Hochschule für
Technik und Wirtschaft
Dresden
University of Applied Sciences

Projektbericht

OPAL DRUCKSERVICE

Project Manager: Francesco Ryplewitz (FR)

Analyst: Paul Rogge (PR)

Architect: Felix Müller (FM)

Developer: Paul Jannasch (PJ), Annabelle Zimmer (AZ), Felix Müller (FM)

Tester: Paul Rogge (PR)

Deployment Engineer: Jan Heelemann (JH)

Technical Writer: Jan Heelemann (JH)

INHALTSVERZEICHNIS

1 PLANUNG

1.1 EINFÜHRUNG (JH)

Dieses Dokument beinhaltet den Projektbericht des „Opal Druckservices“. In diesem werden die genutzten Praktiken und Tools sowie die wichtigsten Ziele, Entscheidungen und Ergebnisse erläutert. Jedes Teammitglied hat für diesen Bericht seine jeweiligen Erfahrungen und Erkenntnisse ausgeführt. Ziel des Berichts soll die Darstellungen des Prozesses, von der Erteilung der Aufgabenstellung bis zur Fertigstellung der finalen Software, sein.

1.2 AUFGABENSTELLUNG DES AUFTRAGGEBERS (JH)

Das System soll es den Studierenden ermöglichen, PDF-Dokumente, welche auf der Plattform Opal verfügbar sind, über die UNIDruckerei der HTW-Dresden auszudrucken. Hierbei soll der Nutzer die Möglichkeit haben zu entscheiden, wie viele Folien jeweils auf eine A4 Seite gedruckt werden sollen (entweder 1,2 oder 4). Außerdem soll eine Hintergrundverarbeitung der PDF-Dateien erfolgen, welche beinhaltetet, dass diese in Graustufen umgewandelt werden, alle Druckaufträge eines Nutzers von einem Tag zu einem Druckjob zusammengefasst werden und ein Deckblatt generiert wird. Dadurch wird für die Studierenden ein schneller und unkomplizierter Druck von Vorlesungsunterlagen ermöglicht, ohne dass dies einen eigenen Drucker bedarf. Des Weiteren soll die Unidruckerei mit Hilfe des Opal-Druckservice die Möglichkeit zur Verwaltung der Druckjobs und Nutzer bekommen, um den allgemeinen Workflow zu verbessern. Weitere Anforderungen an das System finden sich in der Anforderungsanalyse.

1.3 AUSGANGSSITUATION ZUM SEMESTERBEGINN (01.03.2020) (JH)

Die Ausgangsituation zu Beginn dieses Semesters sah wie folgt aus. Die Inception – sowie Elaboration Phase, inklusive der Anforderungsspezifikation und der Use Cases war abgeschlossen (Software Engineering 1). Zu diesem Zeitpunkt wurde von Seitens des Teams noch keine Software geschrieben. Der Prozess des Opal-Druckservices bildete zu diesem Zeitpunkt folgenden Sachverhalt ab: Studierende der HTW konnten über das externe Opal-System einen Druckauftrag absenden. Dies erfolgte durch den Aufruf eines Dialogfenster mit Infotext und der Auswahl der Druckoptionen. Die ausgelösten Druckaufträge mit den entsprechenden Druckdaten wurden an ein Postfach

weitergeleitet. Das System, welches auf dem lokalen Rechner des Auftraggebers lief, sammelte die Aufträge. Jeden Morgen um 5 Uhr wurden alle Druckaufträge in Graustufen umgewandelt, die Seiten nach Wunsch konvertiert und die Aufträge pro Student zu einem Druckjob mit Deckblatt inklusive Barcode zusammengefasst. Da das alte System auf einem lokalen Rechner lief und keine wirkliche Fehlerbehandlung beinhaltete, war dieses relativ Fehleranfällig, wodurch einzelne Druckjobs oft händisch verarbeitet werden mussten. Anschließend wurden die Druckjobs in der HTW ausgedruckt und durch einen Mitarbeiter der Bibliothek der HTW abgeholt und gelocht. Am nächsten Tag konnte sich der Student seinen Druckjob mit Vorlage des Studentenausweises in der Bibliothek abholen. Die Mitarbeiter hatten außerdem die Möglichkeit, den Infotext im Opal anzupassen.

Der Part des Ausdruckens und der Ausgabe der Druckjobs sollte laut unserem Auftraggeber Herrn Naumann bereits im März an die UNIDruckerei übergeben werden. Ein komplettes Neuaufsetzen des PDF-Verarbeitungsprozesses war geplant und eine Benutzerschnittstelle als Oberfläche für die Mitarbeiter der UNIDruckerei vorgesehen. Da der Dienst nur probeweise in Betrieb war, konnte er kostenlos durch die Studierenden genutzt werden. Unser Auftraggeber entschied sich jedoch während des Wintersemesters für die Bezahlung der Aufträge durch die Studierenden, um die bis dato angefallenes Kosten nicht noch weiter zu erhöhen. Die Druckkosten belaufen sich auf 5 ct pro Seite. Der unbenutzte Barcode auf dem Deckblatt sollte in Zukunft eine bargeldlose Bezahlung ermöglichen.

Den Stand der Team Konstellation lesen Sie in Punkt 4.1 in diesem Dokument nach.

1.4 PROJEKTORGANISATION (JH)

1.4.1 Rollenverteilung im Projekt

- Auftraggeber (AG): Prof. Dr.-Ing. Gunther Naumann
- Coach: Prof. Dr.-Ing. Jürgen Anke

Dieses Projekt ist in folgende grobe Aufgabenbereiche unterteilt, welche von benannten Teammitgliedern abgedeckt werden.

- Project Manager: Francesco Ryplewitz
- Analyst: Kai Rudolf Kasimir Löning, Tom Jandke (Backup: Paul Rogge)
- Architect: Felix Müller
- Developer: Paul Jannasch, Annabelle Zimmer (Backup: Francesco Ryplewitz)
- Tester: Paul Rogge (Backup: Paul Jannasch)

- Deployment Engineer: Dennis Gruhlke
- Technical Writer: Annabelle Zimmer

Ab dem 30.03.2020 treten einige Veränderungen der Teamkonstellation in Kraft. Nachfolgend ist die neue Rollenverteilung aufgeführt.

- Project Manager: Francesco Ryplewitz
- Analyst: Paul Rogge
- Architect: Felix Müller (Backup: Francesco Ryplewitz)
- Developer: Paul Jannasch, Annabelle Zimmer, Felix Müller, Paul Rogge
- Tester: Hoang Phuc Vo (Backup: Paul Jannasch)
- Deployment Engineer: Jan Heelemann
- Technical Writer: Jan Heelemann (Backup: Annabelle Zimmer)

Ab dem 13.04.2020 scheidet Hoang Phuc Vo aus dem Projekt aus. Die Rolle des Testers wurde neu vergeben. Die neue Rollenverteilung wurde nachfolgend abgebildet.

- Project Manager: Francesco Ryplewitz (FR)
- Analyst: Paul Rogge (PR)
- Architect: Felix Müller (FM) (Backup: Francesco Ryplewitz)
- Developer: Paul Jannasch (PJ), Annabelle Zimmer (AZ), Felix Müller (FM)
- Tester: Paul Rogge (PR) (Backup: Paul Jannasch)
- Deployment Engineer: Jan Heelemann (JH)
- Technical Writer: Jan Heelemann (JH) (Backup: Annabelle Zimmer)

1.4.2 Kommunikation im Projekt

Die Kommunikation innerhalb der Projektgruppe funktionierte sehr gut. Diese wurde aufgrund der aktuellen Lage (Remotesemester) in wöchentlichen Skype-Meetings realisiert. Kurzfristige Dialoge wurden mittels E-Mail oder Kurznachrichtendiensten realisiert. Die Versionsverwaltung wurde durch ein GitHub-Repository umgesetzt. Diese Variante hat sich als überaus effektiv herausgestellt, da jedes Teammitglied zu jederzeit Zugriff auf die Projektdokumente/Quellcodes hat, nachvollziehen kann wer welche Aufgaben abgearbeitet hat, und bei Fehlern auf die vorherige Version zurückgreifen kann.

Die Kommunikation mit dem Auftraggeber, dem Coach Und weiteren Akteuren rund um das Projekt erfolgte ebenfalls via E-Mail und Skype Meetings. Diese funktionierte meistens problemlos und schnell.

1.4.3 Eingesetzte Tools

Zur Versionsverwaltung und Dokumentenaustausch wurde ein GitHub Repository verwendet.

Zur Dokumentation wurde JavaDoc für den Java-Quellcode und einige Funktionen von GitHub verwendet.

Zur Kommunikation wurden Skype, E-Mail und der Kurznachrichtendienst Whatsapp verwendet.

Weitere Tools welche im Rahmen der Softwareentwicklung unter Linux verwendet wurden:

- pdfinfo
- ghostscript
- pdftk
- pdfnu
- barcode
- ps2pdf
- pdfjam

1.4.4 Eingesetzte Techniken und Praktiken

Dieses Projekt wurde durch das Vorgehensmodell “Unified Process“ organisiert, was zur Folge hat, dass der Opal – Druckauftrag iterativ entwickelt wurde.

Während jeder Iteration hat jedes Teammitglied verschiedenen Aufgaben, welche dem Team zum Projektziel (also der Fertigstellung der Softwarelösung für den Opal Druckservice) verhelfen soll.

Um eine Iteration zu organisieren, erstellt der Projektmanager im Vorfeld immer einen Iterationsplan, in welchem die Ziele der Iteration (2 Wochen), sowie die detaillierten Aufgabenzuordnungen aufgeführt sind. Jene Aufgaben findet man auch in der Workitems – Liste wieder.

Während der Inception- und Elaboration-Phase fanden live-Teamtreffen im ca. Zwei- Wochen-Rhythmus statt. Besonders in der Inception-Phase haben wir zusätzlich zu unseren Teamtreffen auf regelmäßige Konsultationen mit unserem Auftraggeber und den Kollegen der UNIDruckerei gesetzt, um unsere bisherigen Ergebnisse der Analyse und Planung mit deren Vorstellungen und Erwartungen abzugleichen. Gearbeitet wurde meist zusammen in der HTW bzw. wurden kleine Aufgaben vereinzelt zu Hause erledigt.

Zu Beginn des dritten Semesters und der damit beginnenden Construction-Phase gab es bei der Weiterarbeit durch die Corona Pandemie zunächst einige Schwierigkeiten, da live-Teamtreffen und

Treffen mit dem Auftraggeber auf einmal nicht mehr möglich waren. Somit mussten wir unsere Arbeitsweise etwas umstrukturieren. Die Teamtreffen haben wir von fort an Wöchentlich per Skype abgehalten und die Kommunikation mit dem Auftraggeber sowie der UNIDruckerei beschränkte sich auf Telefonate.

2 DURCHFÜHRUNG

2.1 BEGINN (FR)

Zum Beginn des Projektes war einer der ersten Herausforderungen die interne Organisation innerhalb des Teams. Aufgrund dessen, dass in unserem Team Studierende aus drei verschiedenen Studiengängen (Wirtschaftsinformatik, Informatik, Wirtschaftsingenieurwesen) zusammen agierten, kannten sich die meisten Teammitglieder untereinander nicht. Somit war es zu Beginn einer der Hauptaufgaben, alle Teammitglieder untereinander zu vernetzten und sich für eine gute Zusammenarbeit kennen zu lernen. Nach einigen kurzen Gesprächen untereinander, starteten wir mit unserer Arbeit.

Das eigentliche Projekt begann hieran anschließend mit einem ersten Treffen mit dem Themenersteller Prof. Dr.-Ing. Gunther Naumann. Hier wurde uns erstmals das Projekt und der zugehörigen Aufgabenstellung umfassend erklärt, woraufhin wir die eigentliche Anforderungsanalyse beginnen konnten. Aus dem ersten Treffen ergab sich, dass die gewünschte Software bereits in einer Betaversion existierte. Hierbei wies uns der Auftraggeber auf einige wesentliche Schwächen seines Systems hin, wodurch dieser Druckaufträge manuell bearbeiten mussten. Besonders diese manuelle Nachverarbeitung sorgte bei Prof. Naumann für einen hohen Arbeitsaufwand, welche er gerne in Zukunft vermeiden möchte.

