Digitale Schaltungssimulation im Browser

Projektdokumentation

Version 5.1

Niels Andrä
Maximilian Engelhardt
Dario Götze
Jonas Knüpper
Paul Lindner
Dorian Müller
Markus Seeber
Alexander Zenkner

Autor des Dokuments	Jonas Knüpper	Erstellt am	11.04.2017
Projektname	Digitale Schaltungssimulation im Browser		
Seitenanzahl	34	© 2017	

Historie der Dokumentversionen

Version	Datum	Autor	Änderungsgrund / Bemerkungen
1.0	25.04.2017	Jknu	Ersterstellung
2.0	02.05.2017	Jknu	Vorläufige Version
3.0	03.05.2017	Jknu	Abgabe FG System- und Software-Engineering und Abschluss der Planungsphase
4.0	31.05.2017	Jknu	Hinzufügen Durchführungsphase
4.1	02.06.2017	Jknu	Zwischenversion Korrekturlesen
5.0	06.06.2017	Jknu	Abgabeversion FG System- und Software-Engineering und Abschluss der Durchführungsphase
5.1	02.07.2017	Jku	Vorläufige Version zur Korrektur

Inhaltsverzeichnis

Historie der Dokumentversionen 2
Inhaltsverzeichnis 3
1 Einleitung 5
2 Projektdefinition 6
2.1 Allgemeines 6
2.2 Ist-Analyse 6
2.3 Soll-Konzept 6
3 Projektplanung 7
3.1 Anforderungen an das Projekt 7
3.2 Ressourcenplanung 7
3.2.1 Personalressourcen 7
3.2.2 Sachressourcen 8
3.3 Interne Organisation 8
3.3.1 Rollenverteilung 8
3.3.2 Kommunikation 8
3.4 Vorgehen innerhalb des Projektes 9
3.4.1 Vorgehensmodell 9
3.4.2 Projektstrukturplan 9
3.4.3 Aufwandsabschätzung 9
3.4.4 Umsetzung in BCS 10
3.5 Projektziele 12
3.6 Terminplanung 12
3.7 Machbarkeitsanalyse 13
4 Projektdurchführung 16
4.1 Refactoring-Phase 16
4.1.1 SimcirIF 16
4.1.2 Klassen-Auslagerung 16
4.2 Designphase 16
4.2.1 MockUp der Benutzeroberfläche 16
4.2.2 Konzeption der Funktionsimplementierung 17
4.2.3 BEAST-Datenmodell 19
4.3 Realisierungsphase 21
4.3.1 Anlegen eines Webprojektes 21
4.3.2 Erstellen der Benutzeroberfläche 21
4.3.3 Integration der simcir.js Funktionalität 22
4.3.4 Implementierung der Funktionen 22
5 Projektabschluss 24
5.1 Allgemeine Auswertung 24
5.2 Auswertung der umgesetzten Funktionalitäten 24
5.3 Auswertung der Personalressourcen 26
5.4 Bugreview 28
5.5 Fazit 29
5.5.1 Bewertung des Vorgehens 29

5.5.2 Bewertung des Projekterfolges 30
6 Glossar 31
7 Anhang / Ressourcen 32
8 Abbildungsverzeichnis 34

1 Einleitung

Das Softwareprojekt ist eine Lehrveranstaltung des Fachgebiets System- und Software-Engineering der TU Ilmenau. Hierbei wählen Studierende aus einem Themen-Pool eine ihnen zusagende Thematik aus und bearbeiten diese in Form eines Projektes im Laufe eines Semesters.

Die Projektgruppe entschied sich aus dem Themen-Pool für das Projekt „Digitale Schaltungssimulation im Browser“.

Ziel des Projektes ist es, dem Fachgebiet Integrierte Kommunikationssysteme ein Tool zur Verfügung zu stellen, welches es Studierenden ermöglicht digitale Schaltungen für ihre Praktika der Technischen Informatik im Browser zu entwerfen, zu simulieren und zu speichern um sich besser auf die Versuchsdurchführung vorbereiten zu können.

Darüber hinaus geht es im Rahmen des Softwareprojektes aber auch darum, dass Studierende mit den verschiedensten Aspekten des Projektmanagements und der Softwareentwicklung unter möglichst realen Bedingungen vertraut werden und Erfahrungen sammeln können. Die Studierenden sollen alle grundlegenden Phasen eines Projektes von der Planung über die Konzeption, Implementierung, Validierung und Auslieferung gemeinsam durchlaufen und nicht nur das gewünschte Produkt mit modernen Methoden der Softwareentwicklung erstellen, sondern auch den Projektablauf fachgerecht planen, kontrollieren und dokumentieren, sowie vor kritischem Publikum zu präsentieren und zu verteidigen.

Des Weiteren sollen die Studierenden beweisen, dass sie unter Zeitdruck Entscheidungen treffen und sowohl die theoretischen als auch praktischen Probleme meistern können, welche auch in einem realen Softwareprojekt auftreten können. Dabei können die Projektmitarbeiter mit Problemen technischer, als auch menschlicher Natur konfrontiert werden.

2 Projektdefinition

2.1 Allgemeines

Blockschaltbilder werden zur Darstellung von Wechselwirkungen zwischen verschiedenen Bauteilen oder Funktionseinheiten genutzt. Ein Einsatzgebiet stellt die Darstellungen von digitalen Schaltungen dar. Hierbei werden unterschiedliche Komponenten wie AND/OR-Verknüpfungen oder Negierungen durch ihre Ein- und Ausgänge miteinander verbunden. Als Ergebnis können komplexe Schaltungen entstehen, welche zum Teil auch wieder als Definition eines neuen Bauteils verwendet werden. So kann beispielsweise anhand des dazugehörigen Blockschaltbildes die Funktionsweisen und Unterschiede verschiedener FlipFlop-Typen verdeutlicht werden.

2.2 Ist-Analyse

Aktuell werden die Praktikumsvorbereitungen für die Praktika der Technischen Informatik per Hand ausgearbeitet. Hierzu gehören auch die Schaltungsstrukturen, welche für die Bearbeitung des Praktikums notwendig sind. Hierbei sollen Studierende anhand von Problemstellungen Automaten entwerfen. Anhand der Automatengleichungen kann dann die Schaltung in Form eines Blockschaltbildes erstellt werden.

Innerhalb von GOLDi-Labs existiert bereits ein Modul namens GIFT (Graphical interactive FSM Tool), welches es ermöglicht Automatengraphen digital darzustellen und Gleichungen anzeigen zu lassen.

Neben einer manuellen Steuerung ist es auch möglich durch Importieren von Gleichungen die verschiedenen virtuellen und realen Modelle in GOLDi-Labs zu bedienen.

Außerhalb von GOLDi-Labs existiert bereits ein Projekt namens simcir.js. Mit dieser Webanwendung ist es möglich, Schaltungen mittels eines vorgefertigten Sets aus Komponenten zu erzeugen, welche dann auch simuliert wird.

2.3 Soll-Konzept

Als Ergebnis des Softwareprojektes soll ein Webmodul entstehen, welches es ermöglicht Blockschaltbilder anzulegen, zu bearbeiten und zu simulieren. Dabei soll der Benutzer auf ein Set von Grundkomponenten zurückgreifen können, welche real auch in den Praktika der Technischen Informatik an der TU Ilmenau zur Verfügung stehen.

Das Blockschaltbild soll während der Bearbeitung simuliert werden, wobei eine Simulation der farblichen Markierung von logisch eins und logisch null belegten Ein-Ausgängen bzw. Leitungen entspricht. Komponenten, welche eine Simulationsmöglichkeit beinhalten, wie zum Beispiel LEDs (leuchten, wenn Eingang mit logisch eins belegt), sollen ebenfalls Simulationsfunktionen bereitstellen.

Die Auswahl der Komponenten zur Anordnung auf einer Arbeitsfläche soll durch eine Art Toolbox in Form eines Navigationsbaumes erfolgen. Zusätzlich zur freien Anordnung sind auch weitere Funktionen zur Anordnung der Komponenten notwendig. Hierzu gehört die Möglichkeit, eine Komponente zu rotieren.

