Handbuch für Auftraggebende im Bachelor-Praktikum

Version vom 28. August 2023

TECHNISCHE UNIVERSITÄT DARMSTADT

Bachelor-Praktikum Informatik
bp@cs.tu-darmstadt.de
Fachbereich Informatik

Inhaltsverzeichnis

Vorwort

Sehr geehrte Auftraggebende,

vielen Dank, dass Sie sich dazu entschlossen haben, ein Projekt im Rahmen des Bachelor-Praktikums (BP) des FB Informatik anzubieten. Dieses Handbuch bietet einen Überblick über den Rahmen und den Ablauf der Veranstaltung, sowie die Aufgaben, die Ihnen als Auftraggebende (AG) zufallen. Um einen erfolgreichen Ablauf Ihres Projekts zu gewährleisten, bitten wir Sie, dieses Handbuch sorgfältig zu lesen. Die Übersicht in Abschnitt 1 hilft Ihnen dabei. Bei Fragen darüber hinaus sind wir unter `bp@cs.tu-darmstadt.de` gerne für Sie da.

Dieses Jahr veranstalten wir ein Kick-Off Treffen für Auftraggebende, bei dem wir alle wichtigen Informationen zum Bachelor-Praktikum geben und Fragen beantworten. Wir bitten Sie, an einem der beiden Kick-Off Termine teilzunehmen.

Mit freundlichen Grüßen
Ihr BP-Organisationsteam

1 Übersicht

Dieses Handbuch ist wie folgt unterteilt: Abschnitt 2 gibt Hinweise für geeignete Projekte und Details zur Projekteinreichung. Abschnitt 3 gibt einen Überblick über die Bestandteile und Ziele des BP, die beteiligten Personen und ihre Rollen, sowie den Softwareentwicklungsprozess. Abschnitt 4 erläutert die Aufgaben, die Ihnen als AG zufallen. Abschnitt 5 erläutert die Qualitätssicherungsziele und -maßnahmen, die sie mit ihren Gruppen festlegen müssen. Abschnitt 6 unterstützt Sie bei der Bewertung ihrer Gruppe.

Während alle diese Hinweise wichtig sind, empfehlen wir Ihnen besonders die Checklisten aktiv zu nutzen, und die Abschnitte zu Projekteinreichung und Bewertung zu den jeweiligen Zeitpunkten zu beachten.

1.1 Termine im Wintersemester 2023/24

Unten finden Sie die Planung für das Wintersemester 2023/24. Die wichtigsten Termine für Sie:

- **06.10.2023**: Deadline Projekteinreichung
- **23.10.2023, 09:00 Uhr** oder **02.11.2023 16:00 Uhr** Kickoff-Veranstaltung für Auftraggeber*innen (wir bieten 2 Termine an)
- **06.11.2023 - 12.11.2023**: Erstes Treffen mit der Gruppe
- **11.03.2024 - 17.03.2024**: Projektübergabe an Sie

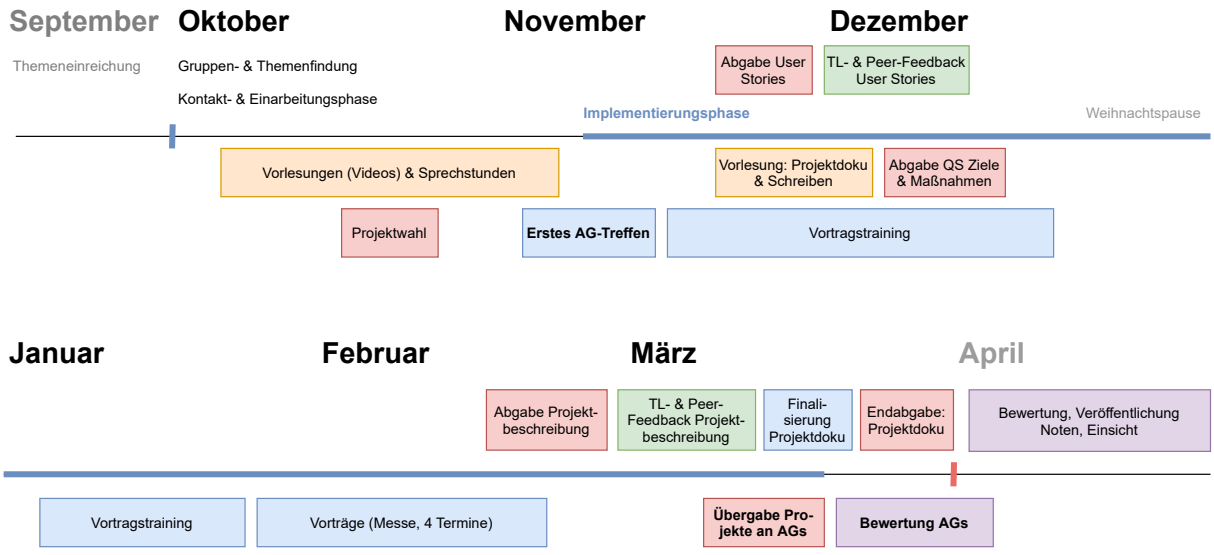

Abbildung 1: Allgemeiner Zeitplan des BPs.

1.2 Checklisten

Vor Beginn der LV

- ☐ Aktuelle Informationen von der Website einholen[1]
- ☐ ggf. Rückfragen per E-Mail klären: `bp@cs.tu-darmstadt.de`
- ☐ Folien zur Projektvorstellung erstellen
- ☐ Thema einreichen, für das Wintersemester 2023/24 unter `https://goto.dm.informatik.tu-darmstadt.de/bp/23-AG/`
- ☐ Bestätigung des Projekts durch das Organisationsteam erhalten
- ☐ AG Handbuch lesen
- ☐ Terminvorschläge (>1, min. 1,5 h) in der Woche des ersten Treffens mit der Projektgruppe freihalten
- ☐ Projektvorstellung erstellen und hochladen
- ☐ an einem der beiden AG Kick-Off Termine teilnehmen

Das 1. Treffen mit der Projektgruppe findet in der Woche vom 06.11.2023 - 12.11.2023 statt.