Das grundlegende Ziel, aus der Managementperspektive, war eine gute und effiziente Zusammenarbeit zu erreichen, welche abschließend zur Lösung des fachlichen Problems führen soll. Fachlich war das Hauptziel die Aufgabenstellung bestmöglich, fristgerecht und im Einklang mit den Anforderungen des Auftraggebers, der Kollegen der UNIDruckerei sowie der technischen Begrenzungen zu erfüllen.

2.2 ANFORDERUNGSANALYSE (AZ)

Im Anschluss begannen wir während der Inception-Phase alle Anforderungen des Auftraggebers sowie der Kollegen der UNIDruckerei zu sammeln und gemeinsam zu evaluieren, wie die finale Software

agieren soll. Es wurde über die Aufnahme weiterer Funktionalitäten gesprochen. Eine Möglichkeit der Stornierung der Aufträge wurde erwähnt sowie eine Einsicht des Verarbeitungsfortschrittes durch den Studenten. Hier gab es Probleme bei der Vorstellung einer möglichen Umsetzung, da uns zu diesem Zeitpunkt Informationen zur Opal-Schnittstellen fehlten bzw. wir über mögliche Zugriffsmöglichkeiten auf den Opal-Server nicht informiert waren.

Zur Kommunikation mit den Studierenden sollten die bisherigen Informationen über Eingang des Auftrages und Meldung im Fehlerfall übernommen werden. Auch eine Bestätigung des Druckes und eine Erinnerungsmail bei Nichtabholung nach einer bestimmten Anzahl Tagen wurden als Möglichkeiten aufgeführt. Unser Auftraggeber sprach ebenfalls von erweiterten Druckeinstellungen wie Farbdrucke und das Anbieten weiterer Bindungsmöglichkeiten abgesehen vom Lochen an. Diese wurden allerdings außen vorgelassen, da diese nur Zusatzfunktionalitäten mit geringerer Priorisierung darstellten.

Für die Benutzerschnittstelle, welche ursprünglich als Desktopanwendung realisiert werden sollte, wurden vorerst folgende Funktionen zur Verwaltung der Druckjobs durch die Mitarbeiter der UNIDruckerei angefordert: Download der Druckjobs, Druck bestätigen, Druckjob-Übersicht, Suche nach Nutzer, Nutzer sperren und entsperren, Abholung bestätigen und Druckjob löschen. Diese Funktionalitäten sollen nur durch die Mitarbeiter der UNIDruckerei zugänglich und nutzbar sein. Die Lösung hierfür ist ein passwortgeschützter Zugang. Des Weiteren wurde im Infotext eine Information zur Weitergabe der Daten aus dem Opal in Absprache mit Herrn Prof. Westfeld integriert. Dies soll den Datenschutz der Benutzerdaten abdecken.

Die bestehenden Dokumente zur Anforderungsanalyse wurden anschließend unserem Coach zur formalen Kontrolle vorgestellt. Hierbei wies uns dieser darauf hin, dass es bei einer Desktopanwendung zu aktuell nicht abschätzbaren Komplikationen kommen kann. Aufgrund dessen entschieden wir uns für die Verwaltung der Druckjobs mittels einer Webapplikation.

Während der nachfolgenden Elaboration-Phase haben wir viele wichtige Details zur Software geklärt und evaluiert. Unter anderem haben wir über die bestehenden Fehler des aktuellen Systems gesprochen. Es kam häufig zu Fehlern bei der Verarbeitung von alten PDF-Dateien und bei Dateien mit Sonderzeichen im Dateinamen. Eine Robustheit gegenüber speziellen Dateinamen sollte daraufhin umgesetzt werden (siehe Systemwide Requirements: NRAR-1). Dies wurde durch eine Anpassung von Zeichen im Dateinamen sichergestellt.

Im Team wurde diskutiert, wann der Eingang des Druckjobs im Postfach bestätigt werden soll. Hier stand zur Debatte: direkt nach dem Eingang, nach der Vorverarbeitung oder erst bei der Druckjoberstellung pro Druckjob. Wir entschieden uns für eine Bestätigung nach der Vorverarbeitung. Argumente hierfür waren das zeitnahe Feedback des Einganges und eine Bestätigung der bereits

erfolgreichen Vorverarbeitung. Es sprachen jedoch auch Argumente gegen diese Umsetzung: Wenn viele Aufträge von einem Studenten ausgelöst werden, wird sein Postfach unnötig mit vielen Bestätigungen „zugemüllt“. Hier könnte ggf. in Zukunft die Umstellung auf eine Mail pro Job realisiert werden. Umgesetzt wurde außerdem die Zusendung der Erinnerungsmail, wenn der Druckjob nach sieben Tagen noch nicht abgeholt wurde. Dies wird ebenso durch einen Timer ausgelöst und geschieht täglich vor der Druckjoberstellung. Die Sperrzahlen der Nutzer und liegengebliebene Jobs werden hierdurch minimiert.

Eine systemweite Anforderung bezüglich der Wartbarkeit wurde nicht umgesetzt. Hierbei handelt es sich um die Erweiterbarkeit des Systems um mögliche Druckeinstellungen. Dies ist zwar auf der virtuellen Maschine gut umsetzbar, aber durch fehlende Anpassbarkeit auf dem Opal-Server nicht realisierbar.

Durch das Ausscheiden unserer Analysten und des Deploymet Engineers nach dem Wintersemester wurden die Zuständigkeiten der Dokumente der Anforderungsanalyse neu verteilt. Die Use Cases und die systemweiten Anforderungen wurden durch die Developer angepasst. Die Vision durch den Tester und den Projektmanager.

Wegen der klar abgrenzbaren Anforderungen unseres Auftraggebers und der Mitarbeiter der UNIDruckerei gab es glücklicherweise keine widersprüchlichen oder in Konflikt stehenden Anforderungen. Der Auftraggeber stellte Anforderungen an die grundlegende Architektur und den Prozessablauf und die Unidruckerei an die einzelnen Funktionalitäten der Schnittstelle zur Druckjobverwaltung. Alle Systemanforderungen sind im entsprechenden Git-Repository im Visionsdokument nachzulesen.

2.3 ARCHITEKTUR (FM)

Ziel der Elaboration Phase war es nicht nur einen validen Grobentwurf, der nicht nur die Implementierung von Use Cases ermöglicht, zu erstellen, sondern auch Usability (betrifft die Web-Anwendung), Reliabilty, Performance und Supportability und eine Strategie zur technischen Realisierung des Softwaresystems zu erarbeiten.

Wir haben uns dazu entschlossen eine Web-Anwendung als grafische Benutzeroberfläche zu entwerfen, diese ist nur lose mit dem restlichen System verbunden. Deren konkrete Realisierung

könnte ohne Einfluss auf das restliche System modifiziert, erweitert oder komplett ausgetauscht werden.

Modelle und Diagramme haben wir mit Visual Paradigm erstellt. Obwohl dieses Tool mächtig ist und von uns kostenfrei genutzt wurde, hätte der Einsatz von Plant UML ein zeitgünstigeres Vorgehen ermöglicht, da es leichter gewesen wäre, Spezifikationen zu versionieren und modifizieren. Textuelle und Tabellarische Bestandteile der Entwurfsspezifikation wurden als AsciiDoc-Dokumente erstellt.

Neue Entwürfe und Modelle, beziehungsweise wesentliche Änderungen der Entwurfsspezifikation wurden bei jedem der regelmäßigen Team-Meetings diskutiert und bei Bedarf in der nächsten Iteration verbessert oder verfeinert. Aufgrund der besonderen Bedingungen während dieses Semesters haben wir auch Pair-Programming eingesetzt, was unter normalen Bedingungen möglicherweise nicht zwangsläufig geboten gewesen wäre, uns aber durchaus Entwurf und Entwicklung erleichtert haben.

Wir haben uns für die Entwicklung des Hintergrundsystems für Java entschieden. Diese Programmiersprache stellt wesentliche Schnittstellen und Funktionen zur Lösung des Problems (Mail-API, Prozesssteuerung, ProcessBuilder, JDBC) und eine langfristige Einsatzmöglichkeit bereit. Java ist außerdem die Programmiersprache, die von allen Mitgliedern des Teams am besten beherrscht wird.

Da der Opal-Druck-Service bisher relative hohe Gesamtkosten verursacht hat, war es uns wichtig, dass alle von uns in das Softwaresystem eingebrachten Frameworks, Klassenbibliotheken und ausführbaren Dateien quelloffen und kostenfrei sind.

Die eingesetzte Datenbank und das Dateisystem des Servers dienen als Datenschnittstelle zwischen der Benutzeroberfläche für die Uni-Druckerei und dem Hintergrundsystem.

Das Hintergrundsystem ist für das Erfassen neuer Aufträge verantwortlich, diese werden vom Server der BPS GmbH in Form einer E-Mail an ein E-Mail-Konto der HTW Dresden versendet. Das Softwaresystem enthält eine Komponente, die unter Benutzung von IMAP E-Mails von diesem Postfach herunterlädt. Die Verarbeitung der Dokumente entsprechend der Formatierungswünsche des Benutzers, das Prüfen der Benutzer auf Sperrung und das Versenden von informativen E-Mails bezüglich des Bearbeitungsstatus sind ebenfalls Aufgaben des Hintergrundsystem.

Zunächst war es nötig, aus Use Cases und nicht-funktionalen Anforderungen technische Notwendigkeiten möglichst genau und vollständig abzuleiten. Wir haben jedem Use Case und jeder nicht-funktionalen Anforderung die zur Implementierung nötigen Architekturmechanismen und die beteiligten Systemkomponenten zugeordnet. (siehe Abschnitt 4 Architecture Notebook) Zum Beispiel ist es für das Verwalten von Sperrvermerken der Benutzer und das Versenden von Erinnerungs-E-Mails nötig, Benutzerdaten persistent zu machen und gegebenenfalls ändern zu können. Diese Aufgabe wird

von einer Datenbank und entsprechenden Schnittstellen (DataAccessObject und JPA-Repositorys) übernommen.

Wir haben weiterhin für jeden Architekturmechanismus, bei dem sein Ausmaß relevant ist, bestimmt oder geschätzt, wie viele Entitäten oder Beziehungen zwischen Entitäten des Gesamtsystems von ihm abhängen. Unser Themensteller hat uns eine Statistik über den Einsatz des bereits vorhandenen Altsystems im Jahr 2019 bereitgestellt. Wir konnten daraus ermitteln, dass im Jahr 2019 knapp 4000 verschiedene Studierende den Opal-Druckservice nutzten. Wir haben die Annahme getroffen, dass in Zukunft die Zahl der verschiedenen Benutzer gleich bleibt oder zunimmt. Das war die Grundlage für die Entscheidung zur Verwaltung der Benutzerinformationen eine Datenbank einzusetzen, da andere Mechanismen, wie einfache Dateiarbeit in diesem Ausmaß unübersichtlich und ineffizient wären. Diese Entscheidung erlaubt es uns zudem, Metainformationen der Druckaufträge und Druckjobs konsistent und isoliert zu speichern. Durch Einsatz von Table Constraints ist außerdem die (referentielle) Integrität der Daten sichergestellt.

Die konzeptionelle Konkretisierung der Datenbank erfolgte durch ein Entity-Relationship-Modell. Als Abbild der Fachzusammenhänge und damit als Vorlage für das Entity-Relationship-Modell diente ein Domänenmodell. (siehe 6.1.3 Datenbank und Domänenmodell Architektur-Notizbuch).