Um aus angeordneten Komponenten ein Blockschaltbild erstellen zu können, ist es wichtig, dass Ein- und Ausgänge von Komponenten miteinander verbunden werden können. Dies soll entweder direkt geschehen, oder über Knoten, um ein übersichtliches Blockschaltdiagramm zu gewährleisten.

Im Sinne der Übersichtlichkeit, soll es auch möglich sein, mehrere Komponenten zusammenzufassen und als eine Komponente abzuspeichern. Hierbei soll die Arbeitsfläche einer Tab-Struktur unterliegen,

um auch eine parallele Bearbeitung zu ermöglichen. Die Tabs können auch zur Ansicht von zusammengesetzten Komponenten genutzt werden. So kann die Struktur hinter einer Komponente in einem weiteren Tab angezeigt werden (zum Beispiel FlipFlop – Struktur).

Komponenten sollen einer Bibliotheken-Struktur untergeordnet sein, sodass die Möglichkeit existiert, neue Komponenten dem Projekt durch den Import einer Bibliothek hinzuzufügen. Außerdem sollen neu-angelegte Komponenten in selbst-erstellte Bibliotheken exportiert werden können.

Um das Projekt mit seinem Blockschaltbild und den Bibliotheken portabel zu machen, soll auch eine Möglichkeit geschaffen werden, ein Projekt lokal zu speichern.

Für die Umsetzung bietet sich eine Erweiterung oder eine Änderung des bereits verfügbaren simcir.js-Projektes an. Inwieweit simcir.js verwendet werden kann, oder angepasst bzw. erweitert werden muss, stellt ein Ergebnis der Entwurfskonzeption dar.

3 Projektplanung

3.1 Anforderungen an das Projekt

Um das Projekt ordnungsgemäß und regelgerecht umsetzen zu können, müssen verschiedene Anforderungen erfüllt sein. Hierzu wurde in Zusammenarbeit mit den Fachbetreuern aus dem Fachgebiet Integrierte Kommunikationssysteme der TU Ilmenau ein Lastenheft ausgearbeitet.

Bezüglich der Programmiersprachen und benutzten Entwicklungsumgebungen, werden als Empfehlungen TypeScript im Zusammenhang mit JavaScript und das Tool Webstorm festgelegt. Sowohl Designvorgaben, als auch Regelungen zur Strukturierung und Gestaltung des Quellcodes werden nicht gegeben. Jedoch ist eine lesbare und wiederverwendbare Code-Basis erwünscht.

Das Webmodul soll autark funktionieren, jedoch auch Schnittstellen zur Integration in bereits bestehende Webseiten besitzen. Des Weiteren sollen Schnittstellen zum Setzen und Lesen von Ein- und Ausgängen geschaffen werden. Um die online ausgearbeiteten Blockschaltdiagramme wiederverwenden zu können, ist eine wichtige Anforderung der Im- und Export von Projekten. Die Dateien sollen in einem lesbaren Format gespeichert werden, damit der Benutzer die Daten auch ohne Verwendung von BEAST lesen kann.

3.2 Ressourcenplanung

Die für das Projekt benötigten Ressourcen unterteilen sich in Personal- und Sachressourcen.

3.2.1 Personalressourcen

Die Personalressourcen ergeben sich rechnerisch aus den für das Softwareprojekt durch die Prüfungsordnung festgelegten acht Leistungspunkten. Ein Leistungspunkt (Credit) entspricht hierbei einem Zeitaufwand von 30 Zeitstunden. Multipliziert man dies mit der Anzahl der Projektmitarbeiter, erhält man einen Gesamtaufwand von 1.920 Stunden.

Da alle Mitarbeiter, welche an diesem Projekt arbeiten, ausschließlich Studenten der TU Ilmenau sind, erfolgt die Projektumsetzung parallel zu weiteren universitären Lehrveranstaltungen. Daher ist es nicht möglich feste tägliche Arbeitszeiten festzulegen. Anstelle dessen, wurde jedem Projektmitarbeiter eine wöchentliche Arbeitszeit von 18,5 Stunden zugewiesen.

3.2.2 Sachressourcen

Für die Durchführung des Projektes wird vom betreuenden Fachgebiet Integrierte Kommunikationstechnik der TU Ilmenau ein Webspace, sowie ein SVN Repository gestellt. Um an dem Webmodul entwickeln zu können, nutzen jeder Projektmitarbeiter einen privaten Entwicklungs-PC. Die Entwicklungs-PCs verfügen über die Entwicklungsumgebung *Webstorm* von *Jetbrains*, das SVN-Tool *Tortoise*, die Kommunikationssoftware *Slack* und das *Office-365* Paket von Microsoft zur Erstellung von Pflichtdokumenten, Dokumentationen und Präsentationen. Zusätzlich wird für die Erstellung von Diagrammen auf einigen Entwicklungs-PCs *Microsoft Visio* verwendet. Die Erstellung von MockUps zur Darstellung und Konzeption von Benutzeroberflächen erfolgt mit dem Tool *Balsamiq*. Für das Testmanagement, sowie das manuelle und automatisierte Testen wird *testOffice* von Spirit-Testing Software & Services GmbH verwendet.

Um das Projekt planen und organisieren zu können, wird von dem Unternehmen projektron ein Server mit der installierten Projektmanagement-Software *projektron BCS* dem Projektteam zur Verfügung gestellt.

Für die Umsetzung wurde den Projektmitarbeitern der Zugang zu einem Besprechungsraum innerhalb der Räumlichkeiten der TU Ilmenau gewährt. Des Weiteren wurde durch die Projektgruppe ein Seminarraum für wöchentliche Treffen reserviert.

3.3 Interne Organisation

Für die erfolgreiche Umsetzung des Projektes ist die Festlegung einer Organisation innerhalb der Projektgruppe erforderlich. Neben der Festlegung von unterschiedlichen Rollen, spielt hierbei auch das Planen von regelmäßigen Gruppentreffen ein Aspekt.

3.3.1 Rollenverteilung

Die Projektgruppe setzt sich aus zwei Projektleitern und sechs Entwicklern zusammen. Dabei teilen sich die Projektleiter verschiedene Administrations- und Controlling-Aufgaben unter sich auf. Die restlichen Entwickler übernehmen je nach Projektphase unterschiedliche Aufgaben. Hierzu gehören neben der Planung von Teilaufgaben die Implementierung, sowie die Dokumentation und das Testen von Funktionalität.

Da die Entwickler ihre Aufgaben vollständig untereinander aufteilen, wird auf eine exakte Aufgabenverteilung in diesem Dokument verzichtet. Unter den Projektleitern erfolgt eine Aufteilung der Leitungsaufgaben. So wird ein Projektleiter sich um die technische Leitung kümmern, während der andere Projektleiter die Projektorganisation übernimmt.

3.3.2 Kommunikation

Zur internen Kommunikation der Gruppe wurde zu Beginn des Projektes ein wöchentlicher Termin festgelegt, bei dem Fragen der Projektmitarbeiter geklärt werden können. Zu diesen Terminen geben die Projektmitglieder Aufschluss darüber, welche Aufgaben sie in der vergangenen Woche bearbeitet haben. Des Weiteren geben sie einen kurzen Ausblick über die noch anstehenden Tätigkeiten.

Für die Kommunikation mit den Fachbetreuern wurde ebenfalls ein wöchentlicher Termin festgelegt. Innerhalb dieses Termins werden Fragen geklärt, welche eine Entscheidung der Fachbetreuer, die in diesem Projekt als Kunden fungieren, benötigen.

Ein weiterer wichtiger Kommunikationsweg ist die Kommentarfunktion in den Tickets der Projektmanagement Software BCS. Hier werden aufgabenspezifische Informationen gegeben und Fragen beantwortet.

3.4 Vorgehen innerhalb des Projektes

Um das Projekt bearbeiten zu können, ist die Festlegung eines Vorgehens innerhalb des Projektes notwendig. Neben der Wahl des Vorgehensmodells, spielt hierbei auch die Definition von Begleitaufgaben und die Identifikation von phasenspezifischen Aufgaben eine wichtige Rolle. Die Phasen sind je nach Vorgehensmodell unterschiedlich definiert. Ein weiterer wichtiger Aspekt ist die Umsetzung des festgelegten Vorgehens in der Projektmanagement-Software BCS.