- ☐ Vorstellungsrunde
- ☐ Regelmäßige Treffen vereinbaren (alle 2-3 Wochen, zweites Treffen max. 2 Wochen später)
- ☐ Projektidee besprechen und Rückfragen klären
- ☐ Rahmen klären: Programmiersprache, Framework, Infrastruktur, Lizenz, etc.
- ☐ Aufgaben für den 1. Sprint festlegen (Einarbeiten, vorbereitende Programmieraufgaben, etc.)
- ☐ Ersten Inhalt für User Stories erfassen

Treffen zur Qualitätssicherung findet im Rahmen des 2. Treffens statt. (s. auch nächster Abschnitt) Es sollten min. 45 Min. für QS eingeplant werden. Das Treffen muss spätestens in der Woche vom 20.11.2023 - 26.11.2023 stattfinden.

- ☐ Welche Aspekte des Projekts benötigen besonderes Aufmerksamkeit, was die Qualität angeht?
- ☐ Gibt es innerhalb dieser Aspekte eine für Sie wichtige Priorisierung?
- ☐ Welche Qualitätssicherungsziele ergeben sich daraus?
- ☐ Mit welchen Maßnahmen könnten diese Ziele erreicht werden (die Vorschläge sollten von der Gruppe kommen)?

Detaillierte Erklärungen zur QS finden sich in Kapitel 5.

Ab dem 2. Treffen

- ☐ Fertiggestellte User Stories abnehmen
- ☐ Ggf. neue User Stories schildern
- ☐ User Stories für die nächste Iteration priorisieren

[1] `https://www.informatik.tu-darmstadt.de/studium_fb20/im_studium/studiengaenge_liste/bachelor_praktikum.de.jsp`

Projektabschluss

- ☐ Übergabetermin bis zum 17.03.2024 (einschließlich) vereinbaren.
- ☐ Klären, was bis zum 17.03.2024 in welcher Form abgegeben werden muss.
- ☐ Bis zum 31.03.2024 anhand der Bewertungskriterien (siehe Kapitel 6) eine Punktzahl zwischen 0 und 100 für die Gruppe festlegen und an uns übermitteln.

2 Geeignete Projekte & Projekteinreichung

2.1 Geeignete Projekte

Beim Bachelor-Praktikum handelt es sich um ein Softwareentwicklungsprojekt. Ein Großteil der Aufgabe sollte sich also auf das Entwerfen, Implementieren und Testen von Software beziehen, und weniger auf die Erforschung neuer Sachverhalte (insofern unterscheidet sich das BP also von den Praktikums- und Projektpraktikumskursen, die viele Fachgebiete anbieten). Die Studierenden sollten in der Lage sein, die Aufgaben ohne vertiefte Kenntnisse in der jeweiligen Domäne zu bearbeiten. Informatikkenntnisse sind Ihrerseits nicht erforderlich – Sie handeln aus Ihrer fachlichen Perspektive heraus. Wenn Sie sich unsicher sind, ob Ihr Projekt für das BP geeignet ist, wenden Sie sich bitte an das Organisationsteam.

2.2 Projektvorstellungsfolien

Die Studierenden wählen Projekte gemäß ihren Interessen und Vorkenntnissen aus. Um den Studierenden die Auswahl zu erleichtern, stellen Sie als Auftraggeber*in einen Foliensatz pro Projekt bereit, in dem das Projekt vorgestellt wird. Je genauer die Folien das Projekt beschreiben, desto wahrscheinlicher ist es, dass Sie eine Projektgruppe bekommen, die sich genau für Ihre Themen interessiert.

Beachten Sie bei der Erstellung der Folien, dass diese die Grundlage der Projektwahl für die Studierenden darstellen. Die Folien sollen das Projekt in Kontext setzen und möglichst detailliert beschreiben, was von den Studierenden umgesetzt werden soll. Die Folien werden den Studierenden digital als PDFs bereit gestellt, daher können Sie gerne Links zu existierenden Projekten, Videos etc. einfügen, die für das Projekt relevant sind. Gerne dürfen Sie auch eine Aufzeichnung anlegen in der Sie die Folien und ihr Projekt vorstellen. Die Aufzeichnung sollten Sie dann ebenfalls in den Folien verlinken. Außerdem sollen benötigte Vorkenntnisse und die zu verwendeten Technologien (Frameworks, Programmiersprachen, etc.) enthalten sein, falls diese bereits feststehen. Falls Sie noch nicht auf bestimmte Frameworks etc. festgelegt sind, schreiben Sie dies auch gerne dazu. Den Foliensatz zur Projektvorstellung laden Sie bei der Einreichung Ihres Projektes in unserem Online-System hoch.

2.3 Projekteinreichung

Die Anmeldung erfolgt über ein Onlinesystem. Wenn Sie ein Projekt anbieten wollen, besuchen Sie bitte `https://goto.dm.informatik.tu-darmstadt.de/bp/23-AG/` und füllen das Formular aus. Falls Sie mehrere Projekte anbieten wollen, müssen Sie es mehrfach ausfüllen.

Das Formular fragt zunächst die Kontaktdaten (Email-Adresse und Name) der hauptsächlich betreuenden Person ab. Diese Person sollte während des kompletten Semesters für regelmäßige (Online-)Treffen mit der Gruppe zur Verfügung stehen.

Außerdem müssen ein Titel für das Projekt und eine Kurzbeschreibung angegeben werden. Weitere Details sollten in einer Präsentation zur Projektvorstellung (PDF) enthalten sein (siehe Abschnitt 2.2). Diese dienen genauso wie die darauffolgenden Fragen zur Art und Umsetzung der Anwendung dazu, dass die Studierenden ein Projekt wählen können, dass sie inhaltlich und technisch interessiert.

Hardware, Software oder anderweitiges Material, die ein Projektteam ggf. zur Durchführung benötigt, sind von Seiten der Auftraggebenden zur Verfügung zu stellen. Alle notwendigen Geräte und Systeme müssen zum Beginn der Implementierungsphase Anfang November bereitstehen. Wenn Ihnen nicht ganz klar ist, was für das Projekt benötigt wird oder wer dafür am besten angesprochen werden sollte, helfen wir jedoch gerne weiter. Markieren Sie dazu bitte das entsprechende Feld im Formular, wir werden uns dann im Vorlauf bei Ihnen melden.