Im nächsten Schritt wurde die Komponentenkomposition spezifiziert. Dabei war die Betrachtung der Abhängigkeiten der einzelnen Architekturmechanismen notwendig. Ziel war es unter anderem, das Separation-of-Concerns-Pattern bestmöglich einzuhalten. Auf einer hohen Hierarchieebene erfolgt die Gliederung in Hintergrundsystem (Zuständigkeit: Aufträge annehmen, Dokumente bearbeiten), Datenaustauschkomponente (Datenbank und Dateisystem des Servers, Zuständigkeit: Datenpersistenz) und Web-Anwendung (Zuständigkeit: Benutzerschnittstelle für die Uni-Druckerei). Bei der iterativ-inkrementellen Detaillierung der Komponentenstruktur hat sich für uns das C4-Modell bewährt. Durch seinen Einsatz war zu jedem Zeitpunkt im Verlauf des Projekts bei Bedarf möglich eine feinere Hierarchieebene zu modellieren ohne grobe Zusammenhänge aus dem Blick zu verlieren und möglich Abhängigkeiten zu erkennen. („feine“ Komponentenstruktur siehe Abschnitt 8.1 Architecture Notebook).

Schwierigkeiten beim Entwurf gab es bei der Spezifikation der Schnittstelle zwischen dem Opal-Server und unserem Softwaresystem. Durch eine zu ungenaue Anforderungsspezifikation und falsche Annahmen in der Elaboration-Phase konnten wir die Verbesserung der verwendeten Kommunikationstechnik zu spät aufgreifen. Wir waren dadurch gezwungen, für die entwickelte Software weiterhin die bereits vorhandene, aber ineffizientere Schnittstelle über ein E-Mail-Konto der HTW zu nutzen.

Es ist geplant, dass die BPS GmbH eine REST-API zur Übertragung der Druck- und Benutzerinformationen und der PDF-Dateien entwickelt, deren Benutzung im Druckservicesystem erst zu einem späteren Zeitpunkt erfolgt. Dazu sind ausschließlich Änderung an einer Komponente, dem Order Management, notwendig. Alle bisher genutzten Schnittstellen zu anderen Systemkomponenten, wie der Datenpersistenz und der eigentlichen Verarbeitung der PDF-Dateien können weiterhin benutzt werden.

Der erste Entwurf der grafischen Benutzeroberfläche für die Mitarbeiter der Uni-Druckerei wurde am Ende der Elaboration-Phase noch einmal grundlegend überarbeitet. Der Plan, eine Desktop-Anwendung zu entwickeln wurde verworfen. Die Web-Anwendung bietet den Vorteil, dass sie einfacher auszuliefern ist. Die Benutzer benötigen lediglich einen aktuellen Internetbrowser und ein Tool zum Entpacken von komprimierten Verzeichnissen das Installieren zusätzlicher Software ist nicht nötig. Somit ist es bei Updates und Erweiterungen der Benutzerschnittstelle nicht mehr nötig auf einem Rechner der UNIDruckerei Update-Pakete zu installieren. Eine Desktop-Anwendung hätte zudem auf virtuelle Netzwerke der HTW zugreifen müssen, da das Hintergrundsystem auf einem Server innerhalb der HTW ausgeführt wird. Mögliche Auswirkungen auf die Qualität des Datentransfers zwischen Hintergrundsystem und Desktop-Anwendung waren für uns nicht abschätzbar.

2.4 Projektbericht Implementierung (az)

Zur Umsetzung der einzelnen Systemteile haben wir uns in Teilgruppen untergliedert, um so die Zuständigkeitsbereiche stärken abzugrenzen. Die Entwickler waren für Umsetzung der Serveranwendung zuständig. An der Webanwendung haben neben dem Architekten als Hauptentwickler auch der Tester und der Projekt Manager als Unterstützung mitgewirkt. Diese Einteilung wurde aufgrund der Rollenkapazitäten getroffen. Die Entwicklung der Serveranwendung stellte sich als umfangreich heraus und die Kapazitäten der Developer waren voll ausgeschöpft. Daraufhin übernahmen andere Rollen, bei denen noch Freiräume vorhanden waren, die Umsetzung der Webanwendung. Die Implementierungsfortschritte der einzelnen Teilkomponenten wurden bei jedem Meeting ausgetauscht und durch den Architekten abgesegnet.

2.4.1 Datenbank

Zu Beginn der Construction Phase wurden in der Datenbank entsprechende Relationen angelegt. Beispielsweise wurde eine Relation für die Benutzerzuordnung angelegt, um bei Nichtabholung ggf.

eine Sperrung bestimmter Nutzer durchführen zu können. Zugleich wurde eine für die Druckaufträge eingerichtet, um Angaben wie Dateiname und Seitenzahl zu erfassen, sowie eine für die Druckjobs, welche die Daten der Druckaufträge aggregiert, daraus das Deckblatt erstellt und die Daten für die Webapp bereitstellt. Um eine Zuordnung der Druckaufträge und Druckjobs zu den Benutzern herstellen zu können, wurden diese jeweils mit Fremdschlüsselbeziehungen versehen. Der Aufbau des Datenbankschemas erfolgte auf Basis der Anforderungsspezifikationen der Analysten. Dieses stützte sich auf die eingehenden Daten der Aufträge aus dem OPAL und der umzusetzenden Funktionalitäten der Druckjobverwaltung. Nachträglich wurde hier noch in der Phase Transition die Speicherkapazität des Attributes Dateiname_origininal der Relation Druckauftrag erweitert. Hier kam es zu einem Fehler beim Einfügen eines Datensatz aufgrund eines zu langen Dateinamens. Auf Wunsch der Mitarbeiter der Uni-Druckerei wurde noch nachträglich ein Attribut hinzugefügt, um den Druckstatus der Dokumente anzugeben.

2.4.2 Server mit VM

Shell Skripte

In Absprache mit dem Architekten fiel die Entscheidung, die Hauptfunktionen der PDF-Verarbeitung als Shell Skripte umzusetzen. Zur Debatte stand noch eine komplette Umsetzung über Java-Programme. Da auf der VM Linux läuft und hier eine Reihe vorgefertigter Tools wie Ghostskript und pdf nup bereitstehen, welche nach individuellem Bedarf per Kommandozeilenaufruf genutzt werden können, haben wir uns recht schnell für diese Variante entschieden. Auf dessen Grundlage entstanden am Anfang der Construction Phase die Skripte zur Graustufenumwandlung, Seitenkonvertierung, Zusammenfassung, Barcodegenerierung und Deckblatterzeugung. Nach Rücksprache mit dem Tester kam es zu Fehlern bei der Seitenkonvertierung der Dokumente. Nachträglich wurde so noch ein Skript einwickelt, um die Ausrichtung einer PDF-Datei zu erkennen. Dies ist wichtig, um die Seitenkonvertierung korrekt je nach Hoch- oder Querformat durchzuführen. Eine Überprüfung der PDF-Dateien auf Passwortschutz wurde ebenso ergänzt, da passwortgeschützte Dateien nicht verarbeitet werden können und zu Fehlerfällen führen. Die Informationen auf dem Deckblatt, welche im Altsystem neben der s-Nr., Anzahl der Seiten, Anzahl der Dateien und dem Preis ebenso eine Auflistung aller Dateien mit Größe beinhaltete, wurden nach Absprache mit dem gesamten Team reduziert. Die Auflistung der einzelnen Dateien wurde weggelassen, da diese schwer umzusetzen ist und der Student auch durch die Seiten- und Dateianzahl seinen Druckjob kontrollieren kann. Außerdem haben wir durch die Mitarbeiter der Unidruckerei in der Transition Phase erfahren, dass der Barcode doch nicht benötigt wird. Durch die fortgeschrittene Zeit und den hohen

Entwicklungsaufwand dieser Komponente haben wir uns dazu entschieden, den Barcode nicht vom Deckblatt zu entfernen, zumal dieser perspektivisch nützlich sein könnte.

Process Handler

Der Zugriff auf die Skripte sollte über einen Process Handler erfolgen. Hier werden die einzelnen Aufrufe via Übergabe der benötigen Parameter wie Dateipfad, s-Nr. und weiteren Druckeinstellungen aus anderen Klassen stattfinden. Nach der Fertigstellung haben wir die Skriptaufrufe zuerst mit fest einprogrammierten Parametern und ohne Übergabe aus anderen Klassen getestet. Die Implementierung dieser Klasse erfolgte zu Beginn der Construction Phase.

Datenbankkomponente

Auf Grundlage der Relationen der Datenbank wurden Grundgerüste von Klassen mit den gleichnamigen Attributen erstellt (Benutzer, Druckauftrag, Druckjob). Der Datenbankzugriff sollte zuerst über JPA erfolgen. Da wir Entwickler aber keine Erfahrung mit dieser Schnittstelle besitzen und wir nur grundlegende Abfragen wie insert, update und delete benötigten, haben wir uns schlussendlich für einen Zugriff mittels JDBC entschieden. Die Datenbankverbindung und ein Interface für die Abfragen wurden umgesetzt.

Mail Client

Zur Realisierung des UseCases „Druckauftrag erteilen“ wurde in der Construction Phase ein Mail Client eingerichtet. Dieser soll das Empfangen und die Weiterverarbeitung der aus dem Opal abgesandten Druckaufträge sowie das Absenden einer Empfangsbestätigung und die Realisierung des UseCases „Erinnerungsmail senden“ umsetzten. Zu Start der Entwicklung wurde zuerst ein Testmailaccount angelegt, um beim Testen der Verbindungen den laufenden Betrieb nicht zu stören. Danach wurden benötigte Librarys heruntergeladen und sich Beispiele von Mail-Clients angeschaut. Das Verständnis über die Verbindung zu einem Account über Javamail wurde sich außerdem erarbeitet. Hierbei stellte die große Anzahl an Mailservern mit unterschiedlichen Protokollen ein Problem dar. Folgend wurde ein erster Client für den Testaccount erstellt und erste Mails ausgelesen. Anschließend erfolgte der Umbau zum Client für die Opal-Druck-Mailadresse. Anfangs wurde nicht direkt aus dem Postfach verarbeitet, sondern aus einem Testorder im Mail-Konto, welche Mails zwischenspeichert. Die Implementierung von Funktionen zum Lesen und Herunterladen der Dateien folgte sowie Funktionen

zum Senden von Mails. Der rekursive Aufbau von E-Mails mit verschiedenen Bodyparts und Multiparts bereiteten hier Probleme. Des Weiteren wurde sich das Verständnis für die Unterscheidung der verschiedenen Mailparts angeeignet und ein Feature für das rekursive Auslesen der Mails geschrieben. Somit war die Umsetzung zur Speicherung, zum Lesen und zum Senden der Mails fertig. Ein Parser wurde geschrieben, um s-Nr. und Einstellungen auszulesen sowie eine Methode hinzugefügt, um den Mailcontent zu setzen. Damit die Dateinamen für die Skripte entsprechend angepasst sind, erstellten wir eine Methode zur Zeichenumwandlung. Die Testung der Sendefunktionalität stellte sich als schwierig heraus, da ein wahlloses Zusenden von Mails an die Studenten nicht zumutbar ist. Die Testung erfolgte so über Testmails. Außerdem ist noch zu erwähnen, dass der IMAPListener mehrmals umgebaut wurde.