3.4.1 Vorgehensmodell

Für die Umsetzung des Projektes wurden durch das Fachgebiet System- und Software-Engineering drei verschiedene Modelle vorgeschlagen. Diese setzen sich aus dem Wasserfallmodell, dem Modell des Unified Process und dem agilen Vorgehen zusammen.

Die Projektgruppe hat sich gemeinsam mit den Fachbetreuern für das Modell des Unified Process entschieden. Hauptgrund für die Wahl, waren schlechte Erfahrungen mit agilen Vorgehensmodellen seitens der Fachbetreuer, wodurch eben jenes Vorgehensmodell ausschied. Zwischen Wasserfall- und Unified-Process-Modell wurde sich gegen das Wasserfallmodell entschieden, da die klare Abgrenzung der Phasen unrealistisch erschien. So sollte das Projekt auch zu einem späteren Zeitpunkt in der Lage sein auf eventuelle Änderungen mit Planung und Konzeption, sowie Entwurf reagieren zu können. Unified Process bietet diese Vorteile.

3.4.2 Projektstrukturplan

Um einen Projektstrukturplan erstellen zu können, wird eine Auswahl von definierten Aufgaben und Arbeitspaketen benötigt. Diese unterteilen sich in Projektphasen-spezifische Aufgaben und Begleitaufgaben, welche während des gesamten Projektes aktiv sind.

Innerhalb des Vorgehensmodelles Unified-Process wurden die Projektphasen Planung und Entwurf, Implementierung und Validierung identifiziert. Diese orientieren sich auch an den durch das bewertende Fachgebiet System- und Software-Engineering festgelegten Review-Terminen. Anhand der Phasen konnten die spezifischen Aufgaben Projektplanung und Konzeption, Implementierung und Validierung identifiziert werden. Zwei dieser Aufgaben ziehen auch eine Begleitaufgabe nach sich, welche über den gesamten Projektzeitraum läuft und dafür vorgesehen ist, phasenübergreifende oder phasenverschobene Tätigkeiten abzudecken. Hierzu existieren die Aufgaben „Controlling“ als Planungsaufgabe, „Allgemeine Entwicklungsaufgaben“ für die Implementierung und „Review“ für Test- und Dokumentationsaufgaben. Zusätzlich wurden die Begleitaufgaben „Kommunikation“, „Recherche“ und „IT-Administration“ definiert, da innerhalb dieser Aufgaben ein ungefähr konstanter Aufwand während der Dauer des Projektes zu erwarten ist. Um die Aufwände für die interne Organisation und allgemeine Organisationsaufgaben während des Projektes ebenfalls erfassen zu können, wurden für die Projektleiter zusätzlich die Aufgaben „Projektorganisation“ zum Start des Projektes und „Allgemeine Projektorganisation“ während des Projektes angelegt.

Aufschluss über die Projektstruktur gibt der sich im Anhang befindliche Projektstrukturplan (Abbildung 15).

3.4.3 Aufwandsabschätzung

Um Aufwände innerhalb des Projektes abschätzen zu können, wurde jeder identifizierten Aufgabe aus Punkt 3.4.2 ein Planaufwand zugewiesen. Da das Pflichtenheft zusätzlich zu den Pflicht-Anforderungen auch Kann-Anforderungen enthält, sollte die gesamte vom Studienplan zur Verfügung gestellte Zeit zur Umsetzung des Projektes genutzt werden. In den Personalressourcen aus 3.2.1 wurde jedem Mitglied der Projektgruppe eine Gesamt-Arbeitszeit von 240 Stunden festgelegt. Aus diesem

Gesamtarbeitszeit-Kontingent wurden für die Begleitaufgaben unterschiedliche Grundlasten festgelegt. Diese Grundlasten sind in Tabelle Abbildung 1 dargestellt.

Aufgabe	Projektleiteraufwand	Entwickleraufwand
IT-Administration	14	14
Kommunikation	56	56
Recherche	42	42
Allgemeine Projektorganisation	28	0
Controlling	14	14
Allgemeine Entwicklungsaufgaben	14	14
Review	14	14
SUMME	182	154

Abbildung 1: Begleitaufgaben Aufwände

Abzüglich dieser Grundlasten entstehen die Planaufwände für die phasenspezifischen Aufgaben. Dies sind bei den Projektleitern 152,5 Stunden und bei den Entwicklern 189,5 Stunden. Geht man davon aus, dass jede Phase eine gleiche Projektdauer hat, entsteht pro Phase ein Planaufwand für Projektleiter von knapp über 50 Stunden und bei Entwicklern von etwas mehr als 63 Stunden. Genauer wird dies in der folgenden Tabelle über phasenspezifische Aufwände aufgeschlüsselt.

Phase	Aufgabe	Projektleiteraufwand	Entwickleraufwand
Planung und Entwurf	Anforderungs- und Risikoanalyse	9	15
	Feinkonzeption	9	15
	Projektorganisation	4	0
Implementierung	Umsetzung	22	30
Abschlussphase	Validierung	12	24
	Übergabe	2	2
SUMME		58	86

Abbildung 2: phasenspezifische Aufwände

Addiert man die Aufwände aus den Begleitaufgaben mit den Aufwänden aus den phasenspezifischen Aufgaben, erhält man die 240 Stunden, die jeder Projektmitarbeiter aufwenden muss.

3.4.4 Umsetzung in BCS

Die für das Projekt festgelegte Struktur, inklusive des Projektstrukturplans und der Aufwandsplanung wurde für die Umsetzung in die Projektmanagement-Software BCS eingefügt. Für die Phasen wurde

unterhalb des Projektes jeweils ein Arbeitspaket mit der Dauer der Phase angelegt. Unterhalb des Arbeitspaketes wurden die phasenspezifischen Aufgaben integriert. Die Begleitaufgaben befinden sich als angelegte Aufgaben in der Strukturhierarchie direkt unterhalb des Projektes. So wurde eine Anpassung der Projektstruktur in BCS an die Struktur des realen Projektes mit dem Vorgehensmodell Unified-Process geschaffen.

Um die Iterationen in der Entwurfsphase mittels BCS zu berücksichtigen wurden einzelne Teilaufgaben als Tickets angelegt. Ein Ticket beschreibt einen Arbeitsauftrag. In BCS gibt es die Möglichkeit Tickets mit sogenannten Schlagwörtern zu versehen. Damit wurde eine Struktur erstellt, mit der es möglich ist, durch Setzen und Entfernen von Schlagwörtern eine Iteration inklusive der darin enthaltenen Aufgaben durchzuführen. Zusätzlich können innerhalb eines Tickets sogenannte Nachfolgetickets erstellt werden. Diese Nachfolgetickets bieten noch einmal eine Strukturebene unterhalb eines Tickets. In dem Modell des Unified-Process würde nun ein Ticket mit seinen Nachfolgetickets einer Iteration entsprechen.

Um dies zu verdeutlichen, wird der Prozess an einem Beispiel aufgezeigt. Ein Ticket wird für die Implementierung des Baumes angelegt. Ein Projektleiter weist das Ticket einer Aufgabe zu und setzt neben den Oberflächen- und Dokumentationsschlagworten das Schlagwort „Konzeption“. Anschließend weist er das Ticket einem Mitarbeiter zu, der durch das Schlagwort „Konzeption“ erkennt, dass ein Entwurf für das Ticket zu erstellen ist. Sobald er dies umgesetzt hat, gibt er das Ticket dem Projektleiter zurück, welcher nun ein Nachfolgeticket für die Implementierung anlegt und den Entwurf anhängt. Der Mitarbeiter, der den Navigationsbaum nun implementiert, hat alle dazu benötigten Informationen aus dem Entwurf. Sobald die Implementierung abgeschlossen ist, wird das Schlagwort „Test“ gesetzt. Nun bekommt ein weiterer Mitarbeiter das Ticket zum Testen. Ist der Test erfolgreich, wandert das Ticket mit dem neuen Status „Abnahme“ zum Projektleiter. Dieser nimmt nun das übergeordnete Ticket, welches noch die Dokumentationsschlagworte beinhaltet und gibt es weiter zur Dokumentation. Wenn der Baum im Handbuch dokumentiert ist, und auch die relevanten Review-Handbücher (ebenfalls durch Schlagworte gekennzeichnet) ergänzt wurden, werden die Schlagworte wieder entfernt. So ist es möglich zu jedem Zeitpunkt Aufschluss darüber zu erhalten, in welchem Status das Ticket ist und was noch erledigt werden muss.

