars aus dem Rechercheberichtund verwenden Sie die Glossarbegriffe in den Ausführungen des Arbeitsplans auf die vereinbarte Weise. Damit soll erreicht werden, dass sich die Begrifflichkeiten (Glossar – was?) auf konsistente Weise in der Planung (Arbeitsplan – wie?) wiederfinden und ein Bruch zwischen konzeptioneller und Planungsebene vermieden wird.

3.4 Qualitätssicherung

Um ein größeres Projekt erfolgreich und termingerecht zu bewältigen ist es sinnvoll, vorab auch einige Energie auf die organisatorische Aufstellung des Teams (die Prozessdimension) zu verwenden und dabei neben der eigentlichen Software-Entwicklung auch das Qualitätsmanagement im Blick zu behalten. Dazu sollen Sie sich in Ihrem Team über die Grundanforderungen an das Projektmanagement verständigen und diese in einem **Qualitätssicherungskonzept** verbindlich fixieren.

Die **Qualität einer Software** wird neben ihren funktionalen Leistungsparametern auch wesentlich durch die Qualität von Entwurf und Code bestimmt. Ein Maß für diese Qualität ergibt sich aus den verschiedenen Anforderungen an die Software, die in unterschiedlichen Phasen des Softwarelebenszyklus entstehen. Dabei sind auch Fragen des Bugfixing, der Wartbarkeit und Änderbarkeit zu beachten, für welche es wichtig ist, dass sich projektfremde Entwickler schnell in Entwurfsaspekte der Software einarbeiten können.

Die Einhaltung einer Reihe von **allgemeinen Prinzipien** unterstützt diese Anforderungen. Es ist sowohl ein Aspekt der Selbstkontrolle als auch Aufgabe des Qualitätsmanagements, die Einhaltung dieser Regeln zu überwachen. Dazu sind die erforderlichen Qualitätsstandards auszuarbeiten, im Team verbindlich zu vereinbaren sowie deren Einhaltung zu organisieren und zu kontrollieren.

Dokumentationskonzept

Die (technische) Dokumentation Ihres Projekts soll sich aus drei wesentlichen Komponenten zusammensetzen,

- der *internen Dokumentation* des Quelltexts durch Kommentare,
- einer *quelltextnahen strukturierten Dokumentation*, die mit einem geeigneten Werkzeug (z. B. `javadoc`) extrahiert werden kann,
- sowie der *Entwurfsbeschreibung* (siehe Abschnitt 3.5), in der alle wichtigen Architekturentscheidungen und Entwurfsaspekte verständlich dargelegt und begründet sind.

Beachten Sie hierzu auch die Ausführungen im Dokument *Softwaredokumentation im Praktikumseinsatz der Abteilung BIS* im Materialordner des Kurses, in dem verschiedene Aspekte genauer erläutert sind, die Einfluss auf das zu entwickelnde Dokumentationskonzept haben.

Coding Standard

Zur besseren Lesbarkeit und schnelleren Verständlichkeit des Quellcodes ist es außerdem sinnvoll, sich an allgemein übliche *Coding Standards* zu halten. Die Durchsetzung von Coding Standards im Rahmen von *Continuous Integration* (CI) erfolgt üblicherweise durch pre-commit-hooks im git Workflow. Mit dieser *konstruktiven Qualitätssicherungsmaßnahme* wird verhindert, dass Code eingecheckt werden kann, der nicht den vereinbarten Coding Standards entspricht.

Ist Ihr Projekt Teil eines größeren Projekts, so sollte der dort verwendete Coding Standard und auch die dafür verwendeten Werkzeuge[2] Anwendung finden. Anderenfalls sollten Sie sich auf einen verbreiteten Coding Standard für Ihre Programmiersprache einigen, für den ebenfalls Werkzeuge existieren.

Dies ist im Dokumentationskonzept ebenso zu fixieren wie die Konzepte und Werkzeuge, mit denen Sie Ihre quelltextnahe Dokumentation verwalten, damit alle Coder im Team diese Rahmenbedingungen bei ihrer Arbeit beachten und die vereinbarten Werkzeuge nutzen.

Testkonzept

Das Testkonzept berücksichtigt die Qualität der einzelnen Teile Ihrer Entwicklung (Komponententests) sowie die Qualität der Zusammenführung der Teile (Integrations- und Systemtest).

Im Mittelpunkt von **Komponententests** steht die Funktionalität der einzelnen Komponenten, die am besten durch eine gut überlegte Auswahl von Testfällen überprüft werden kann. Ziel von Komponententestfällen ist es, einen Indikator dafür zu haben, ob eine Komponente oder Teile davon spezifikationsgemäß funktionieren. Dazu dient eine Suite von Testfällen, mit denen alle mehr oder weniger wichtigen funktionalen Aspekte und Programmzweige abgedeckt werden und die vor dem eigentlichen Programmieren der Komponente oder der Änderungen erstellt sein sollten. Mit entsprechenden Werkzeugen kann auch der Grad der Testabdeckung (test coverage) bestimmt werden.