Programme zur Verarbeitungssteuerung

Ursprünglich lief die gesamte Druckverarbeitung in einem Schritt ab. Dazu gehörten die Graustufenkonvertierung und Seitenkonvertierung der Druckaufträge sowie die Erstellung des Barcodes und des Deckblattes mit dem Zusammenfassen der Aufträge pro einen Druckjob je Studenten. Die Developer haben sich zu Ende der Elaboration Phase nach Empfehlung des Architekten dazu entschieden, die Druckverarbeitung in zwei Prozesse, die Vorverarbeitung und die Druckjoberstellung zu untergliedern. Die Vorverarbeitung geschieht direkt nach Eingang des Auftrages und beinhaltet die Graustufenkonvertierung sowie die Seitenkonvertierung. Die Druckjoberstellung wird durch den täglichen Timer ausgelöst und enthält das Zusammenfassen der einzelnen Druckaufträge zu einem Job und die Erstellung von Barcode und Deckblatt für jeden Job. Da die Druckaufträge über den ganzen Tag eingehen, ist somit die Last des Systems mehr verteilt. Auch die Druckjoberstellung ist dadurch schneller, da nur noch ein Teil der ursprünglichen Schritte zu einem Zeitpunkt ausgeführt werden müssen. Dies führt zu einer Performanceverbesserung. Hierdurch entfällt die Vorverarbeitung als Use Case, da sie nicht mehr durch den Akteur Timer durchgeführt wird. Sie ist jedoch Vorbedingung des Use Cases „Druckaufträge zusammenfassen“.

Mit der Implementierung der Klasse für die Vorverarbeitung der Druckaufträge wurde zu Beginn der Construction Phase begonnen. In der Vorentwicklung starteten wir mit einer Testfunktionalität, einen Ordner auf das Eintreffen neuer Dateien abzuparsen. Dieses Verfahren sollte später durch das Abparsen des Mailpostfaches ersetzt werden. Danach wurde die eigentliche Funktionalität umgesetzt. Eingehende Mails wurden ausgelesen und im Ordner abgelegt. Der Prozess wurde über einen Trigger gesteuert, welcher ansprang, wenn eine neue Datei im Ordner landete. Später kam uns die Erkenntnis, dass die Verarbeitung auch einfacher funktioniert. Mit einer for-Schleife wird über alle Mails gegangen und jede Mail einzeln verarbeitet bzw. umgewandelt. Jedoch gibt es hier das Problem, dass Mails, die

während der Verarbeitung eingehen, übersprungen werden. Daraufhin fand nochmals eine Umarbeitung statt. Nun werden die eingetroffenen Mails einzeln verarbeitet. Die weitere Entwicklung verlief folgendermaßen: Die Funktionalitäten zum Einfügen der Benutzer wurde umgesetzt, dann jene zum Einfügen des Druckauftrag und anschließend die der PDF-Verarbeitung über Aufruf der Shell Skripte. Nachfolgend wurde die Endverarbeitung nach 5 Uhr umgesetzt. Während eines Probelaufes fiel dem Tester auf, dass bei den Aufträgen mit ungerader Anzahl an Seiten die Aufträge nicht auf separate Blätter gedruckt werden. Daraufhin wurde ein Überprüfen der Seitenzahl im Originaldokument hinzugefügt sowie das Anheften einer leeren Seite bei ungerader Seitenanzahl.

Die Implementierung der Druckjoberzeugung (Endverarbeitung) begann ebenso zu Beginn der Construction Phase. In der Vorentwicklung wurde zuerst eine Klasse zur Auffindung passender weiterzuverarbeitender Dateien erstellt. Die Funktionalität dieser Klasse wurde später in die eigentliche Klasse integriert. Eine eigenständige Klasse war hier logisch nicht sinnvoll. Der Timer für das Starten der täglichen Job-Erzeugung wurde implementiert sowie die Übergabe der Dateien an die Klasse ProcessHandler. Im nächsten Schritt wurde die Dokumentenverarbeitung mittels der Übergabe an die Skripte entwickelt. Ursprünglich war es geplant, alle Dokumente eines Druckjobs gleichzeitig zusammenzufügen. Da dem Process Handler aber keine beliebige Anzahl an Parametern übergeben werden kann, werden nun immer jeweils zwei Dokumente zum Zusammenfassen übergeben. Die zusammengefassten Dokumente werden dann im nächsten Schritt wieder mit einem neuen Dokument verbunden. Danach erfolgte die Implementierung der Datenbankzugriffe mit der Erstellung der Druckjobs aus den Druckaufträgen. Hier musste beachtet werden, dass nicht der gesamte Druckjob abbricht, wenn ein Dokument nicht verarbeitbar ist. Zum Schluss erfolgte die Fehlerfallbehandlung. Durch einen nicht in die Datenbank einfügbaren Druckauftrag kam es zu einem Fehler entstehend durch die Inkonsistenz zwischen den Aufträgen in der Datenbank und den Dokumenten im Ordner. Das Problem wurde durch einen genauen Abgleich der beiden Datenquellen vor der Endverarbeitung gelöst.

Bei der Durchführung der Zusammenfassung muss die Vorverarbeitung gestoppt werden, um Fehler durch gleichzeitig eingehende Aufträge zu vermeiden. Der Timer zum Anstoßen der Druckjoberzeugung wurde im Laufe der Construction Phase in die Vorverarbeitung ausgelagert. Grund hierfür war die geplante komplette Steuerung des Verarbeitungsprozesses über eine Klasse. Da in der Vorverarbeitung ebenso ein Timer umgesetzt werden sollte, welcher um 1 Uhr nachts nach nicht abgeholten Druckjobs sucht, war die Zuordnung so am sinnvollsten. Die Klasse der Vorverarbeitung entwickeltet sich dadurch zur „Hauptklasse“, welche im Dauerbetrieb läuft und die Endverarbeitung triggert. Hier stellte sich die Frage, wie die Steuerung des Dauerbetriebes umgesetzt werden sollte.

Entweder über eine einfache while(true) – Schleife oder über einen Deamon Process. Wir entschieden uns in Absprache mit dem Architekten für die Schleife unter Nutzung der lokalen Zeit, da dies einfacher umzusetzen ist und trotzdem eine gute Performance gewährleistet.

Gegenseitige Klassenaufrufe

Die Klassen Benutzer, Druckauftrag und Druckjob wurden zu Ende der Construction Phase um benötigte Abfragen als Methoden erweitert und diese Methodenaufrufe in den Verarbeitungsklassen umgesetzt. Ebenso wurden die Methodenaufrufe des Mail Clients zur Sendung der Info- und Fehlermails implementiert (Use Case „Erinnerungsmail senden“).

Sonstiges

Die Erteilung des Druckauftrags über die Opal-Schnittstelle wurde vom Vorsystem komplett übernommen, ebenso wie die Bearbeitung des Infotextes im Dialogfenster von OPAL. Der Use Case „Druckauftrag stornieren“, wobei es möglich ist, dass der Student seinen Druckauftrag wieder zurückzieht, wurde nicht umgesetzt. Grund hierfür ist, dass die Interaktion mit den Studenten ausschließlich über das externe Opal-Portal und ausgehend per Mail erfolgen kann. Im Übrigen wurde der Use Case „Druckjob löschen“ nicht umgesetzt. Das Löschen erfolgt immer automatisch nachdem die Abholung bestätigt wurde.

2.4.3 Webapplikation

Folgende Umsetzungen wurden in der Phase Construction vorgenommen. Am Anfang fand eine umfangreiche Einarbeitung in das Spring Framework statt, da keine Vorkenntnisse vorhanden waren. Dies stellte sich als große Herausforderung dar. Danach wurde eine erste Oberfläche erstellt, um sich Datensätze anzeigen zu lassen. Das erste Feature welches implementiert wurde, war das Anzeigen der Druckjobs (Use Case „Druckübersicht öffnen“). Eine Erweiterung zur Druckjobsuche wurde hinzugefügt, sowie die Möglichkeit abgeholte Druckjobs zu kennzeichnen und damit aus der Datenbank zu löschen. Später kam die funktionale Erweiterung hinzu, sich kritische Druckjobs anzeigen zu lassen. Also jene Druckjobs, welche schon sieben Tagen auf die Abholung warten. Zeitgleich wurde eine Sperrfunktion für Nutzer des Services implementiert, welche ihren Druckjob endgültig nicht abgeholt haben (Use Case „Sperrstatus ändern“). Anschließend fügten wir die Downloadfunktion hinzu, welche es ermöglicht, die Druckjobs von der virtuellen Maschine herunterzuladen, um diese ausdrucken zu können (Use Case „Bereitstellung der Dokumente“). Zum Schluss wurde eine Login-

Seite umgesetzt. Diese soll eine Zugriffsbeschränkung auf die Druckjobverwaltung garantieren. Hier können sich die Mitarbeiter per Nutzername und Passwort anmelden. Für die Bereitstellung der Webanwendung wurde eine Apache Tomcat Installation verwendet, welche auf der VM installiert ist.

Nachträglich wurde die Webapp noch um die Funktion erweitert, den erfolgreichen Ausdruck der Dokumente vermerken zu können (Use Case „Druck bestätigen“). Das komplette Datenbankschema musste daraufhin angepasst werden und eine Erweiterung der Abfragen auf der VM durchgeführt werden. Die größten Probleme bestanden im Allgemeinen in der Konsistenzhaltung der Daten sowie in der Sicherstellung der Unveränderbarkeit der Daten durch die Webapp-Nutzer.

2.5 PROJEKTBERICHT TEST (PR)

Ich werde im Folgenden oft die erste Person Plural verwenden um zu verdeutlichen, dass ich als Tester zwar die Verantwortlichkeit für Tests trug, weitreichende Entscheidungen, die andere beeinflussten, jedoch oftmals zusammen entschieden wurden. Ich habe als Tester dafür gesorgt, dem Team mehrere Optionen zu bieten, sodass wir gemeinsam Entscheidungen treffen konnten, die für jeden gut tragbar waren. Ich bin der Meinung, dass Rollen zwar Verantwortlichkeiten schaffen, jedoch keine alleinigen Entscheidungsbefugnisse über die jeweiligen Tätigkeiten bringen sollten.

Zu Beginn wurden die Kernkomponenten der Software ermittelt und aus diesen die Testobjekte definiert. Die Testobjekte entsprechen den Objekten aus dem Architekturentwurf.

2.5.1 Testobjekte

Das erste Objekt stellt die Kommunikation mit Opal und somit die Übertragung der Druckaufträge (zu übermittelnde Datei und für die Verwaltung benötigten Daten) auf den Zielserver dar. Da die bereits existierende Schnittstelle zum Opal-service weiterhin genutzt werden muss (Aufnahme der Randbedingung über den Auftraggeber) und die Entwickler keinen Einfluss auf diese haben konnten, musste ausschließlich die Nutzung der Schnittstelle getestet werden. Die Wahrscheinlichkeit für schwerwiegende Fehler wurde aufgrund dessen als gering eingestuft. Die Schnittstelle liefert die zu druckende Datei und die diesbezüglich notwendigen Benutzer- und Druckdaten. Mit den dazugehörigen Tests wurde überprüft, ob die Daten korrekt erfasst und gespeichert wurden, um somit die weitere Verarbeitung zu gewährleisten. Bei diesem Testobjekt entwickelte sich das Problem, dass wir erst am Ende des Moduls SE 2 die Möglichkeit bekamen, dieses ordnungsgemäß zu nutzen und testen. Da die Funktionalität des Abschickens eines Druckauftrages durch den Kunden (Student) eine

essenzielle Kernfunktion darstellt und die Schnittstelle kontinuierlich vom aktuellen Produktivsystem genutzt wurde, konnten wir diese erst sehr spät in einem kurzen Testbetrieb verwenden. Um jedoch die darauffolgenden Komponenten, wie beispielsweise das Verarbeiten, Zusammenfassen und Übermitteln an die Unidruckerei, sowohl in der Entwicklung als auch während der Tests zu überprüfen, entschieden wir uns für ein Skript, welches einen Systemzustand nach dem Aufgeben von einigen Druckaufträgen durch Kunden herstellt. Dieses Skript sorgte für viel Zeitersparnis bei mir und den Entwicklern, da nun Druckaufträge schnell eingefügt/simuliert werden konnten. Das Testen dieses Objektes verlief, abgesehen von einigen Unstimmigkeiten bezüglich der Dateinamenkonvertierung, reibungslos. Eine Schwierigkeit war jedoch, dass wir nicht wussten welche Daten die Schnittstelle liefert und demnach war es nicht leicht, die Auswahl der Daten für die Simulation festzulegen. Ein Gespräch mit Prof. Dr. Thomas Naumann sorgte jedoch für Klarheit und ich konnte auf Grundlage dieser Informationen das Skript zum Herstellen des Systemzustands erstellen.