Nach der Anforderungsanalyse und der Grobkonzeption wurden anhand des Pflichtenheftes sämtliche Anforderungen in Form von Tickets in der Projektmanagement-Software angelegt. So ist sichergestellt, dass zu jeder Anforderung jederzeit alle Informationen verfügbar sind.

Zusätzlich ist BCS in der Lage Diagramme und Auswertungen zu erstellen. Diese werden sowohl für die Review-Dokumente, als auch für die Präsentationen genutzt. Auch die Aufwandspläne sind in BCS angelegt. So kann zum Ende des Projektes eine exakte Auswertung des Projektes erstellt werden.

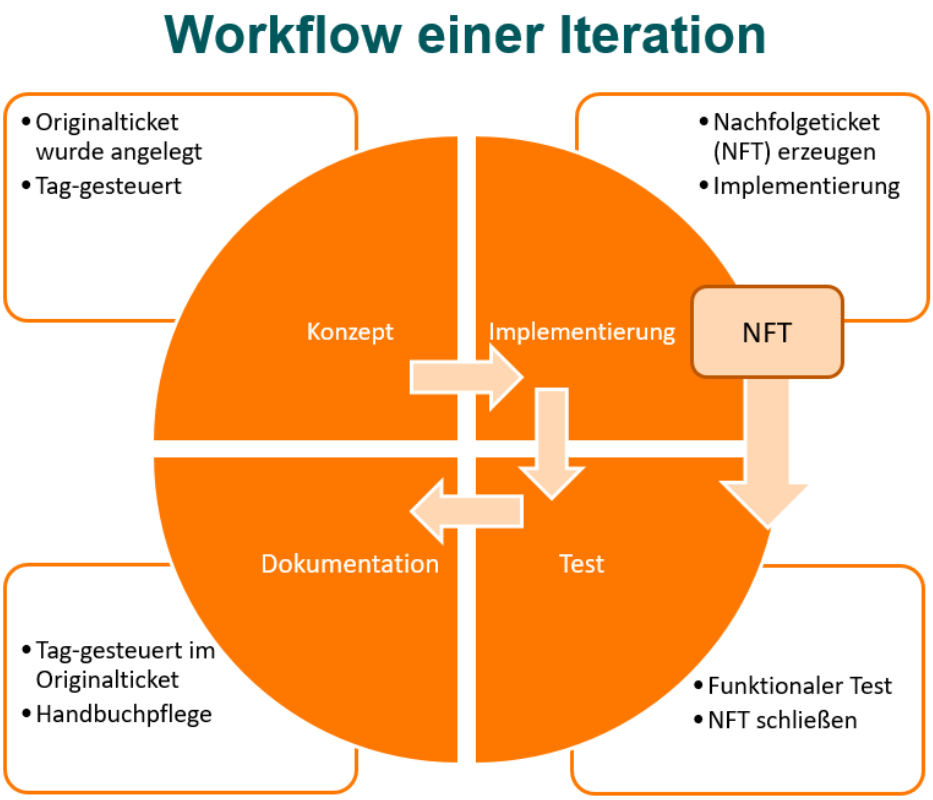

Abbildung 3: Workflow einer Iteration

3.5 Projektziele

Das Primärziel des Projektes ist die Vereinfachung des Vorbereitungsprozesses auf Praktika der Technischen Informatik. Studierende können BEAST verwenden um Blockschaltbilder online anzulegen und vor dem Praktikum bereits zu überprüfen. Hierdurch erfolgt eine Zeitersparnis auf Seiten der Studierenden, da die Vorbereitung auf das jeweilige Praktikum schneller und effektiver erstellt werden kann.

Aus dem Primärziel ergeben sich unterschiedliche Sekundärziele. Hierzu gehört auch die Erleichterung der Aufgaben des Praktikum-Betreuers, da die Vorbereitungen bereits vor Antritt des Praktikums auf Richtigkeit geprüft werden können. Ein weiteres Sekundärziel ist es, schnelleres Verständnis für die Funktionalität von einigen, in den Lehrveranstaltungen verwendeten, zusammengesetzten Komponenten (wie zum Beispiel FlipFlops) zu schaffen.

3.6 Terminplanung

Der Projektzeitraum beginnt mit der allgemeinen Einführungsveranstaltung zum Softwareprojekt am 06.04.2017 und endet mit der Abschlusspräsentation am 06.07.2017. Innerhalb dieses Zeitraums wird das Projekt organisiert. Zwischenergebnisse werden mittels Meilensteine innerhalb des Projektzeitraums terminlich festgelegt. Eine Übersicht der Meilensteine befindet sich in folgender Tabelle:

Vorbereitungsphase	
Überprüfung Lastenheft	06.04.2017
Freigabe Lastenheft	13.04.2017
Grobentwurf	
Freigabe/Präsentation Pflichtenheft	21.04.2017
Feinentwurf	
Freigabe/Präsentation Feinentwurf	03.05.2017
Implementierung	
Fertigstellung GUI	18.05.2017
Implementierungsabschluss	07.06.2017
Test	
Testabschluss	30.06.2017
Abschluss	
Abschlusspräsentation	06.07.2017
Übergabe/Bewertung	07.07.2017

Abbildung 4: Meilensteine

3.7 Machbarkeitsanalyse

Um abzuschätzen, ob das Projekt innerhalb der vorgegebenen Zeit durchführbar ist, wurde eine grobe Machbarkeitsanalyse gestartet. Innerhalb dieser Analyse wurden verschiedene Entwurfsmethoden verglichen und Entscheidungen getroffen.

Da ein Teil der Funktionalität in simcir.js bereits vorhanden ist, fällt für die Entwicklung die Einarbeitung in ein komplett neues Framework größtenteils weg. Simcir.js kann ebenfalls als Beispiel einer funktionierenden Quellcode-Basis genutzt werden.

So sind in simcir.js Beispiele für die Implementierung von Grundfunktionen wie der freien Anordnung, sowie der Schaltungssimulation enthalten. So wurde anhand der Anforderungen des Pflichtenheftes eine Tabelle mit den gewünschten Funktionalitäten und ihrer Verfügbarkeit in simcir.js erstellt.

Funktionalität	**bereits in simcir.js vorhanden**
Projektmanagement (Anlegen, Öffnen, Exportieren)	Nein
Bibliotheken-Management (Anlegen, Importieren, Exportieren)	Nein

Basisbibliotheken	Vorhanden, aber nicht vollständig
Komponentenmanagement (Anlegen, Bearbeiten, Löschen)	Nein
Grundkomponenten	Ja
Baumauswahl inklusive Verschieben und Bibliothekenzuordnung	Nein
Toolbox-Auswahl	Ja
Freie Komponentenanordnung	Ja
Komponentenlöschung	Ja
Komponentenverbindung	Ja
Simulation/Interaktion	Unvollständig
Bereichsauswahl	Ja
Tooltip-Anzeige	Nein
Tab-Struktur	Nein
Editor-Arbeitsfläche Bewegen	Nein
Zoomen der Arbeitsfläche	Nein
Rotieren von Komponenten	Nein
Parametereingabe von Komponenten	Unvollständig (nur Name)
Speichern/Extrahieren von Komponenten	Nein
Bedienung via Shortcuts	Nein
Simulation der Ein-Ausgänge	Unvollständig
Simulation der Komponenten	Unvollständig (da Komponenten unvollständig)
Benutzerinteraktion	Unvollständig (da Komponenten unvollständig)
Export in explizite Dateiformate	Nein
Export in menschenlesbares Format	Unvollständig (Daten im menschenlesbaren Format, allerdings kein Export)

GUI	Unvollständig (nur Toolbox und Arbeitsfläche) Baum, Menüleiste, Toolbar fehlen
Integrationsschnittstelle	Nein

Abbildung 5: simcir.js vorhandene Funktionalität

Des Weiteren gibt es für JavaScript-Funktionen, sowohl für Oberflächenerstellung als auch die Implementierung der Funktionalität, viele öffentliche Foren und Beispiel-Sammlungen, an denen sich die Entwickler bei der Durchführung orientieren können.