Nach dem Ausfüllen und Abschicken des Formulars wird Ihnen automatisch ein Link zugesendet, über den Sie die Details der Anmeldung auch im Nachhinein bis spätestens zum 06.10.2023 noch bearbeiten können. Wir werden das vorgeschlagene Projekt auf seine Eignung für das Bachelor-Praktikum prüfen und freigeben, wofür Sie eine weitere Bestätigungsmail erhalten werden. Falls wir Schwierigkeiten sehen, werden wir uns bei Ihnen melden.

Bei Problemen mit der Anmeldung schicken Sie uns bitte eine Mail.[2] Eine Ankündigung des Projekts über das Ausfüllen des Formulars hinaus ist nicht notwendig.

[2] bp@cs.tu-darmstadt.de

3 Rahmen und Ablauf des Bachelor-Praktikums

Das von Ihnen gestellte Projekt wird von einem Team aus vier bis fünf Studierenden bearbeitet. Die Zielsetzung des Projekts, Diskussion der Anforderungen und des Fortschritts des Projekts mit dem Team, und Betreuung bzgl. Domänenwissen geschieht durch die AGs. Die Organisation und fachliche Betreuung softwaretechnischer Aspekte erfolgt durch den Fachbereich Informatik. Die Organisation innerhalb der Gruppen wird unterstützt durch studentische Teamleitungen, die selbst nicht Teil des Entwicklungsteams sind. Das BP ist verpflichtend für Studierende im Studiengang B.Sc. Informatik.

3.1 Bestandteile des BP

Im Folgenden finden Sie die wesentlichen Bestandteile des BP:

Softwareprojekt Im Mittelpunkt des BP steht das von Ihnen gestellte Softwareprojekt. Ziel der Gruppe ist es, Ihre Anforderungen zu beidseitiger Zufriedenstellung umzusetzen, soweit es in der Zeit und mit den Vorkenntnissen der Studierenden möglich ist.

Projektbegleitende Vorlesung Das Bachelorpraktikum ist eine integrierte Lehrveranstaltung. Die begleitende Vorlesung ergänzt die Veranstaltung *Software Engineering* und dient der Vermittlung und Wiederholung von Kenntnissen, die für die systematische Softwareentwicklung notwendig sind. Insbesondere wird auf die Nutzung von hilfreichen Werkzeugen zur Durchführung von Software-Engineering-Projekten eingegangen. Einen weiteren Schwerpunkt der Vorlesungen ist die Erstellung der Projektdokumentation.

Projektpräsentation Im Laufe des BPs stellt jede Gruppe ihr Projekt und ihre Ansätze zur Qualitätssicherung (QS) in Form von zwei kurzen Pitches vor. In Vorbereitung darauf besuchen die Studierenden ein Vortragstraining. Das Vortragstraining und die Präsentationen dienen der Vermittlung von essenziellen Fähigkeiten zur Präsentation und werden vom BP-Organisationsteam organisiert.

Qualitätssicherung (Dokument und Belege) Der letzte wesentliche Bestandteil des BPs ist die Sicherung der Qualität der Software. Hierzu müssen die Studierenden mit Ihnen gemeinsam drei QS-Ziele festlegen, auf die sie sich im Rahmen des Projekts konzentrieren. Die QS-Ziele werden durch das (i.d.R. regelmäßige) Durchführen von QS-Maßnahmen erreicht. Zwecks Dokumentation des Entwicklungsprozesses und Benotung müssen die Studierenden die Durchführung der Maßnahmen dokumentieren. Am Ende des Projektes gibt jede Gruppe eine Projektdokumentation zur Bewertung ab. Diese enthält u.A. eine Beschreibung der QS-Ziele und -Maßnahmen sowie Belege zur Durchführung der beschriebenen Maßnahmen. Um die Qualität der Software an die jeweiligen Projektanforderungen anzupassen, fordern wir Sie auf, im Rahmen der Projekttreffen mit der Gruppe gemeinsam die für Ihre Anwendung wesentlichen Kernziele festzulegen.

3.2 Zeitaufwand der Gruppe

Das BP sollte in 1000 Personenstunden[3] bei 4 Studierenden bzw. 1250 bei 5 Studierenden durchführbar sein. Dies beinhaltet die Erstellung der Software, Einarbeitungsphase, Arbeiten im Zuge der Vorlesung, so wie die Erstellung der Dokumente und des Vortrags.

3.3 Rolle der Teambegleitungen

Während des BP wird der Studierendengruppe eine Teambeleitung zur Seite gestellt, welche (im Allgemeinen) das BP bereits absolviert hat und in einem höheren Semester studiert. Die Teambegleitung entwickelt nicht selbst an dem Projekt, sondern unterstützt die Studierenden bei der Organisation, der ersten Kommunikation mit den AGs und der Umsetzung des Entwicklungsprozesses. Sie gibt Feedback zu Abgaben im Rahmen der Vorlesung, ist aber nicht an der Notengebung beteiligt. Sollte es Probleme mit der Gruppe geben, die Sie nicht direkt klären können, ist die Teambegleitung Ihre erste Ansprechperson.

3.4 Der agile Softwareentwicklungsprozess

Im Rahmen des BPs nutzen die Studierenden einen agilen Softwareentwicklungsprozess. Das Ziel des agilen Softwareentwicklungsprozesses ist die zügige und zielstrebige Entwicklung der Software unter sich ständig ändernden Anforderungen. Um dieses Ziel zu erreichen, ist es notwendig Entwurfsprinzipien anzuwenden, die die Software flexibel und anpassbar bzw. wartbar machen. Dazu muss man sich entsprechender Entwurfsmuster bewusst sein. Darüber hinaus ist es erforderlich, Vorgehensweisen anzuwenden, die die nötige Disziplin sicherstellen und den Fortschritt ermitteln. Das wesentliche Ziel ist es, Ihnen zu ermöglichen, das Entwicklungsteam in die Richtung zu steuern, welche den höchsten Wert verspricht. Dazu ist es erforderlich, dass man auf Änderungen flexibel reagiert. Wesentliche Eckpunkte sind:

- Die regelmäßige und häufige Zusammenarbeit mit Ihnen als AG.
- Die inkrementelle Auslieferung neuer Funktionalitäten. Hieran wird der Projektfortschritt gemessen.
- Sich selbst organisierende Teams mit motivierten Individuen.
- Die Reflexion des Teams über die Entwicklungsprozesse.
- Das Streben nach technischer Exzellenz und einem guten Entwurf.