Das zweite Testobjekt bezieht sich auf die Verarbeitung der PDF-Dateien, welche eine Kernkomponente des zu entwickelnden Systems darstellt. Hier vermutete ich aufgrund der Komplexität eine hohe Wahrscheinlichkeit für Fehler. Die Verarbeitung beinhaltet das Vorbereiten der PDF-Dateien mittels Graustufenumwandlung, Passwortkontrolle und Seitenbearbeitung. Zusätzlich wird ein Deckblatt als Kundeninformation erzeugt und die Druckaufträge, welche über 24 Stunden eingegangen sind, zusammengefasst. Parallel müssen Daten und der Status bezüglich Benutzer und Druckaufträge kontinuierlich in der Datenbank angepasst werden. Die Integrität der Daten ist von enormer Relevanz. Da die Bearbeitung in mehreren kleinen Schritten mittels einer hohen Anzahl von Skripten realisiert wird, kann bei fehlerhaften Skriptausführungen die weitere Verarbeitung verhindert werden. Der Test von alternativen Abläufen und Randfällen stellte einen verhältnismäßig großen Teil der Tests für dieses Testobjekt dar, um die Belastbarkeit und Qualität im Allgemeinen zu fördern. Die Wahrscheinlichkeit für Fehler und die daraus resultierenden negativen Auswirkung auf das System und das Projekt wurde als sehr hoch eingeschätzt. Die Komponente der PDF-Verarbeitung besteht aus einer Vielzahl eigens testbaren Skripts. Aufgrund dessen entschied ich mich dazu, diese Komponente sowohl feingranularer als auch im Ganzen zu testen, um somit kleinere Detailfehler schneller identifizieren zu können. Dies bedeutet, dass ich sowohl Skripte einzeln als auch in Kombination entsprechend des Standardablaufs im Produktivbetrieb getestet habe. Hierbei erwies sich das Skript zur Simulation von Druckaufträgen erneut als äußerst hilfreich. Das Verarbeiten der Aufträge war die erste vollständig ausprogrammierte Komponente und konnte zu einem frühen Zeitpunkt getestet werden. Leider gab es bei dieser Komponente viele Fehler, da nicht alle Teammitglieder einen ausreichenden Überblick über die Anforderungen hatten. Diese wurden zwar in der Anforderungsspezifikation dokumentiert aber ließen während der Entwicklung Interpretationsspielraum. Das Häufige durchführen von

manuellen Tests und das darauffolgende Verbessern der einzelnen Komponenten sorgte jedoch dafür, dass die Skripte zur PDF-umwandung fehlerfrei funktionieren.

Die dritte Komponente stellt die Webanwendung dar. Diese ermöglicht es den Mitarbeitern der Unidruckerei Druckjobs und Benutzer zu verwalten. Die Webanwendung interagiert ausschließlich mit der Datenbank. Da die Daten jedoch zu jedem Zeitpunkt konsistent sein müssen, muss gewährleistet werden, dass die Webanwendung ausschließlich konsistente Daten erzeugt und die Mitarbeiter keine Möglichkeit haben, dies zu verhindern. Zusätzlich muss die Webanwendung immer erreichbar sein, da sie ein unverzichtbares Werkzeug zur Erfüllung des Systemziels darstellt. Die Wahrscheinlichkeit für Fehler wurde zu Beginn als gering eingestuft, da die Entwickler der Webanwendung an der Erstellung der Anforderungsspezifikation und Architektur direkt beteiligt waren. Aufgrund dessen entschieden wir uns für einen Akzeptanztest mit dem späteren Anwender dieser (Unidruckerei) und einigen ausgewählten manuellen Systemtests. Die Tests bezüglich der Webanwendung verliefen reibungslos und zeigten keine Fehler auf. Der Akzeptanztest, welcher mit den Entwicklern der Webanwendung, Herrn Buschmann der Unidruckerei und mir durchgeführt wurde, verlief gut. Herr Buschmann war mit der Funktionalität, Bedienbarkeit und Anschaulichkeit dieser sehr zufrieden.

2.5.2 Testebenen

Die Tests wurden ebenfalls auf drei verschiedenen Ebenen durchgeführt. Die meisten unsere Tests waren Systemtests, welche das komplette System hinsichtlich der funktionalen und nichtfunktionalen Anforderungen testeten. Diese Tests wurden alle von mir durchgeführt und dokumentiert. Zudem entschieden wir uns für Modultests, um das System mittels Überprüfung der einzelnen Funktionen auf einer niedrigen Testebene auf funktionale und nichtfunktionale Fehler zu überprüfen. Bei den Unit-Tests entschied ich mich gegen eine Priorisierung, da bereits nur die wichtigsten Methoden für Modultests dokumentiert wurden. Ergänzend entschieden wir uns für einen Akzeptanztest, welcher mit der Unidruckerei als Endanwender der Webanwendung durchgeführt wurde. Abgesehen von den Unit-Tests wurden alle Tests manuell ausgeführt, da dies mithilfe von einmalig angefertigten Skripts und einer detaillierten Dokumentation und Anleitung für die jeweiligen Tests gut möglich war. Wir haben uns gegen automatisierte Systemtests entschieden, da dann der Kostenfaktor (Erstellungszeit, Zeit zur Aneignung notwendigen Wissens) höher als der daraus möglicherweise resultierende Nutzen gewesen wäre. Wir konnten somit schnell und unkompliziert weitere Testdurchläufe starten, was sich bei späterer Betrachtung als sehr gute Vorgehensweise herausstellte. Die Wichtung der Arbeitszeit für die einzelnen Testebenen sorgte für große Unklarheit und Unsicherheit bei mir. Für explizite Integrationstest entschieden wir uns nicht, da die entsprechenden Tests sich mit unseren durchgeführten Systemtest überschnitten hätten. Obwohl Modultests äußerst wichtig sind, um

frühzeitig Systemfehler zu erkennen, spielten sie in unseren Tests eine etwas untergeordnete Rolle. Ein Grund dafür ist die Beschäftigung der Beteiligten. Obwohl die Verantwortung der Modultests beim Tester liegt, werden sie von den Entwicklern erstellt und beim Starten der Anwendung automatisch ausgeführt. Die Entwickler waren mit dem Entwickeln der einzelnen Komponenten sehr beschäftigt, wodurch die Zeit für viele Unit-Tests als Modultests fehlte. Bei nachträglicher Überlegung hätten wir auf das Verfahren des testgesteuerten Entwickelns setzen können, um so die Tests besser in die Entwicklung integrieren zu können. Diese Entscheidung hätten wir allerdings möglichst vor dem Beginn der Entwicklung treffen müssen. Zu diesem Zeitpunkt war uns die aktuelle Problematik der Zeit noch nicht bewusst und das Verfahren der testgetriebenen Entwicklung noch unbekannt. Ein weiteres Problem, welches unseren Fokus eher auf Systemtests lenkte, war die geringe Kohäsion und hohe Kopplung in unserer Codestruktur. Uns ist mittlerweile klar geworden, dass dies nicht nur für Tests, sondern auch für Erweiterungen und das nachträglich Verstehen der einzelnen Teilkomponenten nicht sinnvoll ist. Das Erhöhen der Kohäsion und das Verringern der Kopplung war uns zu diesem Zeitpunkt aufgrund fehlender Zeit nicht mehr möglich, wodurch wir die Entscheidung trafen, nur einige weniger aber dennoch mit Bedacht gewählte Methoden zu testen.

2.5.3 Organisation

Zur Organisation unserer Tests benutzten wir ein Test-Case-modell, welches unter anderem den Zweck der Tests in einem Fließtext kurz beschreibt. Zusätzlich wurde die Teststrategie in einem weiteren Fließtext festgehalten. Anschließend folgen vier Tabellen entsprechend der drei Testebenen (Modultests, Systemtests und Akzeptanztests), wobei die Systemtests noch in funktionale und nicht funktionale Systemtests untergliedert wurden. Um effektiv zu testen wurden alle Tests priorisiert und die Test Cases in der dementsprechenden Reihenfolge erstellt. Aufgrund von Zeitmangel stellte sich die Priorisierung der Tests als sehr sinnvoll heraus, da so die wichtigsten Funktionalitäten und nichtfunktionale Eigenschaften zuerst getestet wurden. Da die Test Cases für funktionale Tests von den Use Cases abgeleitet wurden, stellte ich ebenfalls sicher, dass diese zu jeder Zeit aktuell sind. Die nichtfunktionalen Tests leiteten sich von den nichtfunktionalen Anforderungen, welcher in der supplementary specification dokumentiert waren, ab. Im Test-Case-modell wurde ebenfalls der Verantwortliche und der durchführende jedes Tests dokumentiert.

Abgesehen von der Implementierung/Durchführung der Unit-Tests war ich als Tester für die Verantwortung und Durchführung aller Tests verantwortlich. Um die Übersicht über den aktuellen Stand der einzelnen Tests zu bewahren, wurde in der Tabelle ein Statusfeld eingebracht, welches folgende Auswahlmöglichkeiten bat: Durchführung ausstehend, Test nicht erfolgreich und Test erfolgreich. Dieser Status zeigt das Ergebnis des letzten Testdurchlaufs, sofern dieser existiert.

Ergänzend zum Status der einzelnen Tests wurde für jeden Test das Datum des nächsten Testdurchlaufs notiert. Jeder Test wird zwei Wochen nach der letzten Testdurchführung oder nach einer Anpassung der beteiligten Komponenten durch die Entwickler erneut ausgeführt. Somit konnten wir ausschließen, dass bereits erfolgreich getestete Komponenten aufgrund von Anpassungen später nicht mehr die geforderten Anforderungen erfüllen. Das Komponenten nach ihrer Anpassung möglichst zeitnah getestet werden, konnte durch eine detaillierte Zeitplanung sehr gut umgesetzt werden. Jedoch fiel es mir als Tester schwer, den zwei-Wochen-Rhythmus bei bereits erfolgreich durchgeführten Tests einzuhalten. Bei wenigen ausgewählten Tests, die aufgrund meiner Einschätzung im Nachhinein keine Fehler mehr zeigen sollten, erhöhte ich das Intervall je nach Situation auf drei bis vier Wochen. Bei der Auswertung der Testdurchläufe herrschte zwischen den Entwicklern und mir während der Team-meetings eine gute und zielführende Kommunikation. Die einzelnen Tests wurden in Test-Cases festgehalten, die folgende Gliederungspunkte haben: Beschreibung, Vorbedingungen, benötigte Daten, Handlungen, erwartete Reaktion des Systems, Nachbedingungen und Test log. Bei der Planung der Tests habe ich mich dazu entschieden, viel Aufwand für die Erstellung der Test-Cases aufzubringen, solang aufgrund dessen Zeit bei den einzelnen Durchführungen der Tests gespart werden kann. Dies ist durch Skripts zur Herstellung von bestimmten Systemzuständen und einer Detaillierten Beschreibung in den genannten Teilen des Test-Cases sehr gut gelungen.

2.5.4 Erkenntnisse

Dieser Abschnitt handelt weniger von Erkenntnissen über Tests, welche wir über die Vorlesung erlangt haben, sondern eher von denen, die wir über eigene Recherche, aber vor allem durch das Anwenden in unserem Projekt, erlangt haben.