Durch das Vorhandensein der Code-Beispiele und der in Abbildung 5 aufgelisteten simcir.js-Funktionen ist das Projekt innerhalb des Zeitrahmens umsetzbar. Ohne diese bereits vorhandene Bibliothek wäre die Machbarkeit des Projektes nicht gewährleistet.

4 Projektdurchführung

Die Projektdurchführung unterteilt sich in drei verschiedene Phasen. Hierzu gehören die Phase des Refactorings, die Designphase und die Realisierung. Mit Hilfe der Projektmanagement-Software BCS werden die Aufgaben organisiert und dokumentiert.

Hierzu wird die Struktur der Tickets genutzt, wobei jeweils zwei Tickets eine Software-Iteration beschreiben. Einzelne Anforderungen werden im Zuge einer Iteration umgesetzt. Die Verteilung einer Iteration auf zwei Tickets ermöglicht ein paralleles Arbeiten an den Review-Dokumenten für Entwurfsdokumentation und Implementierungs-Review. So gibt es für jede Iteration sowohl ein Konzeptions-Ticket, als auch ein Implementierungs-Ticket. Der Vorgang wurde in 3.4.4 visualisiert.

4.1 Refactoring-Phase

Das Refactoring ist notwendig, da simcir.js als bereits vorhandene Bibliothek mit einigen Funktionalitäten vorliegt. In diesem Vorgang wird bereits bestehende Funktionalität anhand des dafür vorhandenen Quellcodes analysiert. Anschließend erfolgt eine Strukturierung des Quellcodes und Anpassung an das MVC-Entwurfsmuster. Hierfür wird die komplexe simcir.js-Bibliothek in autarke Teile zerlegt, die im BEAST Webprojekt wiederverwendet werden können.

4.1.1 SimcirIF

Das SimcirIF ist eine Schnittstelle, die den Zugriff auf die simcir.js Funktionalität ermöglicht. Da simcir.js als eigene, umstrukturierte Bibliothek im Projekt erhalten bleibt, müssen Zugriffe auf einzelne Funktionen standardisiert stattfinden. Das SimcirIF regelt dabei die Kommunikation und gewährleistet die Zugriffe auf die Arbeitsfläche. Diese Zugriffe können entweder durch die Baumansicht zur Anordnung von Komponenten stattfinden, oder durch die Toolbar zur Interaktion mit einzelnen Komponenten, wie zum Beispiel Rotieren oder Entfernen.

Des Weiteren übernimmt das SimcirIF die Funktionalitäten des Speicherns und des Ladens. Da diese Möglichkeiten in der Bibliothek simcir.js noch nicht gegeben sind, werden sie durch Funktionen in der Schnittstelle SimcirIF implementiert. Hierbei wird jeweils die gesamte Arbeitsfläche gespeichert oder geladen.

4.1.2 Klassen-Auslagerung

Einige Klassen aus der simcir.js-Bibliothek werden mit dem Refactoring ausgelagert und als eigene, autarke Klassen verwendet. Hierzu gehören beispielsweise die SVG-Grafiken (skalierbare Vektorgrafiken), die das Darstellen der Komponenten auf der Arbeitsfläche ermöglichen. Die neu entstandenen Klassen werden als Objekte in simcir.js instanziiert und können somit analog zur vorherigen, in simcir.js aktiven Verwendung als Variablen genutzt werden.

4.2 Designphase

Innerhalb der Designphasen der Iterationen erfolgt die Konzeption der einzelnen Anforderungen und ihrer Umsetzung innerhalb des BEAST-Webprojektes. Hierzu werden Schnittstellen, Klassen und Objekte im Entwurfsmuster der objektorientierten Programmierung identifiziert und ein Entwicklungsplan erstellt. Es wird kontinuierlich ein initial angelegtes Feinkonzept überarbeitet, welches mit den Iterationen aktualisiert wird.

4.2.1 MockUp der Benutzeroberfläche

Grundlage der Konzepte bildet eine Oberflächen-Vorlage, welche die grundlegenden Bedienelemente zeigt. Diese Vorlage, als MockUp bezeichnet, stellt die Anwendung unterteilt in eine Menüleiste, einen Baum zur Komponentenauswahl und einen Arbeitsbereich dar. Hierbei unterteilt sich der

Arbeitsbereich nochmal in eine Tab-Leiste, eine Toolbar und eine Arbeitsfläche. Die Aufteilung der Oberfläche ist aus den Anforderungen in ständiger Kommunikation mit dem Fachgebiet Informations- und Kommunikationstechnik entstanden.

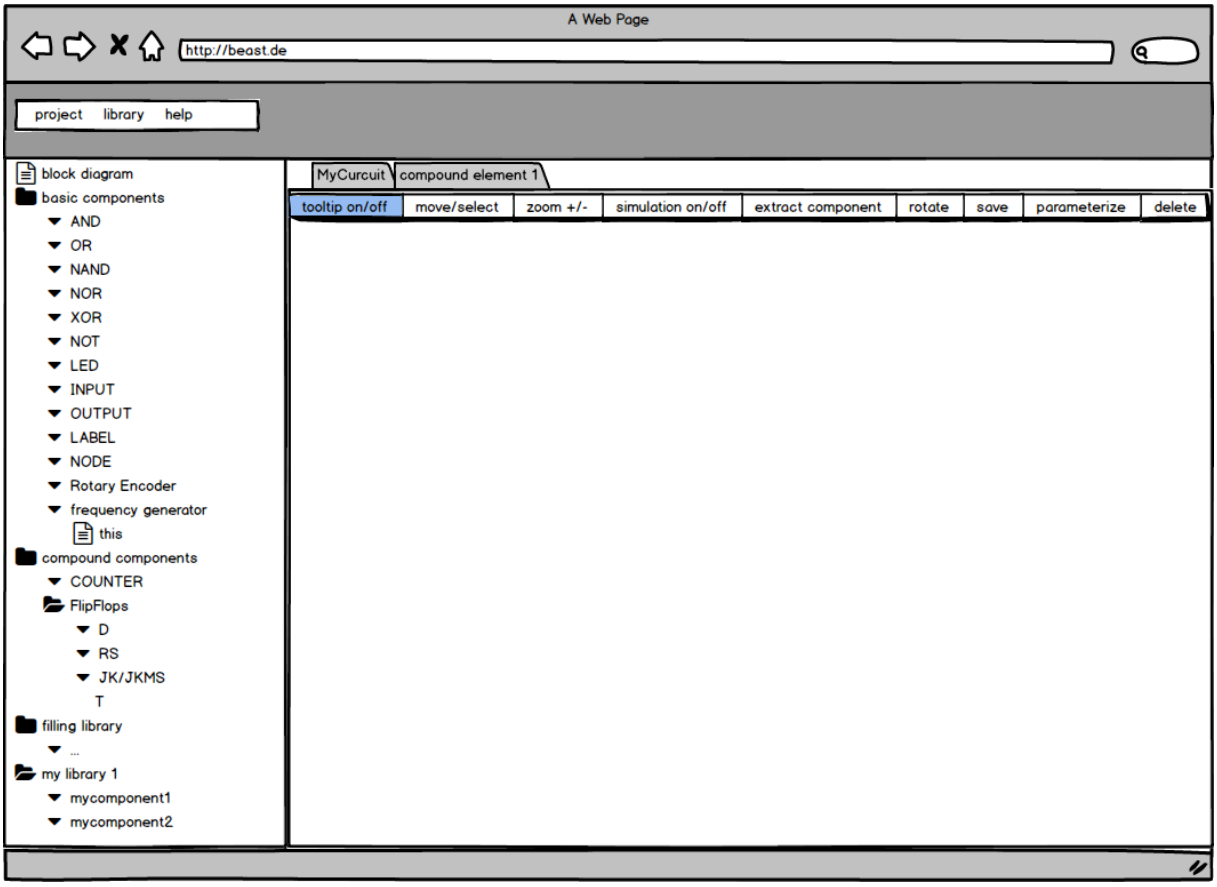

Abbildung 6: MockUp Benutzeroberfläche

Mit Hilfe des MockUps kann die Projektstruktur, sowie die Organisation von Bibliotheken in der Baumansicht gezeigt und über die Menü-Leiste bearbeitet werden. Für die Bearbeitung der Blockschaltbilder dient der Arbeitsbereich. Die Tab-Struktur ermöglicht die parallele Bearbeitung von extrahierten von Komponenten, während die Toolbar die Schaltungsoperationen steuert.