Der Entwicklungszeitraum wird in Iterationen (2-3 Wochen) aufgeteilt, in denen Funktionalitäten der Software (als User Stories, s. 3.5.1) entworfen, implementiert und getestet werden. Am Ende der

[3] d.h. 250 h pro Person.

Iteration muss ein Treffen mit Ihnen als AG stattfinden, in dem die Ergebnisse vorgestellt werden. Durch diese Vorgehensweise soll ein schnelleres Testen, ein stetiger Konsens zum Verständnis der Anforderungen an die Anwendung zwischen AG und Team, so wie das schnelle Reagieren auf sich ändernde Anforderungen ermöglicht werden. Sollte eine für die aktuelle Iteration geplante User Story vor Ende der Iteration nicht fertig werden, ist dies **nicht ungewöhnlich**. Sie wird dann in die nächste Iteration verschoben.

3.5 Ablauf während des Praktikums

Zu Projektbeginn Im Folgenden wird davon ausgegangen, dass die einzusetzenden Technologien (Programmiersprachen etc.) bereits gesetzt sind – falls dies nicht der Fall ist, dann kann bei Bedarf erst noch eine Explorationsphase vorangestellt werden. Gleiches gilt, falls das Team nicht über notwendige fachliche Kenntnisse verfügt, oder Ihre Vorstellungen zur Software am Anfang noch sehr vage sind. In diesem Fall bietet es sich an, weitere Techniken zur Anforderungsermittlung (z.B. Anwendungsfälle - Use Cases, offene Interviews oder Szenariotechniken) einzusetzen, um ein breites Verständnis für die zu entwickelnde Anwendung zu bekommen.

Im Rahmen der regelmäßigen Treffen mit Ihnen sind die Anforderungen an die zu entwickelnde Software in Form von User Stories zu erfassen bzw. zu verfeinern. Sie besprechen hierbei die Anforderungen mit den Studierenden, welche diese dann als User Story formalisieren. Im Rahmen des ersten Treffens sollte versucht werden, auf einem hohen Abstraktionsniveau alle wesentlichen User Stories zu erfassen. Die Anforderungen können anhand des Projektverlaufs später angepasst, ergänzt oder ausformuliert werden. Die User Stories, die im Rahmen der ersten Iteration umgesetzt werden sollen, müssen natürlich im Rahmen der ersten Iteration bereits passend heruntergebrochen und ausformuliert werden. Aufbauend auf den erfassten User Stories wird eine Releaseplanung durch die Studierenden für das gesamte Projekt durchgeführt (zwei bis drei Releases). Ein Release ist dadurch gekennzeichnet, dass es einen aus fachlicher Sicht sinnvollen Funktionsumfang umfasst. Diese soll dann im Rahmen des zweiten Treffens mit Ihnen vorgestellt werden.

Ablauf einer Iteration Am Ende einer Iteration treffen Sie sich mit dem Team und besprechen die entstandenen Ergebnisse. In diesem Rahmen stellt das Team die umgesetzten User Stories vor und Sie nehmen diese ab. Auf Ihren Wunsch muss das Team dazu in der Lage sein, am Ende jeder Iteration eine lauffähige Version zur Verfügung zu stellen, damit Sie diese testen können, um ggf. die weitere Entwicklungsrichtung zu bestimmen. Sollten neue Anforderungen aufkommen, so sind diese unmittelbar als neue User Stories zu erfassen. Weiterhin wird besprochen, welche User Stories im Rahmen der nächsten Iteration umzusetzen sind (Planning Meeting). Es ist darauf zu achten, dass das Team sich nur so viele User Stories vornimmt, wie auch umgesetzt werden können. Falls die Geschwindigkeit des Teams noch nicht bekannt ist bzw. stark schwankt, dann ist es sinnvoll, dass Sie den User Stories Prioritäten zuordnen, damit das Team eine genaue Vorstellung davon hat, in welcher Reihenfolge die User Stories umzusetzen sind. Insbesondere sollten große User Stories in mehrere kleinere heruntergebrochen werden, um die Komplexität im Griff zu behalten.

Projektübergabe Um der Gruppe genug Zeit zum Erstellen der Dokumentation zur Qualitätssicherung zu geben, endet das Softwareprojekt bereits am **18. März**. Nach diesem Zeitpunkt sollen keine Implementierungsarbeiten mehr durchgeführt werden.

3.5.1 User Stories

User Stories sind Arbeitspakete, die innerhalb einer Iteration von dem Team bearbeitet werden. User Stories werden durch die Studierenden formalisiert, ein Beispiel einer solchen User Story findet sich in Abbildung 2.

Mit Auftraggebenden abzusprechen

ID	2
Name	Login
Beschreibung	Als Administrator*in muss ich mich am System mittels Benutzername und Passwort authentifizieren können, um Änderungen vornehmen zu können.
Akzeptanzkriterium	Der Dialog zum Einloggen wird korrekt angezeigt und es ist möglich sich als Administrator*in zu authentifizieren. Ungültige Eingaben werden ignoriert und normale Nutzer*innen erhalten nicht die Rolle „Admin“.
Geschätzter Aufwand (Story Points)	3
Entwickler*in	Michaela Müller
Umgesetzt in Iteration	2
Tatsächlicher Aufwand (Std.)	12
Velocity (Story Points / Std.)	0,25
Bemerkungen	-

Abbildung 2: Beispiel einer formalisierten User Story.

Die wesentlichen Bestandteile einer User Story sind:

1. Eine Beschreibung der User Story, die eine Anforderung von Ihnen so zusammenfasst, dass sich das Team und Sie daran erinnern, was genau zu tun ist. User Stories werden ggf. in Tasks (Aufgaben) weiter untergliedert. Die Aufteilung der User Story in Tasks liegt in der Verantwortung der Gruppe. Ein bewährtes Format für User Stories ist: „Als <Benutzerrolle> will ich <das Ziel> [, so dass <Grund für das Ziel>]“. Wichtig ist, dass der Inhalt von jedem verstanden wird und jeder eine Vorstellung davon hat, was zu tun ist.