Nachdem die Komponente zur Verarbeitung der PDF-Dateien fertiggestellt wurde, konnten diesbezüglich Tests durchgeführt werden. Nachdem die funktionalen Tests erfolgreich abgeschlossen wurden, wobei ausschließliche einzelne Aufträge mit wenigen Dateien testweise verarbeitet und zusammengefügt wurden, stellten wir uns die Frage, wieso es nicht funktionieren sollte, eine hohe Anzahl an Aufträgen zu verarbeiten. Wir dachten, dass es egal sein müsste, ob der Server drei oder 500 (Im Produktivbetrieb äußerst realistisch) Aufträge verarbeitet. Wir entschieden uns dennoch für einen Belastungstest, da wir einsahen, dass wir größtenteils sehr unerfahren waren und somit nicht nach Bauchgefühl agieren sollten. Tatsächlich ergab der Belastungstest, dass nur wenige Druckaufträge hintereinander verarbeitet werden konnten, da Datenbankverbindungen nicht geschlossen wurden und es somit zu einem Fehler der Datenbankschnittstelle kam. Dieses Ereignis war für uns alle sehr lehrreich und zeigte, dass breit diversifizierte Tests hinsichtlich ihrer Art sehr sinnvoll sind.

Des Weiteren verlies ich mich während der Tests auf die Aussagen der Entwickler über eine grobe Einschätzung der Fehlerwahrscheinlichkeit für einzelne Komponenten oder Skripte. Es stellte sich jedoch heraus, dass zwischen den Aussagen der Entwickler über die Fehlerwahrscheinlichkeit und der tatsächlichen aufgedeckten Fehler nur eine sehr geringe Korrelation herrscht. Dies bedeutet, dass Komponenten, welche von den Entwicklern als „sicher fehlerfrei“ bezeichnet wurde, dennoch viele und schwerwiegende Fehler aufweisen können. Dies liegt meiner Meinung nach nicht speziell an den Entwicklern unseres Teams, sondern daran, dass sie die Komponente selbst erstellt haben und somit eher einen technischen und weniger einen funktionalen Blick auf die Komponente haben.

2.6 Übergabe und Abnahme (PR)

Wir entschieden uns aufgrund der gegebenen Projektsituation verschiede Abnahmen durchzuführen. Wir führten mit Herrn Buschmann als Endnutzer der Webanwendung die beiden Akzeptanztests durch und ließen somit nur eine Komponente von ihm abnehmen. Die Hauptabnahme der Software und somit aller Komponenten und der entsprechenden Dokumentation fand mit Herr Prof. Dr. Naumann (Auftraggeber bzw. Themensteller) statt. Alle Abnahmen fanden in der Transition-Phase des Projektes statt.

Die beiden Akzeptanztests mit Herrn Buschmann, dem Endanwender der Webanwendung, verliefen sehr gut. Francesco Ryplewitz (Projektleiter), Felix Müller (Entwickler) und Paul Rogge (Tester) zeigten ihm die Funktionalitäten der Webanwendung und er hatte Gelegenheit, die Webanwendung selbst zu nutzen, Fragen zu Stellen und Anmerkungen zu äußern. Herr Buschmann war mit den Funktionen und der grafischen Oberfläche der Webanwendungen bei beiden Akzeptanztests sehr zufrieden und äußerte keine Verbesserungen. Die Organisation bezüglich der Abnahme mit Herrn Buschmann lief ebenfalls schnell und zielführend.

Nachdem die Webanwendung erfolgreich getestet und die Abnahme durch Herrn Buschmann erfolgreich war, entschieden wir uns dazu, das entwickelte System produktiv zu schalten. Nach Absprache mit Prof. Dr. Naumann nahm er sein bis dahin verwendetes System offline und wir stellten das neu entwickelte System und die darin enthaltene Webanwendung zur Nutzung zur Verfügung. Somit konnte ein reibungsloser Übergang zwischen den Systemen gewährleistet werden.

Die Endabnahme mit dem Themensteller bzw. Auftraggeber Herrn Prof. Dr. Naumann verlief ebenfalls sehr gut. Der Projektleiter traf sich mit ihm am 12.08.2020 und sie tauschten sich über die Projekterfolge aus. Nachdem sich Prof. Dr. Naumann einen Überblick über die Software und die Dokumentation gemacht hat, füllten beide das Abnahmeprotokoll aus. Seitens Prof. Naumann gab es

keine Beanstandungen. Er ist laut eigener Aussage äußerst zufrieden mit dem Endprodukt und der allgemeinen Zusammenarbeit mit dem Team.

2.7 LIZENZEN (JH)

Lizenzen: Bei diesem Projekt werden ausschließlich die Open-Source Programmiersprache Java in Verbindung mit dem Framework Spring verwendet. Auch die verwendete MariaDB Datenbank ist Open Source. Diese unterliegen der “GNU - General Public License” und der “GPL - linking exception“ welche öffentlich im Netz einsehbar sind. Somit sind keine Lizenzkosten zu entrichten, auch nicht zu einem Späteren Zeitpunkt.

- GNU: http://www.gnu.org/licenses/gpl-3.0.html
- GPL: https://de.wikipedia.org/wiki/GPL_linking_exception

2.8 DOKUMENTATION (JH)

Zur Dokumentation des Projektes wurde auf unterschiedliche Tools zurückgegriffen. Als Versionsverwaltungstool nutzten wir GitHub. Dieses gewährleistete einen einfachen Umgang der Versionierung unserer Dateien und des Quellcodes. Weiterer Vorteil war der Zugriff auf alle Dokumente durch jedes Teammitglied zu jeder Zeit. GitHub stellte sich als ein überaus nützliches Tool in diesem Projekt heraus, wodurch wir bei zukünftigen Projekten gerne darauf zurückgreifen werden. Mehr dazu in meiner Reflexion.

Zur Dokumentation unseres Java Quellcodes wurde JavaDoc benutzt. Javadoc ist ein Software-Dokumentationswerkzeug, das aus Java-Quelltexten automatisch HTML-Dokumentationsdateien erstellt. Dieses wurde zeitnah zu Beginn der Entwicklung eingesetzt, wodurch zum Schluss des Projektes eine Menge Zeit und Arbeit erspart blieb.

2.9 BESPRECHUNGSPROTOKOLLE

2.9.1 Meeting 10.12.2019

Teilnehmer: Teammitglieder, Hr. Naumann, Hr. Anke

Themen:

- jetzige Funktionalität des Projektes
- geplante Änderungen/Erweiterungen

Ziel: Service von und für Studierende

Jetziger Stand:

- Eigenentwicklung Herr Naumann (Prof. Fakultät Maschinenbau)
- Kosten bis dato ca. 15.000 €
- verarbeitbar: pdf, bis 20 Mb, ohne Passwortschutz
- Druckfunktionen: ein- od. doppelseitig, ein/zwei/vier Seiten pro Blatt

Grober Ablauf bisher

- Auswahl Druckaktion im Opal
- Eingang Druckauftrag im Mailpostfach plus Bestätigungsmail
- sammeln der Aufträge 24h
- Screening and automatische Verarbeitung durch versch. Batchfiles jeden Morgen 5 Uhr
    - Bündelung der Aufträge pro Person als Sammel-pdf
    - drehen
    - Seitenanpassung
    - Generierung Deckblatt pro Auftrag und Stapel (20 Blatt) mit Barcode …
- Ablage Festplatte
- automatischer Ausdruck und Lochen der Blätter in Hausdruckerei
- Transport zur Bibliothek
- Abholung der Unterlagen Servicetheke Bibliothek

Organisatorische Änderungen/Erweiterungen

- ges. System läuft auf einem Laptop → hohes Ausfall-/Störrisiko
- Lösung:
    - Outsourcing des kompletten Prozesses an Unidruckerei
    - System auf Server in Rechenzentrum der HTW spielen (Sicherung/Backup mögl.) Ansprechpartner: Fr. Dr. Lohse , Hr. Dominik (RZ Fakultät) für VM-Zugang Server

Technische Änderungen/Erweiterungen

- Optimierung und Erweiterung der Batchfiles
    - Probleme mit Sonderzeichen beheben
    - Probleme mit alten PDF-Versionen beheben

- sonstige Erweiterungen noch offen

2.9.2 Meeting 17.12.2019

Teilnehmer: Teammitglieder, Hr. Naumann, Fr. Franz (Unidruckerei), Hr. Buschmann (Unidruckerei)

Themen:

- Absprache mit Unidruckerei

Geplantes allgemein

Ab Beginn Sommersemester: Outsoursing zur Unidruckerei abgeschlossen

- Evaluierung durch Studierende zum jetzigen Druckservice
- "Werbung" für Druckservice - eventull Vorstellung durch Dozenten

Geplantes zum Prozessablauf

- Bei Auslösung Druckauftrag Zustimmung Weiterleitung Unidruckerei (Kontrollhäkchen)
- Drucke sind bereits eher abholbereit (Drucklauf eventuell mehrmals täglich)
- Stornierfkt bereitstellen (Zwischenablage der Aufträge in temp Verzeichnis)
- Erinnerungsmail bei Nichtabholung (Sperrdrohung)
- Sperren von Nutzern bei endgültiger Nichtabholung
- Interface für Unidruckerei
    - gewünscht: Kontrollfkt für Abholung/Nichtabholung (Häkchen setzen)

Sonstiges

- Anz. Mitarbeiter Unidruckerei: 3
- Ansprechnpartner Datenschutzbeauftragter: Prof. Westfeld

2.9.3 Meeting 13.01.2020

Teilnehmer Prof Naumann (Themensteller), Frau Franz und Herr Buschmann (Unidruckerei)

Neuerung Repository

Prof Naumann erhält Zugriff auf Repository (per Link) um auf dem Laufenden zu bleiben

Diskussion Visionsdokument

Prof Naumann stimmt Vision und Priorisierung der Features zu (Fokus liegt auf Dateitransfer und Layout) Herr Buschmann und Frau Franz stimmen ebenfalls zu

Antwort von Herrn Buschmann auf E-Mail der Analysten

1. Aufbewahrungsdauer noch offen
2. Matrikelnummer als Schlüssel zur eindeutigen Identifikation der Benutzer
3. Gestaltung des Preises noch offen (wahrscheinlich Preis pro Seite und "Mengenrabat" ab x Seiten); Wunsch: flexibel
4. Sperrung von Benutzern, die Dokumente nicht abholen: noch offen (laut Herrn Buschmann möglicherweise gar nicht benötigt, da Druck später Geld kostet) → Wunsch: nach x Tagen Erinnerungsmail

Opal Server

1. Ansprechpartner sind: Thomas Heider (Dipl-Inf, HTW Dresden, thomas.heider@htw-dresden.de) und Frank Richter (externe Firma)
2. Prof Naumann erklärt, dass Änderungen am Opal-Server einen langwierigen Prozess erfordern

Statistiken zur Anzahl Druckaufträge und Fehlerbeschreibung des momentanen Systems

1. im Moment wenige Studenten, aber viele/umfangreiche Aufträge pro Student
2. RIP-Fehler/PDF-Fehler
3. Fehlerbeschreibung NUP gesammelt per Mail an Francesco

System Unidruckerei

PC (Windows)

Systemumstellung zum 01.03.2020

Missverständnis mit Prof Naumnann geklärt: Umstellung zur Unidruckerei zum 31.03.2020 (macht Prof Naumann) und Umstellung auf unser System später

Probleme des aktuellen Systems (Liste von Prof Naumann)

1. Robustheit gegen Umlaute und Sonderzeichen in Dateinamen, außerdem führen zu lange Dateinamen zu Problemen, da zu viele Zeichen in der Kommandozeile stehen
2. Verarbeitungsreihenfolge: Viele Aufträge führen zu extremen Wartezeiten (im Moment LIFO 3 Minuten zwischen Abholungen)

3. Datenschutz: Prof Westfeld hat Vorschlag geliefert (kommt per Mail), Abstimmung mit Prof Naumann erfolgt noch
4. RIP-Fehler und NUP-Fehler siehe Statistiken zur Anzahl Druckaufträge und Fehlerbeschreibung des momentanen Systems

2.9.4 Meeting 31.07.2020

Anwesende: Teammitglieder Vollständig

- Tests (FM)
  - Probleme mit Mockopt Checks
  - 1 Test sollte bis nächste Woche stehen
  - Doku Stichpunkte verfassen
- Tests (PJ)
  - Unittests für Webapp funktionieren
  - Letzte Funktion wird noch getestet, bis nächste Woche fertig
  - Punkte in Testdoku aufnehmen
- (FM) Logfile Doku ist angefertigt
- (AZ) Usecase Modell aktualisiert, VM aufgeräumt, Anforderungsanalyse aktualisiert
- Entwickler Doku (FR) Webapp Stichpunkte an (JH)
- Abnahmetest verfassen und in Testdoku aufnehmen | im Projektbericht verlinken
- Betriebsdoku
  - Einige offene Punkte
  - Verteilung an Verantwortliche
- Projektbericht Aufgaben verteilt bis zum nächsten Meeting
- Nächstes Meeting festgelegt: 04.08.20 (15:00)

3 ERGEBNISSE

3.1 PROJEKTERGEBNIS (FR)

Abschließend lässt sich aussagen, dass sowohl das fachliche Ziel als auch das Ziel aus der Managementperspektive während der Projektarbeit erreicht wurden. Wir haben es stets geschafft, eine vernünftige und effiziente Zusammenarbeit zu organisieren.