Die Oberflächenelemente werden mit dem Bootstrap-Framework erstellt. Für den Baum wird eine externe Bibliothek namens fancytree.js genutzt, welche eine Baumansicht bereits in JavaScript implementiert.

4.2.2 Konzeption der Funktionsimplementierung

Die auf die Oberflächenaktionen reagierenden Funktionalitäten werden in Controllern realisiert, welche in 4.3.4 beschrieben werden. Die Konzeption für die Funktionalität erschließt sich aus der jeweiligen Zugehörigkeit zur Oberfläche. Da jeder Oberflächenbereich über einen Controller verfügt, wird dieser auch zur Implementierung der Funktionalität genutzt.

Um die Konzeption des Projektes zu dokumentieren wird eine Entwurfsdokumentation angelegt und kontinuierlich fortgeführt.

Die folgende Tabelle zeigt die durch die Konzeption entstandenen Implementierungsansätze für die Funktionalitäten.

Funktionalität	bereits in simcir.js vorhanden	Implementierungsansatz
Projektmanagement (Anlegen, Öffnen, Exportieren)	Nein	Index.html, MenubarController, TreeController
Bibliotheken-Management (Anlegen, Importieren, Exportieren)	Nein	Index.html, MenubarController, TreeController
Basisbibliotheken	Vorhanden, aber nicht vollständig	Erweiterung durch zusätzliche Komponentenklassen
Komponentenmanagement (Anlegen, Bearbeiten, Löschen)	Nein	ToolbarController, SimcirIF,
Grundkomponenten	Ja	SimcirIF + Klassenerweiterung
Baumauswahl inklusive Verschieben und Bibliothekenzuordnung	Nein	TreeController, BEASTController
Toolbox-Auswahl	Ja	SimcirIF, TreeController
Freie Komponentenanordnung	Ja	SimcirIF
Komponentenlöschung	Ja	SimcirIF, ToolbarController
Komponentenverbindung	Ja	SimcirIF
Simulation/Interaktion	Unvollständig	SimcirIF
Bereichsauswahl	Ja	SimcirIF
Tooltip-Anzeige	Nein	SimcirIF, ToolbarController
Tab-Struktur	Nein	WorkspaceController
Editor-Arbeitsfläche Bewegen	Nein	WorkspaceController
Zoomen der Arbeitsfläche	Nein	SimcirIF
Rotieren von Komponenten	Nein	SimcirIF
Parametereingabe von Komponenten	Unvollständig (nur Name)	SimcirIF
Speichern/Extrahieren von Komponenten	Nein	BEASTController

Bedienung via Shortcuts	Nein	ToolbarController
Simulation der Ein-Ausgänge	Unvollständig	SimcirIF
Simulation der Komponenten	Unvollständig	SimcirIF
Benutzerinteraktion	Unvollständig	SimcirIF
Export in explizite Dateiformate	Nein	BEASTController, Model
Export in menschenlesbares Format	Unvollständig	BEASTController, Model
GUI	Unvollständig	Index.html
Integrationsschnittstelle	Nein	BEASTController

Abbildung 7: Implementierungsansätze

Um die gezeigten Funktionen umzusetzen werden innerhalb der Realisierungsphase die entsprechenden Funktionen in den Paketen des BEAST-Projektes implementiert.

4.2.3 BEAST-Datenmodell

Da BEAST über einen Import und einen Export von Projekten, sowie Bibliotheken verfügt, werden die Daten in strukturierter, für den Anwender lesbarer Form gespeichert. Dies geschieht in den Projekt-proprietären Dateiformaten **.beast** für Projekte und **.bdcl** für Komponentenbibliotheken.

Die gewählte Datenstruktur orientiert sich am Muster der objektorientierten Programmierung und kann als Klassendiagramm dargestellt werden.

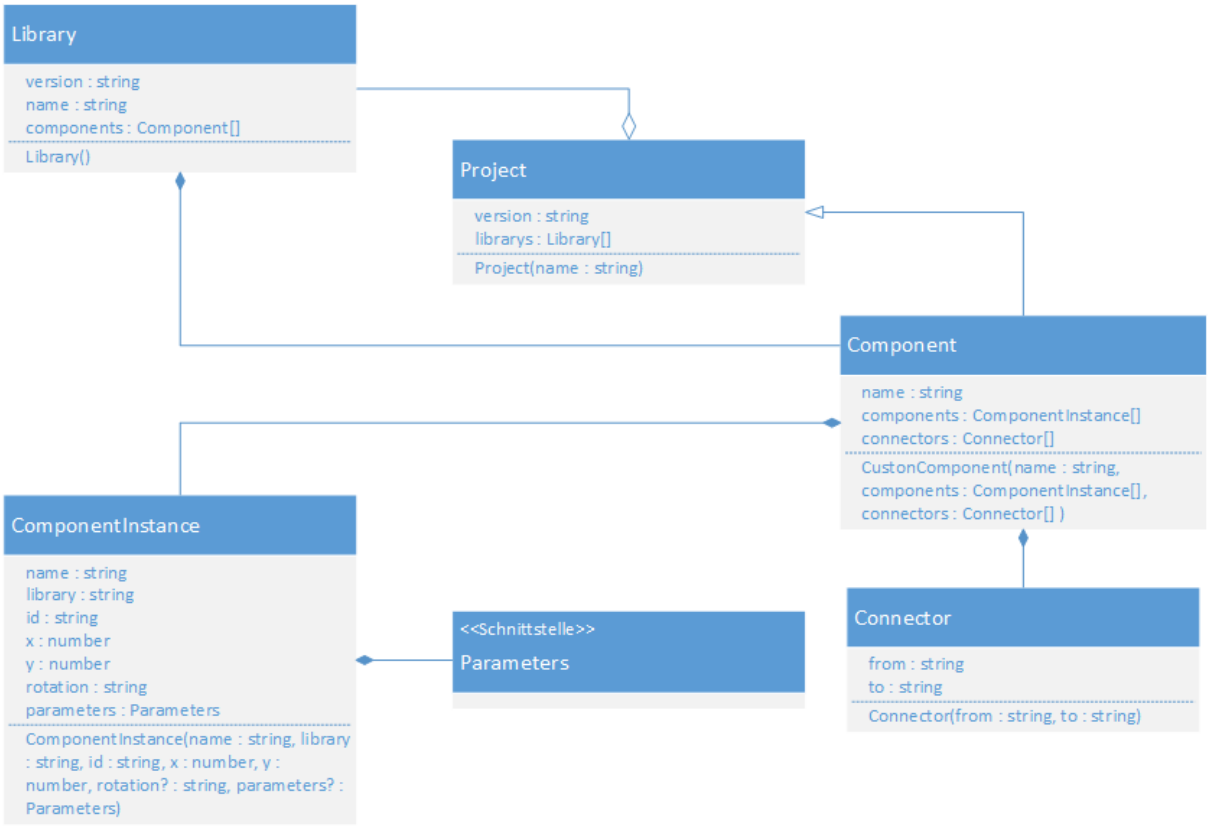

Abbildung 8: Klassendiagramm BEAST-Daten

Innerhalb des Klassendiagramms lässt sich die Zusammensetzung eines Projektes aus Bibliotheken erkennen. Sowohl Projekte, als auch Bibliotheken verfügen über eine Versionsnummer. Diese ermöglicht es Kompatibilitätsproblemen zukünftiger BEAST-Versionen mit veralteten Komponentenbibliotheken vorzubeugen. Demnach ist eine Version der Komponentenbibliotheken auch immer einer BEAST-Version zugeordnet. Eine Komponente ist dabei immer einer Komponentenbibliothek zugeordnet. Zwischen den Komponenten in der Toolbox und den auf der Arbeitsfläche angeordneten Komponenten wird hierbei terminologisch durch die Trennung *Komponente* und *Komponenteninstanz* unterschieden. Um die Parametrisierung der Komponenten auf der Arbeitsfläche zu ermöglichen, werden die Komponenteninstanzen durch die Schnittstelle *Parameter* erweitert.