2. Priorität der User Story bzw. ob die User Story gerade in Bearbeitung ist.

3. Ein Akzeptanzkriterium, dass es erlaubt zu beurteilen, ob die User Story vollständig und korrekt umgesetzt wurde.

4. Geschätzter Aufwand - Basis ist die erwartete Komplexität im Vergleich zu anderen User Stories; insbesondere im Vergleich zu bereits implementierten User Stories. Es ist wichtig, dass der

geschätzte Aufwand aktuell ist; sollte zwischen der ersten Schätzung des Aufwands und dem Zeitpunkt an dem die User Story umgesetzt werden soll, mehrere Iterationen liegen, so ist es sinnvoll, die User Story noch einmal zu schätzen.

5. Tatsächlicher Aufwand in Zeitstunden und die Velocity (Story Points / Std.), Letzteres ist für die Berechnung der Geschwindigkeit notwendig. Abweichungen zwischen dem geschätzten Aufwand und dem tatsächlichen Aufwand sind zu erwarten – in der Anfangsphase mehr als in der Endphase. Die Abweichungen sollten über den Verlauf des Projektes hinweg geringer werden.

6. Wer (Entwickler*in) hat die Verantwortlichkeit für die User Story wann übernommen und wann wurde die User Story abgeschlossen/abgenommen.

Weiterhin kann die Rolle der Benutzer*innen erfasst werden, für die die entsprechende User Story von besonderer Bedeutung ist. Im Allgemeinen bedarf die Erfassung von User Stories keiner besonderen Werkzeuge und die Verwendung von – zum Beispiel – einem Spreadsheet ist ausreichend. Es gibt jedoch im Internet auch (freie) Werkzeuge, die verwendet werden können (zum Beispiel das Issue-Tracking in GitLab). Am Ende des Projektes sind die User Stories an die Veranstalter*innen abzugeben abzugeben.

4 Aufgaben der Auftraggebenden

Im Folgenden finden Sie eine Erläuterung der Aufgaben, die Sie als Auftraggeber*in erwarten.

4.1 Betreuung des Projektes

Sie sollten sich direkt am Anfang des Projektes hinreichend Zeit nehmen (ggf. mehrere Stunden), um die wesentlichen Anforderungen an das Projekt mit der Gruppe zu diskutieren und auch grob zu priorisieren. Es ist zu diesem Zeitpunkt weder erforderlich noch sinnvoll, zu versuchen, alle Anforderungen detailliert zu diskutieren oder zu ermitteln. Die Anforderungen sind jedoch soweit zu diskutieren, dass das Team eine sehr gute Vorstellung von dem Projekt als Ganzes bekommt. Die Anforderungen können bei Bedarf „jederzeit“ während des Projekts geändert, ergänzt oder gestrichen werden.

Damit dieser Prozess im Rahmen eines Projektes angewendet werden kann, ist es erforderlich, dass Sie sich über die Projektdauer zu regelmäßigen Treffen (idealerweise alle zwei bis drei Wochen) verpflichten. Es ist Ihre Aufgabe, bei der Auswahl der umzusetzenden User Stories für die nächste Iteration(en) mitzuwirken bzw. diese zu bestimmen. Die Auswahl sollte von Ihnen unter fachlichen Gesichtspunkten erfolgen. Bitte setzen Sie sich regelmäßig mit der Software auseinander, um sicherzustellen, dass die Anwendung Ihren Wünschen entspricht, und um qualitatives Feedback zu der Software zu geben. D.h. vor dem Beginn einer jeden Iteration können Sie ggf. bestehende User Stories anpassen, löschen oder neue hinzufügen. Sie sollten nicht mehr User Stories auswählen, als Zeit zum Umsetzen innerhalb der nächsten Iteration vorhanden ist. Die Information, wie viel Aufwand das Umsetzen einer Story vermutlich verursacht und wie viel Zeit innerhalb der nächsten Iteration zur Verfügung steht, erhalten Sie vom Team.

Sind Sie am Ende einer Iteration verhindert und können deswegen am Planning Meeting nicht teilnehmen, dann liegt es in Ihrer Verantwortung, eine qualifizierte Stellvertretung zu benennen. Wenn erforderlich, kann die Dauer einer Iteration auf eine, zwei oder drei Wochen geändert werden. Die Iteration, die sich über die Weihnachtsferien erstreckt, wird natürlich um die freie Zeit verlängert.

4.2 Bereitstellung von Ressourcen

Um der Gruppe einen schnellen Einstieg in das Projekt zu vermitteln, bitten wir Sie, sich vor dem ersten Treffen mit der Gruppe, Gedanken über die notwendigen Hardware- und Softwareanforderungen zu machen. Dies betrifft insbesonders die Bereitstellung von notwendigen Ressourcen, sowie die Lizenz, unter der die Software entwickelt werden soll. Bitte beachten Sie, dass jegliche Hardware oder Software, die von einer Gruppe zur Durchführung des Projekts benötigt wird, von Ihnen zur Verfügung gestellt werden muss. Selbiges gilt für etwaige benötigte Räume. Eine Infrastruktur zur Kommunikation (Zoom, BigBlueButton) und Versionsverwaltung (GitLab) wird von uns/der TU Darmstadt zur Verfügung gestellt, muss aber nicht zwangsläufig verwendet werden. Im Folgenden

finden Sie eine Liste an häufiger benötigter Hardware oder Software mit einem Hinweis auf mögliche Anlaufstellen der TU Darmstadt:

- Server erlauben es Anwendungen zentral zugänglich zu machen z.B. kommunizieren bei Webanwendungen die Browser/Clients mit dem Server. Zur Entwicklung und späteren Produktivnutzung ergibt es Sinn, einen Server bereitzustellen, der mindestens über das TU-Netz/das Netz Ihrer Arbeitsgruppe zugänglich ist. Falls Sie Systemadministrator*innen in der Gruppe haben, kann diese Ihnen evtl. Server, z.B. eine virtuelle Maschine (VM) bereitstellen. Weiterhin kann man bei der ISP eine VM mit Linux (Debian) anfragen.[4]

- Matlab und andere kommerzielle Software – bitte beachten Sie, dass Sie bei Einsatz von kommerzieller Software wie z.B. Matlab genügend Lizenzen bereitstellen müssen, damit das Team mit diesen arbeiten kann.