Besonders im zweiten Semester dieses Projekts hat sich herausgestellt, dass ein wöchentliches Skype-Meeting sehr gut funktioniert, um stets einen Überblick über die Aufgabenverteilung zu haben. Des Weiteren zeigte sich durch diese Vorgehensweise auch, dass sehr schnell nachgesteuert werden konnte, falls Entwicklungen in die falsche Richtung verliefen.

Auch grundsätzlich lässt sich aussagen, dass jedes Teammitglied die Prinzipien des Unified Proccess und der agilen Softwareentwicklung verinnerlicht hat, was auch im späteren Verlauf des Studiums und besonders im späteren Berufsleben sehr nützlich sein wird. Außerdem konnten wir durch diese Arbeit auch das Arbeiten im Team sehr gut üben.

Wie bereits erwähnt wurde durch die Erreichung des Managementziels auch das fachliche Ziel erreicht, was bedeutet, dass wir bis auf die Opal-Schnittstelle das Gesamtsystem neu deplant und aufgesetzt wurde. Das neue Softwaresystem beinhaltet folgende Arbeitsschritte. Dateien und Druckeinstellungen werden fehlerfrei vom Opal an Postfach des Druckservices gesendet. Die Verarbeitung der PDF-Dokumente ist, wie in der Vision beschrieben, neu aufgesetzt und läuft auf einer VM eines Servers im Rechenzentrum der HTW. Eingegangene Druckaufträge werden ordnungsgemäß verarbeitet. Das Postfach des Druckservices wird auf neue Mails „abgeparsed“ und diese abgefangen. Es erfolgt eine Prüfung des Nutzers auf Sperrung, eine Prüfung der Datei auf Passwortschutz und die Durchführung der Graustufenumwandlung und Seitenkonvertierung. Um 1 Uhr nachts erfolgt laut umgesetzten Use Case eine Überprüfung auf Druckjobs, welche seit sieben Tagen nicht abgeholt wurden und bei Zutreffen wird eine Erinnerungsmail gesendet. Die Druckjoberzeugung wird um 5 Uhr morgens angestoßen. Gleichzeitig wird die Vorverarbeitung unterbrochen. Die Druckaufträge eines Studenten werden wie im Use Case beschrieben zu einem Druckjob zusammengefasst und ein Deckblatt pro Job erstellt.

Eine Webapplikation für die Mitarbeiter der Unidruckerei auf Basis von Spring ist in Nutzung. Diese beinhaltet alle gewünschten Funktionalitäten und ist durch eine Login Maske geschützt. Die Webapp beinhaltet folgende Features, welche alle durch einen Use Case abgebildet sind: Download der Druckjobs vom Server, Darstellung der Druckübersicht, Sperrung von Nutzern, Bestätigung eines Druckes und Bestätigung der Abholung des Druckjobs.

Nicht umgesetzt wurde die Möglichkeit, andere Zugänge zu schaffen, um Druckaufträge abzusenden. Dies hätte beispielsweise zusätzlich rein per Mail oder über ein Laufwerk umgesetzt werden können. Grund für die Nichtumsetzung ist das begrenzte Zeitbudget. Außerdem wäre aus Nutzersicht der Studenten auch eine Implementierung von Funktionen zur Informationseinsicht und zur Stornierung eines Druckauftrages möglich. Eine Erweiterung der Webapp als Schnittstelle für die Studenten wäre eine gute Lösung gewesen. Auch hier ist der Grund für die Nichtumsetzung der Zeitmangel.

Während der gesamten Bearbeitung des Projektes war uns ein gut durchdachtes Qualitätsmanagement ebenfalls äußerst wichtig, was sich am Endprodukt auch zeigt.

Zum einen haben wir es geschafft für unser Projekt eine konsistente, logische und nachvollziehbare Dokumentation zu erstellen. Dies ermöglicht zukünftigen Administratoren eine vereinfachte Wartung und außerdem zukünftigen Entwicklern eine vereinfachte Weiterentwicklung des Systems.

Zum anderen hat auch unsere eigentliche Software aus unserer Sicht einige Qualitätsmerkmale. So sind die einzelnen Komponenten nur lose miteinander verbunden, wodurch diese einfacher austauschbar sowie wartbar sind. Als zentrale Komponente dient unsere Persistenz Schicht (MariaDB). Mit dieser sind dann die Webanwendung und auch das eigentliche Hintergrundsystem als jeweils separate Komponenten verbunden. Diese zwei benannten Komponenten stellen die Logik dar. Das Hintergrundsystem, welches der PDF-Verarbeitung dient, besteht ebenfalls aus nur losen verbundenen Klassen, wodurch auch hier keine große Abhängigkeit untereinander entsteht. Die eigentliche View stellt uns das Opal-System bereit (zur Druckjoberstellung) sowie das Webframework Thymeleaf (für die Webanwendung).

Durch die klare Trennung der Komponenten und Schichten voneinander ist es uns gelungen eine einfach zu wartende, strukturierte und effiziente Software zu entwickeln. Qualitätsmanagement hat uns in diesem Punkt sehr viel gebracht, da wir ohne dieses wohl eine wesentlich schlecht strukturiertere Software entwickelt hätten.

Auch in Sachen allgemeiner Zuverlässigkeit ist unsere Software gut aufgestellt. Zum einen haben wir ein sehr umfängliches Exceptionhandling, wodurch im Fehlerfall kein kompletter Systemstart notwendig ist, was bei der ursprünglichen Software noch so war. Zum anderen haben wir durch die Bereitstellung einer VM im Rechenzentrum der HTW, auf der das gesamte System gehostet wird, eine sehr geringe Ausfallwahrscheinlichkeit.

Trotz dessen, dass zwei Use Cases, wie in der Implementierung beschrieben, nicht umgesetzt werden konnten und die Opal Schnittstelle zur Auftragserteilung nicht neu erstellt wurden konnten, können wir trotzdem sagen, dass unser Projekt erfolgreich abgeschlossen wurde. Die neue Software ist bereits im Betrieb, läuft wesentlich zuverlässiger als vorher und hat einen wesentlich höheren Funktionsumfang als das Ursprungssystem.

3.2 REFLEXIONEN

3.2.1 Project Manager: Francesco Ryplewitz

Das Softwareprojekt, welches wir gemeinsam in SE1 und SE2 bearbeitet haben, ist bis dato das umfangreichste Softwareprojekt, an dem ich bisher gearbeitet habe. Ich konnte sehr viel aus diesem Projekt mitnehmen.

Vor allem das Zusammenwirken verschiedener Rollen, wie z.B. zwischen dem Architekten und den Entwicklern, war äußerst interessant zu beobachten. Dadurch konnte ich einen guten Einblick in die praktische Softwareentwicklung gewinnen, wodurch ich verstand, dass nicht jeder alles machen kann und man manche Aufgaben einfach mal abgeben muss, um effizient voran zu kommen.

Auch die Anwendung der Prinzipien des Unified Process und der damit einhergehenden agilen Entwicklung war äußerst interessant und werden vermutlich bei vielen weiteren zukünftigen Softwareprojekten hilfreich sein.

Einer meiner größten Learnings während dieses Projektes war, dass man sich niemals auf Annahmen verlassen sollte. So haben wir bis in das zweite Semester des Projektes hinein angenommen, dass eine Anpassung der Opal-Schnittstelle zu unserem System ohne weiteres realisierbar ist. Auf diese Annahme haben wir unsere Entwicklung gestützt, was im Nachhinein betrachtet sehr naiv war. Irgendwann haben wir unsere Fehleinschätzung realisiert, wodurch wir unsere Software noch einmal etwas umstrukturieren mussten. Das war definitiv Arbeit, welche man sich hätte sparen können.

Alles im Allem kann ich jedoch auf eine spannende Zeit zurückblicken, in der ich sehr viel gelernt habe. Besonders weil ich das Gelernte auch im weiteren Verlauf des Studiums und im späteren Berufsleben weiterhin anwenden kann, war das Projekt für mich sehr erfolgreich und nützlich.

3.2.2 Analyst/Tester: Paul Rogge

Das Projekt „Opal-Druckauftrag“ war mein erstes komplexeres software-Projekt. Daher konnte ich die Gelegenheit nutzen, um mit einem Team gemeinsam über einen relativ langen Zeitraum zu arbeiten und diesbezüglich Erfahrungen zu sammeln.

Es stellte sich schnell heraus, dass ein solches Projekt eine gute Organisation, Dokumentation und Planung bedarf. Dank unserer guten und kompetenzbezogenen Rollenverteilung in Verbindung mit guter Kommunikation stießen wir bei der Zusammenarbeit auf keine Probleme, worauf ich sehr stolz bin.

Auch wenn die Dokumentation einen sehr großen Aufwand darstellte, bin ich mir sicher, dass wir ohne diese gegen Ende des Projektes auf organisatorische und inhaltliche Probleme gestoßen wären.

Generell wurde mir bewusst, dass Planung und Modellierung ein wichtiger und sinnvoller Bestandteil eines Projektes sind. Modelle stellen einen Sachverhalt vereinfacht dar und erleichtern die Arbeit meist ungemein. Zudem vereinfachen sie die Kommunikation, was bei komplizierten Sachverhalten zu einer effektiveren Arbeit im Team führte.

Zusätzlich bin ich froh, dass wir mit der Zeit als kritische Nebenbedingung sehr gut umgehen konnten. Besonders zu Beginn des Projektes war unklar, wie gut wir im Projekt voranschreiten können. Ein guter Zeitplan für das ganze Projekt aber auch für den Teilbereich der Tests war sehr hilfreich. Ich bin sehr froh, dass ich mir kontinuierlich selbst gestellte Aufgaben lösen konnte, ohne temporär sehr viel bzw. sehr wenig mit den Aufgaben des Testers beschäftigt zu sein. Auch kurz vor Abschluss des Projektes waren nur Kleinigkeiten zu erledigen, was für einen strukturierten und pünktlichen Abschluss des Projektes sorgte.

Auch wenn das System nicht alle aufgenommenen Anforderungen erfüllt, konnten wir in der gegebenen Zeit ein gutes Resultat erzielen und hoffen, dass der Auftraggeber mit der entstandenen Software zufrieden ist. Die nicht umgesetzten Funktionalitäten sind nicht für das erfolgreiche Nutzen des Systems relevant, sondern stellen kleine Erweiterungen dar.

Ich bin auch sehr stolz, dass die Teamatmosphäre während des kompletten Projekts sehr gut war. Meinungsverschiedenheiten konnten schnell, unkompliziert und ohne Streit geklärt werden.

Bei zukünftigen Projekten würde ich als Tester den Ansatz des testbasierten Entwickelns bevorzugen. Da ich während des Projektes das Gefühl hatte, dass den Entwicklern nicht immer alle Anforderungen im Detail bekannt waren, führte dies zu häufigem Anpassen der Komponenten. Da die Tests direkt auf den Anforderungen aufbauen kann ich mir so vorstellen, dass die daraus entstehende Softwarekomponente eher den tatsächlichen Anforderungen und nicht den angenommenen Anforderungen der Entwickler entsprechen würde. Dies lag nicht speziell an den Entwicklern dieses Teams, sondern eher an der generellen Vorgehensweise.