Die Verbindungsinformationen zwischen Komponenten werden durch die Klasse *Verbindungen* abgedeckt, welche in einer separaten Klasse gespeichert werden.

Um die exportierten Daten für den Anwender lesbar zu gestalten, werden die Informationen der gezeigten Klassen in ein JSON-Format übernommen. Dieses Format kann den objektorientierten Charakter der Daten darstellen. Der Aufbau einer JSON-Datei wird im folgenden MockUp gezeigt.

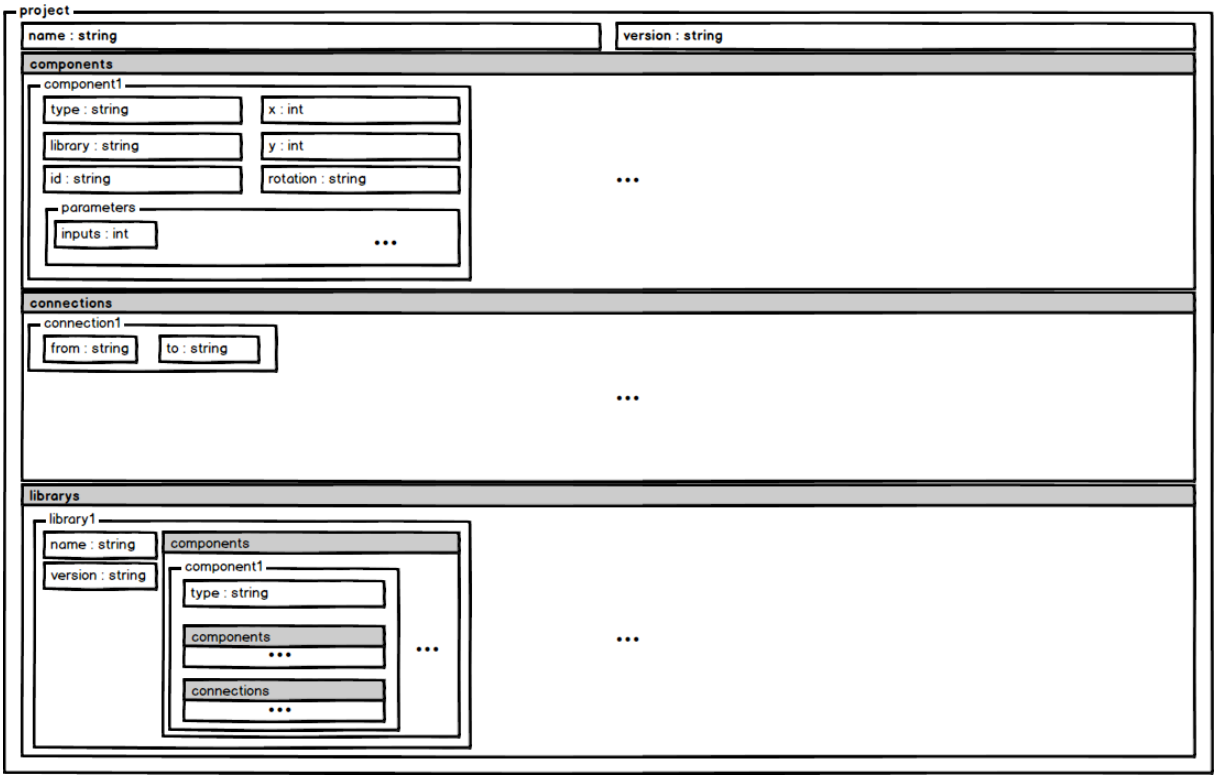

Abbildung 9: Datenstruktur BEAST

4.3 Realisierungsphase

Im Rahmen der Realisierungsphase werden Anforderungen aus der Projektplanung sowie die Vorgaben aus der Designphase umgesetzt. Im Folgenden wird dies in das Anlegen eines Webprojektes, das Erstellen der Benutzeroberfläche, die Integration der Funktionalität aus simcir.js und die Implementierung der Funktionen gegliedert.

4.3.1 Anlegen eines Webprojektes

Das Webprojekt ist eine Verzeichnisstruktur, welche alle Konfigurationsobjekte des Projektes enthält und unter Versionskontrolle steht. Die Verzeichnisstruktur wird mit Hilfe der Webstorm IDE erstellt. Dazu wird eine leere Projektvorlage ausgewählt und gespeichert. Danach wird die Verzeichnisstruktur entsprechend der Beschreibung der Arbeitspakete in der Entwurfsdokumentation erstellt. Es wird eine README-Datei angelegt, welche kurz und prägnant erläutert, worum es sich bei dem Projekt und dem Verzeichnisinhalt handelt. Es erfolgt nun der Import des simcir.js Quellcode in dem Zustand, wie er am 25.04.2017 vorliegt. Schließlich werden sukzessive weitere benötigte Bibliotheken und Konfigurationsdateien in die Verzeichnisstruktur eingefügt. Hierzu gehören auch die Webstorm IDE Konfigurationen, welche ebenfalls unter Versionskontrolle stehen sowie Konfigurationsdateien für TypeScript-Compiler und Zwischenversionen des Benutzerhandbuches.

4.3.2 Erstellen der Benutzeroberfläche

Die Grundgliederung der Benutzeroberfläche wird in der *index.html* in der Sprache HTML in Verbindung mit dem Bootstrap-Framework erstellt. Für die unterschiedlichen Oberflächenelemente werden auch unterschiedliche Implementierungsmöglichkeiten genutzt.

Die Menüleiste wird in HTML in Verbindung mit Bootstrap-Buttons implementiert. Das Bootstrap-Framework wird hierbei verwendet, weil damit eine einheitliche Gestaltung der Elemente möglich ist. Das Erscheinungsbild der Elemente kann vollständig über Stylesheets (*CSS*) bearbeitet werden.

Im Gegensatz zur Menüleiste wird die Baumansicht nicht in HTML und Bootstrap erstellt. Hierzu wird eine externe JavaScript-Bibliothek namens fancytree.js verwendet. Diese Bibliothek bietet bereits die Möglichkeit eine Baumansicht zu erstellen und diese mit Daten zu füllen. Die Darstellung des Baumes ist ebenfalls über Stylesheets anpassbar.

Für den Arbeitsbereich wird eine Kombination aus JavaScript-Funktionen und HTML verwendet. Da die Arbeitsfläche bereits in simcir.js in Form von JavaScript implementiert ist, wird diese auch verwendet und nicht neu erstellt. Die die Arbeitsfläche umgebenden Elemente Toolbar und Tab-bar müssen durch die mehrfache Anzeige dynamisch erzeugt werden. Hierzu dient JavaScript-gesteuertes, dynamisches Darstellen von HTML-Quellcode. Der für die Toolbar benötigte HTML-Quellcode wird dabei dynamisch in jedem offenen Tab des Arbeitsbereiches erzeugt und dargestellt.

4.3.3 Integration der simcir.js Funktionalität

Die während des Refactorings entstandenen Teilkomponenten aus simcir.js können in den TypeScript-Paketen des Web-Projektes wiederverwendet werden. Ein Beispiel stellen die skalierbaren Vektorgrafiken dar, welche in simcir.js zur Darstellung der Oberfläche verwendet werden. Diese sind nach dem Refactoring als eigene Klasse vorhanden und können durch die Controller instanziiert werden.

Wie in 4.1.1 beschrieben, können Funktionen der simcir.js Bibliothek, welche die simcir.js-Arbeitsfläche betreffen, über das SimcirIF aufgerufen werden. Die in simcir.js enthaltene Funktionalität wird dabei nicht bearbeitet, sondern nur um weitere Funktionen erweitert. Hierzu gehören die Parameterbearbeitung, das Rotieren von Komponenten, das Ziehen von Leitungen über Knotenpunkte, sowie die Leitungssimulation und die Parameter-Tooltips.