4.3 Verwertung der Software

Beim BP handelt es sich um ein Pflichtpraktikum. Es gelten deswegen in Hinblick auf die wissenschaftliche/kommerzielle Verwertung dieselben Regeln, wie sie auch bei Praktika, Abschlussarbeiten etc. gelten. Soll die Arbeit nach Abschluss in einer bestimmten Art und Weise wissenschaftlich/kommerziell verwertet werden, müssen Sie dies bei der Themenvorstellung erwähnen und im Rahmen des ersten Treffens noch einmal mit den Gruppen abstimmen. Seitens des Organisationsteams gibt es keine Vorgaben – dem Organisationsteam muss jedoch ggf. Einsicht in den Code gewährt werden, wenn es zu Problemen im Rahmen der Notenfindung kommt.

Zur Festlegung der weiteren Nutzung der Software sollten Sie sich mit der Gruppe auf eine Softwarelizenz einigen. Bei der Auswahl der Lizenz kann die Website ChooseALicense[5] helfen. Im Folgenden finden Sie einige gängige Softwarelizenzen, sowie deren Implikationen auf die Weiterverwertung (sortiert von keiner zur sehr starker Einschränkung):

MIT[6] – Softwarelizenz der MIT. Die Verwendung der Software ist ohne jegliche Einschränkung möglich. Erlaubt kommerzielle und nicht-kommerzielle Weiterverwertung der Software.

APL[7] – Apache Lizenz (v2.0). Ähnlich zur MIT Lizenz, inklusive Klauseln bezüglich Patentierung der Software. Erlaubt kommerzielle und nicht-kommerzielle Weiterverwertung der Software.

GPL[8] – GNU Public License (v3). Open Source Lizenz. Jegliche Weiterverwendung des Codes ist erlaubt, allerdings muss dieser wieder veröffentlich werden.

[4] https://support.rbg.informatik.tu-darmstadt.de/wiki/de/doku/fachgebiete/high-perfomance-vms
[5] https://choosealicense.com/
[6] https://opensource.org/licenses/mit-license.php
[7] https://www.apache.org/licenses/LICENSE-2.0
[8] https://www.gnu.org/licenses/quick-guide-gplv3.html

AGPL[9] – GNU Affero Public License (v3). Variante der GPL, die speziell auf Webanwendungen ausgerichtet ist.

Sollte keine der oben vorgeschlagenen oder bekannten Open-Souce-Lizenzen für Sie und die Gruppe in Frage kommen, erwarten wir, dass Sie gemeinsam einen Lizenz-Text formulieren, den die Gruppe in ihre Projektdokumentation aufnimmt. Dieser muss übliche Lizenz-Regelungen (Nutzungsrechte, Vertriebsrechte, Vergütung, Gewährleistung, ...) abdecken.

[9] `https://www.gnu.org/licenses/agpl-3.0.de.html`

5 Qualitätssicherung (QS)

In einem Softwareprojekt gibt die Qualitätssicherung einen Rahmen vor, um für das Projekt wesentliche Eigenschaften zu definieren und zu überprüfen. Diese Eigenschaften können sowohl die Funktionalität beschreiben (z.B. mit welcher Genauigkeit korrekte Resultate durch die Software ausgegeben werden), als auch Gesamtanforderungen an das System (z.B. wie intuitiv benutzbar die Anwendung ist). QS-Ziele beschreiben diese Eigenschaften konkret für Ihr Projekt. Wir bitten Sie daher darum, sich kurz mit den folgenden Begrifflichkeiten zu beschäftigen, damit Sie mit der Projektgruppe die, für Ihr Projekt wichtigen QS-Ziele festlegen können.

5.1 Was ist QS?

Qualitätssicherung befasst sich mit der Festlegung von Qualitätszielen, anzuwendenden Standards (z.B. Dokumentationsstandards, Prozesstandards wie z.B. Standard zur Testprotokollierung), Einführung von Qualitätsprozessen und Sicherstellung der Durchführung dieser.[10]

Die Abgaben im BP beinhalten eine Angabe der geplanten QS-Ziele und -Maßnahmen, sowie eine Beschreibung der Umsetzung in der Projektdokumentation. Außerdem muss ein Pitch zur Qualitätssicherung vorbereitet und gehalten werden.

5.2 QS-Ziele (Agil)

Folgende Ziele eignen sich besonders gut für den agilen Softwareprozess d. h. den Entwicklungsprozess, der im Regelfall im Bachelor-Praktikum zum Einsatz kommt. Diese sind nach [1] standardisiert.

5.2.1 Benutzbarkeit

Benutzbarkeit beinhaltet verschiedene Faktoren darüber, wie einfach der Umgang mit der Software ist. Dies beinhaltet die Erlernbarkeit durch die Zielgruppe der Software, d. h. die Einarbeitungszeit in die Software sollte möglichst gering sein. Weitere, subjektive Kriterien können die intuitive Verwendung z. B. durch wiedererkennbare GUI-Elemente und die attraktive Ausgestaltung der Software sein. Die Benutzbarkeit kann beispielsweise über die Anzahl der notwendigen Interaktionen zur Ausführung einer bestimmten Aktion gemessen werden.

Einsatz Personen mit verschiedenen Vorkenntnissen oder eine große Anzahl bzw. häufig wechselnde Personen werden die Software verwenden. Sie sollte deshalb intuitiv bedienbar und schnell erlernbar sein.

[10] Ian Summerville. *Software Engineering*. Pearson 2012, 9. akt. Auflage.

Maßnahmen Um die Benutzbarkeit zu gewährleisten, werden üblicherweise Nutzungsstudien durchgeführt. Beachten Sie bitte hierbei, dass das Feedback der Auftraggeber*in alleine nicht ausreicht, um eine qualitative Benutzbarkeit abzuschätzen. In der Regel sollte solch eine Nutzungsstudie fünf oder mehr Teilnehmende haben und früh genug stattfinden, damit die Möglichkeit besteht, Feedback aus der Nutzungsstudie in die Software zu integrieren. Im Gegensatz zu anderen, regelmäßig durchgeführten Maßnahmen, sind ein oder zwei Nutzungsstudien im Rahmen des BP ausreichend. Man kann aber auch die Benutzbarkeit anhand von regelmäßig berechneten Metriken wie die minimale Anzahl von Klicks bzw. die minimale Verschachtelungstiefe zum Durchführen einer Aktion abschätzen. Eine andere Möglichkeit ist es das Einhalten bestimmter Guidelines zur Benutzerfreundlichkeit festzulegen und dann zu überprüfen, dass diese eingehalten werden.