3.2.3 Architect/Developer: Felix Müller

Zunächst sind die gute Teamarbeit und Organisation zu erwähnen. Durch vorausschauende Planung, wöchentliche Meetings, klare Aufgabenzuweisung und grundsätzliches Einhalten von zeitlichen Limits konnten wir unsere Anstrengungen vollständig auf die fachlichen und technischen Aspekte des Softwareprojekts fokussieren. Der wöchentliche Rhythmus der Meetings half dabei „am Ball zu bleiben“ und zeigte deutlich Fortschritte und Defizite auf. Pair Programming hat uns dabei geholfen Teams zu bilden, die verschiedene Teilsysteme, beziehungsweise Komponenten parallel entwickeln.

Wir konnten dadurch sicherstellen, dass das Hintergrundsystem und die Web-Applikation parallel entstehen und nicht ein Systemteil unfertig ausgeliefert werden muss.

Im nächsten Softwareprojekt würde ich gerne Test Driven Development ausprobieren, da ich der Meinung bin, dass sich durch diese Praktik die Vorteile der agilen Softwareentwicklung noch stärker nutzen lassen. In diesem Projekt war es uns nicht möglich Test Driven Development, da unser System zum einen nur unzureichend automatisiert testbar ist, zum anderen durch Fluktuation von Teammitgliedern die Rolle des Testers längere Zeit zum Beginn dieses Semesters unbesetzt war.

Aus dem Praktikum habe ich PlantUML als lohnende Technik mitgenommen, deren Einsatz sich für uns aber nicht mehr gelohnt hätte.

Neu für mich war zudem die Arbeit mit SpringBoot als Framework für unsere Web-Applikation. Die Einarbeitung hat (etwas) Zeit und Aufwand gekostet, der sich aber meines Erachtens sowohl für das Voranschreiten des Projekts als auch für mich gelohnt hat.

Die Kommunikation mit einigen Stakeholdern lief zwischenzeitlich etwas stockend, was dazu führte, dass System nicht so entwerfen und implementieren konnten, wie ursprünglich angedacht.

3.2.4 Developer: Annabelle Zimmer

Zuerst ist im Allgemeinen zu erwähnen, dass ich durch die Projektbearbeitung unter Leitfaden des OpenUP viele grundlegende Kenntnisse und Fähigkeiten zur Durchführung eines Softwareprojektes mitnehmen konnte. Besonders die Anwendung der agilen Entwicklungsmethode hat die Arbeit erleichtert. Durch die Teilergebnisse ist man mehr auf das wesentlich fixiert und hat Teilerfolge, die für einen Motivationsschub sorgen. Ich verbesserte außerdem meine Herangehensweise zum Implementieren von Lösungen zu gestellten Problemen sowie mein Verständnis in Bezug zu Aufbau und Funktionsweise verteilter Systeme.

Durch das Projekt konnte ich meine Kenntnisse in der Java Programmierung verbessern. Vor allem in den Themen Dateiarbeit, Datenbankabfragen und Verarbeitung von Shell Skripten habe ich viel dazugelernt. Auch mein Wissen in der Programmierung von Shell Skripten und der Arbeit mit der Linux-Kommandozeile habe ich auffrischen können. Zudem nutzten wir die Versionsverwaltung GitHub. Hierbei sollte ich in Zukunft versuchen, noch mehr Funktionalitäten, wie das zusammenfügen mehrerer Versionen, anzuwenden. Auch habe ich Javadoc als praktisches Dokumentationswerkzeug kennengelernt und die Erstellung von UML-Diagrammen mittels Visual Paradigm.

Im Team herrschte von Anfang an ein freundlicher und respektvoller Umgang. Auch die Zusammenarbeit unter uns Entwicklern war von viel Hilfsbereitschaft und einer guten Kommunikation geprägt. Alle Mitglieder haben sich engagiert ihren Aufgaben gestellt. Auch habe ich gelernt, dass die

Koordination in größeren Teams nicht einfach ist. Durch unsere regelmäßigen Meetings, die Bildung von Unterteam und der gemeinsamen Aufgabenverteilung ist es uns recht gut gelungen diese Hürde zu meistern.

In folgenden Softwareprojekten werde ich versuchen strukturierter an die Implementierung heranzugehen. Dazu gehört eine bessere Vorabanalyse benötigter Klassen und Funktionalitäten, um eine gute Codestruktur zu gewährleisten sowie Fehlerfallbehandlungen und alternative Abläufe frühzeitig zu erkennen. So kann viel Zeit eingespart werden. Ferner werde ich mir vornehmen, Arbeitsergebnisse öfter zu pushen, um eine bessere Kontrolle des Arbeitsvorschrittes zu ermöglich.

3.2.5 Developer: Paul Jannasch

Da dies mein erstes größeres Softwareprojekt im Team war, habe ich natürlich einiges mitgenommen und gelernt. Da ich selbst als Software Developer tätig war, habe ich selbstverständlich eine Menge neuer Programmierfähigkeiten in der Sprache Java und in der Bash erlernt. Neu war für mich dabei vor allem, die Menge an verschiedenen Teilsystemen (Komponenten des Systems die bei der Serveranwendung den Java-Klassen entsprechen) untereinander zu verknüpfen, kompatibel zueinander zu gestalten und lauffähig zu bekommen. Die Serveranwendung zur Verarbeitung arbeitet dabei mit Java-Klassen für die Komponenten, Shell-Skripten für die Verarbeitung der PDF-Dateien, welche dann wiederum Linux-Tools nutzen, sowie eine Datenbankanbindung über JDBC. All diese Teilsysteme, zu einem funktionierenden Ganzen zu kombinieren, war in diesem Umfang definitiv eine neue Erfahrung. Auch eine große Rolle im Lernprozess hat natürlich die Teamarbeit gespielt, wozu ich nur sagen kann, dass diese hervorragend funktioniert hat, wobei die Wichtigkeit des Austausches nicht aussenvor gelassen werden sollte. Wir haben dafür vor allem GitHub verwendet, wobei mir zwar vorher schon der Nutzen der Plattform, durch das Praktikum bekannt war, ist mir erst im Laufe des Projekts klar geworden, wie praktisch die Plattform doch für Teams ist. Im Großen und Ganzen handelte es sich zwar um eine größere Anlegung des Praktikums, aber dennoch ist mir jetzt klar, wieso viele Firmen auf Git setzen. Der dritte interessante Punkt ist die Dokumentation, wobei man ehrlich sagen muss, dass diese nicht zu allen Zeiten des Projekts auf dem aktuellen und präzisen ausformulierten Stand war, auf dem sie vielleicht hätte sein sollen. Dennoch haben oftmals schon kleine Erklärungen oder auch nur Textdateien mit Stichpunkten dabei geholfen, dass Verständnis und den Überblick zu bewahren. Ich kann mir sogar vorstellen, dass sich diese Methode im Projektverlauf manchmal eher rentiert, als eine lang ausformulierte Dokumentation.

Stolz ist man als Entwickler vor allem auf die neuen Kenntnisse und Programmierfähigkeiten, die man im Laufe eines Projektes erwirbt. Dabei handelt es sich für diesen Fall für mich vor allem um die Arbeit mit den verschiedenen Tools unter Linux für die Verarbeitung der PDF-Dateien, da im Sektor Java-

Programmierung und Datenbanken wenig neues Wissen erlangt wurde. Wichtig ist aber auch die Technik der Softwaredokumentation, hierbei vor allem die Arbeit mit Javadoc neben den bereits bekannten klassischen Kommentaren in den Programmen. Javadoc als Tool war mir davor nur als Name bekannt und ich bin begeistert von dessen Fähigkeiten, wenn man sich nur an eine simple Syntax hält. Ebenfalls bin ich sehr zufrieden mit der Architektur des Systems, da diese wie ich erfahren habe, vor allem zu Projektende oftmals vernachlässigt wird, nach dem Motto, Hauptsache es funktioniert. Dies kam bei uns zwar auch hin und wieder vor, aber nach Reflexion des Codes konnte diese immer wieder ordentlich angepasst werden, um somit ein schlüssiges Gesamtsystem zu bauen.

Sehr gut funktioniert hat bei uns die Arbeit im Team, sowie die Absprache untereinander. Fundament für die erfolgreiche Arbeit war eine ausgiebige Planung, mittels der wöchentlich aktualisierten Iterationspläne. Auch wenn die verschiedenen Iterationen unterschiedliche Längen hatten, die oftmals eine Woche überschritten, so war durch die wöchentliche Gliederung und die Treffen an fast jedem Montag im Verlauf des gesamten Semesters, eine klare Struktur erkennbar. Sowohl die Meetings als auch die Pläne haben außerdem immens die Übersichtlichkeit erleichtert und beim Durchblick des gesamten Projektverlaufs geholfen, somit konnte ein jeder sich über die Arbeit anderer Teammitglieder informieren und hatte einen Überblick über den aktuellen Stand der Dinge. Die zeitliche Einbindung der regelmäßigen Meetings war ebenfalls kaum problematisch und es war fast immer jedes Teammitglied anwesend. Außerdem gab es auch oftmals kleinere Besprechungsgruppen innerhalb des Teams, was für mich vor allem die Absprachen mit Felix Müller unserem Architekten, Annabelle Zimmer unserer zweiten Software Developerin und Paul Rogge unserem Tester bedeuteten, wobei auch hier alles reibungslos funktionierte. Generell wurde sich innerhalb des Teams weitestgehend an Abgabezeiten gehalten, was eine Zusammenarbeit sehr stark erleichtert hat.

Für zukünftige Softwareprojekte möchte ich mir eher die allgemeinen Probleme, die bei fast jedem Projekt auftauchen mitnehmen. Bei uns im Team gab es kaum unvorhergesehenen Schwierigkeiten, die man durch bessere Planung oder ähnliches verhindern hätte können. Es wird wie bereits besprochen wohl auch nie die perfekte Planung geben, daher hätten wir auch durch bessere Absprache mit dem Kunden das ein oder andere Problem vermeiden können, beispielsweise wurde für unser Projekt speziell die Verwendung eines Barcodes für die Preisberechnung am Ende gar nicht mehr benötigt, was mich innerhalb des Projektes viel Zeit gekostet hat. Solche Dinge könnten vermieden werden, gehören aber meiner Meinung nach zu vorhersehbaren Komplikationen. Somit würde ich unser Projekt als sehr erfolgreich beschreiben und dies so mitnehmen.

3.3 Abnahme-Protokoll

Abnahmeprotokoll

über die Software des Opal-Druckauftrags im Rahmen der Lehrveranstaltung Software Engineering der HTW Dresden

Auftraggeber (vertreten durch): **Prof. Dr.-Ing. Gunther Naumann**

Auftragnehmer (vertreten durch): **Francesco Ryplewitz**

Tag der Abnahme: 12.8.2020

Die Ausführung der abgenommenen Leistungen wurde begonnen am 5.12.2019

und beendet am 12.08.2020

Gewährleistung:

Es existiert keine Frist für die Gewährleistung der abgenommenen Software.

Bei der heute erfolgten gemeinschaftlichen Abnahme wurde festgestellt, dass sich die Leistung bis auf die nachstehend bezeichneten Schäden oder Mängel im zu erstellenden Zustand befand. Es bleibt jedoch ausdrücklich vorbehalten, nicht erkannte Mängel zu jedem späteren Zeitpunkt geltend zu machen.

Mängel: **Behebungsfrist:**

Bemerkungen und Vorbehalte:

weitere noch zu erbringende Leistungen:

Dresden, 12.8.2020
(Ort, Datum)

für den Auftragnehmer:

(Unterschrift)

für den Auftraggeber:

J. Naumann
(Unterschrift)