Für das (dokumentierte) Ablaufen solcher Testfälle ist eine Umgebung erforderlich, in der die Abhängigkeiten der Komponenten schon zu einer Zeit aufgelöst werden können, zu der das gesamte System noch nicht fertig ist. Für derartige Szenarien gibt es verschiedene Frameworks wie *JUnit*, *Selenium* oder *phpunit*, die auch automatisierte Tests erlauben, die mit entsprechenden Testwerkzeugen wie *Jenkins*[3] in einen CI-Prozess integriert werden können. Continuous Integration als konstruktive Qualitätssicherungsmaßnahme garantiert hierbei, dass nur Code ins Team-Repo eingecheckt werden kann, der alle Tests erfolgreich passiert hat.

Ist Ihr Projekt Teil eines größeren Projekts, so sollte das dort verwendete Test-Framework und dessen Werkzeuge für Ihre Tests nachgenutzt und angepasst werden.

Neben automatisierten Tests spielen in fortgeschrittenen Projektphasen auch manuelle Tests insbesondere verschiedener GUI-Funktionalitäten eine wichtige Rolle.

Fehler, die während der Tests gefunden werden, sind zu erfassen und als Issues aufzubereiten, damit die weitere Fehlerbeseitigung verfolgt werden kann.

Vorgaben

Die wesentlichen Eckpunkte der Qualitätssicherung Ihres Projekts sind im **Qualitätssicherungskonzept** zu fixieren.

Neben inhaltlichen Vereinbarungen gehören dazu auch organisatorische Festlegungen über Termine und Verantwortlichkeiten für Zuarbeiten sowie über das Zusammenführen dieser Teile zu einem Gesamtbild.

Gliedern Sie Ihr QS-Konzept in die Abschnitte *Dokumentationskonzept*, *Testkonzept* und *Organisatorische Festlegungen*.

[2] etwa `phpcs` und `phpcbf` für PHP-Code.

[3] `https://jenkins.io`

3.5 Entwurfsbeschreibung

Die Entwurfsbeschreibung ist ein zusammenfassendes Dokument, das alle wichtigen Informationen zu den Struktur- und Entwurfsprinzipien der Software enthält, welche ein durchschnittlicher, bisher mit dem Projekt nicht befasster Programmierer kennen sollte, bevor er Änderungen oder Ergänzungen des Quellcodes vornimmt (und dabei die detailliertere, etwa javadoc basierte Dokumentation des Quellcodes nutzt). Die Entwurfsbeschreibung ist damit der Einstiegspunkt für qualifizierte externe Dritte, die sich mit der Umsetzung Ihres Projekts genauer vertraut machen möchten.

In der Entwurfsbeschreibung sind deshalb alle wichtigen Modellierungs-, Struktur- und Designentscheidungen zu begründen. Erläutern Sie dabei insbesondere, mit welchen Entscheidungen Sie welche Konzepte und Ansätze umgesetzt haben.

Im Praktikum ist eine Entwurfsbeschreibung in folgende Hauptpunkte zu gliedern:

1. Allgemeines
2. Produktübersicht
   Beschreibung der äußerlichen Funktionsmerkmale des Systems.
3. Grundsätzliche Struktur- und Entwurfsprinzipien
   Was sollte eine Informatikerin über das Gesamtsystem wissen, ehe sie sich allgemeinen Details zuwendet?
4. Struktur- und Entwurfsprinzipien einzelner Pakete
   Was sollte ein Informatiker über 3. hinaus wissen, ehe er sich Details eines speziellen Pakets zuwendet?
5. Datenmodell
6. Glossar

3.6 Releasebündel

Mit einem iterativen Entwicklungsmodell soll das System als eine Folge von Release-Versionen entwickelt werden, in denen die Funktionalität Schritt für Schritt erweitert wird. Es ist besonders wichtig, *während* jeder Iteration arbeitsteilig vorzugehen, um die gesetzten zeitlichen Rahmen einzuhalten. Dazu muss alles genau geplant und sinnvoll aufgeteilt, am Ende aber auch geprüft werden, ob alles zusammenpasst. Jede Iteration umfasst deshalb neben einer Arbeits- auch eine Integrationsphase, in der geprüft wird, dass bzw. ob alles zusammenpasst *und auch Dokumentation und Tests auf dem aktuellen Stand* sind.

Zur Umsetzung dieser Integrationsanforderung werden zum Abschluss jeder Iteration die Zuarbeiten zu einem neuen lauffähigen (!) Release mit erweiterter Funktionalität integriert. Das hat den Vorteil, dass bereits während der Entwicklung dem Auftraggeber der Prototyp mit eingeschränkter Funktionalität vorgeführt werden kann. In der Praxis wirkt sich das vertrauensfördernd aus und erlaubt es, Anforderungen auf der Basis gewonnener Erfahrungen auch noch unterwegs zu modifizieren, wenn alle beteiligten Seiten mitspielen. Zugleich können mit der Auswahl und Abgrenzung der Tasks Projektteile verschieden priorisiert werden.

Ein **Release-Bündel** besteht aus

- dem Quellcode (mit standardisiertem Inline-Anteil der Dokumentation),
- dem Testmaterial,
- der aktuellen Version der Entwurfsbeschreibung,
- einem aktuellen Testbericht,
- einer aktuellen Aufwandsanalyse sowie
- einer Demoversion des aktuellen Release auf der Webseite Ihres Teams.

  Es kann für einzelne Themen mit dem Betreuer eine andere Form der Präsentation vereinbart werden.

Quellcode, Beschreibungen und Testmaterial sind in jeweils eigenen Verzeichnissen abzulegen und danach alles zu einer gemeinsamen zip-Datei `<Datum>.zip` zusammenzupacken.