5.2.2 Effizienz

Effizienz beschreibt die kostenwirksame Nutzung von Ressourcen z. B. bzgl. Speicherverbrauch, Laufzeit, CPU-Auslastung.

Einsatz Effizienz kann sich aus den Hardware-Einschränkungen eines Projekts als Ziel ableiten, aber auch aus Anforderungen an die Laufzeit. Es kann z. B. sein, dass Nutzer*innen eine Software nur dann annehmen, falls die Laufzeit eine gewisse Grenze nicht überschreitet.

Maßnahmen Um die Effizienz der Software zu gewährleisten, sollten Sie am Anfang Kriterien z. B. zur Laufzeit oder zum Speicherverbrauch festlegen. Für bestimmte Projekte ergeben sich solche Einschränkungen auf ganz natürliche Art und Weise (z. B. der Speicherverbrauch darf nicht den verfügbaren Arbeitsspeicher überschreiten). Durch regelmäßige Laufzeitmessungen und Messungen des Speicherverbrauchs (z. B. am Ende von einer Iteration) kann festgestellt werden, ob die Implementierung angepasst werden muss, um die notwendigen Vorgaben zu erfüllen.

5.2.3 Funktionalität

Funktionalität beschreibt den Grad, zu dem die Funktionen der Anwendung die Anforderungen erfüllen. Im Besonderen gewährleistet diese Eigenschaft dass die Software mit einer bestimmten Genauigkeit die korrekten Resultate liefert. Funktionen bilden User Ziele und Aufgaben angemessen ab d. h. die Benutzer*in kann ihre Aufgaben durchführen, wird aber nicht durch unnötige Funktionen behindert. Dieses Ziel sollte nicht mit *Korrektheit* verwechselt werden, welches sich auf die Korrektheit gemäß einer gegebenen Spezifikation bezieht.

Einsatz Funktionalität ist eines der wichtigsten QS-Ziele zur Validierung der Funktionsweise der Software und kann in den meisten Projekten verfolgt werden.

Maßnahmen Die Funktionalität kann mit automatisierten Tests, welche die korrekte Funktionsweise einer Methode oder einer Klasse überprüfen, gewährleistet werden. Weiterhin können auch

manuelle Tests von der jeweiligen Entwickler*in durchgeführt werden, um beispielsweise die korrekte Funktionsweise einer graphischen Oberfläche (GUI) zu gewährleisten. Solche automatisierten und manuellen Tests sind in einem regelmäßigen Abstand (z. B. am Ende einer jeden Iteration) durchzuführen. Sollten Tests fehlschlagen, so ist es wichtig, dass auch die jeweiligen Konsequenzen daraus erkenntlich werden (z. B. Person XY behebt den Fehler in der nachfolgenden Iteration). Bei automatisierten Tests kann eine Testabdeckung berechnet werden als Hinweis dazu ob ausreichend Tests vorhanden sind.

5.2.4 Kompatibilität

Kompatibilität beschreibt die Leichtigkeit, mit der einzelne Softwaresysteme oder -komponenten miteinander kombiniert werden können. Dies bedeutet, dass sie Informationen untereinander leicht austauschen können (*Interoperabilität*) und gemeinsame Ressourcen nutzen ohne sich zu behindern (*Ko-existenz*).

Einsatz Es ist bekannt, dass die Software mit unterschiedlichen Systemen oder Komponenten Informationen austauschen muss oder es wird z. B. Abwärtskompatibilität mit früheren Versionen der Software gewünscht.

Maßnahmen Es können Code Reviews eingesetzt werden bei denen der Code auf Einhaltung von Kommunikationsstandards überprüft wird oder auf andere Eigenschaften, die Komponenten für das konkrete Projekt kompatibel machen wie der geeignete Einsatz von Schnittstellendesigns wie z. B. REST und Standard -Datenformaten wie json. Es können auch automatische oder manuelle Tests zum Testen von Kompatibilität mit anderen Systemen durchgeführt werden z. B. falls eine Webanwendung mit unterschiedlichen Browsern kompatibel sein soll.

5.2.5 Portabilität

Portabilität beschreibt wie einfach es ist die Software auf andere Hardware oder Software Plattformen zu übertragen.

Einsatz Es ist bekannt, dass die Software auf mehreren Plattformen zum Einsatz kommen soll z. B. verschiedenen Betriebssystemen.

Maßnahmen Die Zielsysteme sollten festgelegt und dann sollte überprüft werden, dass die Software auf ihnen ausführbar ist. Es kann auch regelmäßig überprüft werden ob bzw. wie viele Anteile des Codes plattformabhängig sind und ob automatisierte und manuelle Tests auf allen definierten Plattformen laufen.

5.2.6 Datensicherheit

Datensicherheit beschreibt die Eigenschaft eines Systems, Informationsverluste und unbefugten Datenzugriff zu verhindern.

Einsatz Die Software führt Operationen aus oder verarbeitet Daten deren Manipulation oder Zugriff durch Unbefugte kritische Schäden verursachen kann z. B. Kaufoperationen, Speicherung von persönlichen Daten wie medizinischen Daten etc.

Maßnahmen In der Regel bieten viele heutige Frameworks im Webdesign einen eingebauten Schutz gegen gängige Angriffe. Lediglich die Verwendung eines solchen Frameworks reicht allerdings nicht aus, um dieses QS-Ziel zu gewährleisten. Daher sollte in regelmäßigen Abständen (z. B. am Ende jeder Iteration) die Sicherheit der Software mit simulierten Angriffen getestet werden. Außerdem existieren statische Analysetools, die Code auf typische Sicherheitslücken wie SQL-Injection, CSRF oder XSS analysieren. Ergebnisse dieser Analysetools müssen dann überprüft und falls nötig der Programmcode überarbeitet werden. Es können auch Code Reviews durchgeführt werden, die spezielle Elemente des *Secure Software Development* prüfen wie z. B. das Reinigen von Nutzereingaben.

Hinweis Im Englischen gibt es die Unterscheidung zwischen *security* (Datensicherheit) und *safety* (Sicherheit). Während die Datensicherheit die Unversehrtheit der Daten und deren Schutz gegen externe Angriffe sichert, gewährleistet Sicherheit die Unversehrtheit von Personen (o.Ä.). Im Rahmen der Zivilklausel der TU Darmstadt sollte Sicherheit selten als QS-Ziel in Frage kommen. Sollten Sie ein Projekt bearbeiten, in dem Sicherheit ein potentielles QS-Ziel wäre, bitten wir Sie, uns zu kontaktieren.

5.2.7 Wartbarkeit

Wartbarkeit beschreibt die Effizienz mit der der erarbeitete Code weiterverwendet und erweitert werden kann. Erweiterbare Software ist einfach auf neue oder sich ändernde Anforderungen anpassbar. Wiederverwendbare Software(-teile) können in andere Anwendungen integriert werden. An dieses Ziel schließen sich Ziele wie Modularität und Testbarkeit an. Modularität beschreibt inwieweit die Software in unabhängige Komponenten aufgeteilt ist, welche ohne Auswirkungen auf andere Code-Teile verändert werden können. Testbarkeit beschreibt die Leichtigkeit mit der Testkriterien für die Software bestimmt und Tests durchgeführt werden können.

Einsatz Die Software soll von anderen Programmierern weiterentwickelt werden bzw. Änderungen an der Software oder an bestimmten Teilen sind schon jetzt abzusehen.

Maßnahmen Zur Sicherstellung der Wartbarkeit, können statische Analysetools eingesetzt werden um die Codequalität anhand vorher bestimmter Syntaxregeln zu überprüfen (wie z. B. Länge von Codezeilen, Benennung von Parametern usw.). Für viele Programmiersprachen gibt es Codestil -Standards, die so überprüft werden können und die Lesbarkeit des Codes verbessern sollen. Ebenso

können Code Reviews zu regelmässigen Zeitpunkten (z. B. bei Fertigstellung jedes Features) durch die Entwickler*innen durchgeführt werden. Hierbei werden vorher festgelegte relevante Code - Eigenschaften (z. B. bzgl. Dokumentation innerhalb des Codes, Implementierungseffizienz von Schleifen, Datenbankanfragen, Einsatz von Design Patterns usw.) anhand einer Liste abgeprüft. Modularität kann z. B. durch die Berechnung von Kopplungsmetriken überprüft werden. Es kann auch eine Analyse der Abhängigkeiten und Schnittstellen oder der Einhaltung geeigneter Design Patterns bzw. Design Principles im Rahmen von Code Reviews durchgeführt werden.

5.2.8 Zuverlässigkeit

Zuverlässigkeit bezieht sich auf die Wahrscheinlichkeit des ausfallfreien und korrekten Betriebs der Software über einen bestimmten Zeitraum hinweg. Der Begriff umfasst die Verfügbarkeit, die Fehlertoleranz der Software (Robustheit), sowie ihre Wiederherstellbarkeit. Fehlertoleranz bedeutet hier, dass die Software auch unter ungewöhnlichen Betriebssituationen ausführbar ist. Verfügbarkeit heißt, dass die Anwendung zu einem Zeitpunkt bzw. über eine Zeit hinaus funktionstüchtig ist.

Einsatz Es ist besonders wichtig, dass die Anwendung möglichst durchgehend verfügbar ist oder die Software wird in kritischen Systemen eingesetzt.

Maßnahmen Um die Zuverlässigkeit zu gewährleisten, kann die Software beispielsweise mittels regelmäßigen Lasttests überprüft werden. Weiterhin lassen sich auch automatisierte oder manuelle Tests gegen fehlerhafte Eingaben oder Corner bzw. Edge Cases durchführen um ungewöhnliche Betriebssituationen zu simulieren. Nur Code reviews oder Statische Analysen sind hier nicht ausreichend.

6 Bewertung der Projektgruppe

Da die Vergabe von den üblichen Noten (1.0, 1.3, ...) in der Vergangenheit zu Schwierigkeiten hinsichtlich der Berechnung der Gesamtnote geführt hat, wurde das Bewertungsschema angepasst, um eine transparentere Notenberechnung zu ermöglichen. Wir bitten Sie daher uns nach dem letzten Treffen mit Ihrer Gruppe eine Punktzahl zwischen $0 - 100$ zu übermitteln.

Note	Punkte
1.0	$> 95 - 100$
1.3	$> 90 - 95$
1.7	$> 85 - 90$
2.0	$> 80 - 85$
2.3	$> 75 - 80$
2.7	$> 70 - 75$
3.0	$> 65 - 70$
3.3	$> 60 - 65$
3.7	$> 55 - 60$
4.0	$50 - 55$
5.0	< 50

Tabelle 1: Grobe Umrechnungstabelle der Punkte zu Noten.

Im Folgenden finden Sie einige Vorschläge für Bewertungskriterien, um Ihre Bepunktung feingranularer auszugestalten. Bitte beachten Sie, dass die hier genannten Bewertungskriterien Ihrer Unterstützung dienen, Sie aber nicht an diese gebunden sind. Insgesamt sollten Sie dazu in der Lage sein, der Gruppe im Falle einer Einsicht die Zusammensetzung der Note hinreichend erläutern zu können.

- Software:
  - Entspricht das Produkt Ihren Vorstellungen?
  - Beinhaltet es alle erwünschten Funktionalitäten?
  - Gibt es sinnvolle Erweiterungen, die vorher nicht mit eingeplant waren?
- Projektablauf:
  - Sind sie insgesamt mit dem Ablauf und der Organisation des Teams bezüglich des Projekts zufrieden?
  - Kam es zu unerwarteten Verzögerungen? Falls ja, wieso?
- Kommunikation: Waren Sie zufrieden mit der Kommunikation mit der Gruppe?

- Dokumentation: Falls für Ihr Projekt relevant, ist die Ihnen vorliegende Dokumentation ausreichend?
- Weiterverwertung: Haben Sie vor, die Software weiterhin zu verwenden?

Literatur

[1] ISO. „IEC25010:2011 Systems and software engineering–systems and software quality requirements and evaluation (square)–system and software quality models“. In: *International Organization for Standardization* 34 (2011), S. 2910.