Die genannten Funktionen werden als Methoden in die aus dem Refactoring entstandene, strukturierte simcir.js Bibliothek integriert und können anschließend als Funktionen durch den zugehörigen Controller aufgerufen werden.

4.3.4 Implementierung der Funktionen

Nach der Erstellung der Oberfläche werden die Funktionalitäten in den Controllern implementiert. Die Controller stellen Funktionen bereit, welche von der Benutzeroberfläche aufgerufen werden. Hierzu werden die Controller analog zur BEAST-Oberfläche strukturiert.

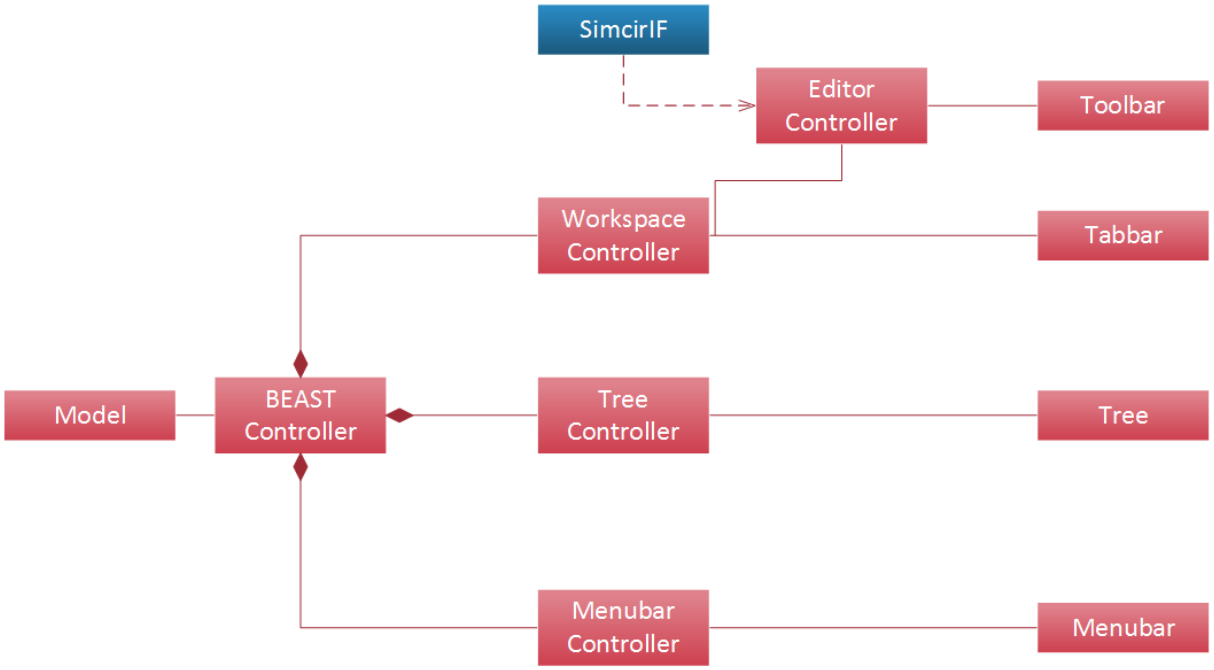

Abbildung 10: MVC-Struktur BEAST

In den einzelnen Oberflächen-Controllern wird die jeweils zum Benutzeroberflächenbereich zugehörige Funktionalität implementiert. Änderungen am Datenmodell werden einheitlich durch das *Model* umgesetzt. Der *BEAST Controller* stellt die Kommunikationseinheit der einzelnen Controller mit dem Model dar. Er verfügt über eine Instanz des Models und gibt diese an die Oberflächen-Controller weiter.

Durch Benutzeraktionen in der Oberfläche werden Funktionen in den Controllern ausgelöst, welche wiederum die Daten, welche durch das *Model* und den *BEAST Controller* bereitgestellt werden, bearbeiten. Anschließend gibt das *Model* die geänderten Daten wieder an die Controller, welche eine Aktualisierung der Oberfläche auslösen.

Diese Strukturierung setzt das Entwurfsmuster *Model-View-Controller* um. Das in Abschnitt 4.1.1 beschriebene SimcirIF erweitert den *Editor Controller*, welcher Teil des *Workspace Controllers* ist. Diese beiden Controller bilden ein Konstrukt, welches sowohl die bereits vorhandenen Funktionalitäten aus simcir.js, als auch neue Funktionen wie die Toolbar und die Tabbar, vereint. Die Integration der bereits in simcir.js vorhandenen Funktionalität ist in Abschnitt 4.3.3 beschrieben.

5 Projektabschluss

Mit dem Projektabschluss werden die Ergebnisse des Projektes erläutert und dargestellt. Hierzu erfolgt zu Beginn eine allgemeine Auswertung, gefolgt von einer Zusammenfassung der umgesetzten Funktionalitäten. Die Auswertung der Ressourcen wird in einem weiteren Kapitel erläutert. Um die Funktionalität auszuwerten, werden im Anschluss die Testergebnisse auf Grundlage des Testdrehbuches in Form von Reporten aufgezeigt.

5.1 Allgemeine Auswertung

In der Allgemeinen Auswertung werden das Projektvorgehen und Abweichungen zu den Planungen aus der Planungs- und Entwurfsphase erläutert.

Das Vorgehensmodell des Unified Process wurde mit Hilfe des Projektmanagement-Tools Projektron BCS umgesetzt und während des gesamten Projektes angewendet. Dabei konnte das Vorgehen aus der Planung genutzt werden. Die Ticketstruktur ermöglichte den Projektleitern sowohl eine Möglichkeit Projektabläufe zu steuern, als auch eine Grundlage für die Kommunikation mit den Fachbetreuern, welche innerhalb des Projekts als Kunden dienen.

Die Zeiterfassung, welche ebenfalls durch BCS realisiert wurde, konnte zu jeder Zeit Aufschluss darüber geben, welche Projektmitarbeiter freie Kapazitäten haben. So konnten einzelne Implementierungsaufgaben dynamisch verteilt werden.

Die Rollenverteilung aus der Projektplanung wurde mit kleinen Abweichungen umgesetzt. Hier entstanden Änderungen, da das Projekt mit einem sehr engen Zeitfenster geplant wurde. So übernahmen Projektmitarbeiter, welche als Entwickler eingeplant wurden mehr Dokumentations- und Testaufgaben als Implementierungen. Ursache dessen war das Einsparen von Einarbeitungszeit, welche von den Projektleitern als kritischer Faktor für einen erfolgreichen Projektabschluss angesehen wurde.

Die Kommunikation des Teams basierte zu Beginn des Projekts auf zwei wöchentlichen Treffen. An einem der beiden Treffen nahmen auch die Auftraggeber, bzw. Kunden oder Fachbetreuer teil. Mit dem Laufe des Projekts stellte sich heraus, dass zwei Treffen und die damit einhergehende halbwöchige Zeitdauer bis zur nächsten Möglichkeit für die Klärung von Fragen nicht ausreichten. Daher vereinbarten beide Projektleiter Aufenthaltszeiten, in den zur Verfügung stehenden Räumlichkeiten, sodass diese Zeiten ebenfalls zur Klärung von Fragen dienten.

Das Projektziel, Schaltungen mit BEAST zu erstellen und zu simulieren wurde erreicht. Einzelheiten zu den Funktionalitäten werden in Kapitel 5.2 erläutert. Die Sekundärziele wie das Erleichtern der Aufgaben der Praktikumsbetreuer können mit Abschluss des Projekts noch nicht bewertet werden. Da BEAST zum Zeitpunkt des Projektabschlusses noch nicht im Praktikumsbetrieb aufgenommen wurde, sind noch keine Erfahrungen zu Erleichterungen möglich.

5.2 Auswertung der umgesetzten Funktionalitäten

Das Projekt wurde den Anforderungen entsprechend umgesetzt und durchgeführt. Als Ergebnis ist eine Webanwendung mit dem Namen BEAST entstanden, die alle im Pflichtenheft aufgeführten Anforderungen erfüllt. Die Anwendung setzt sich aus der Projektorganisation, der Schaltungsbearbeitung und der Schaltungssimulation zusammen.

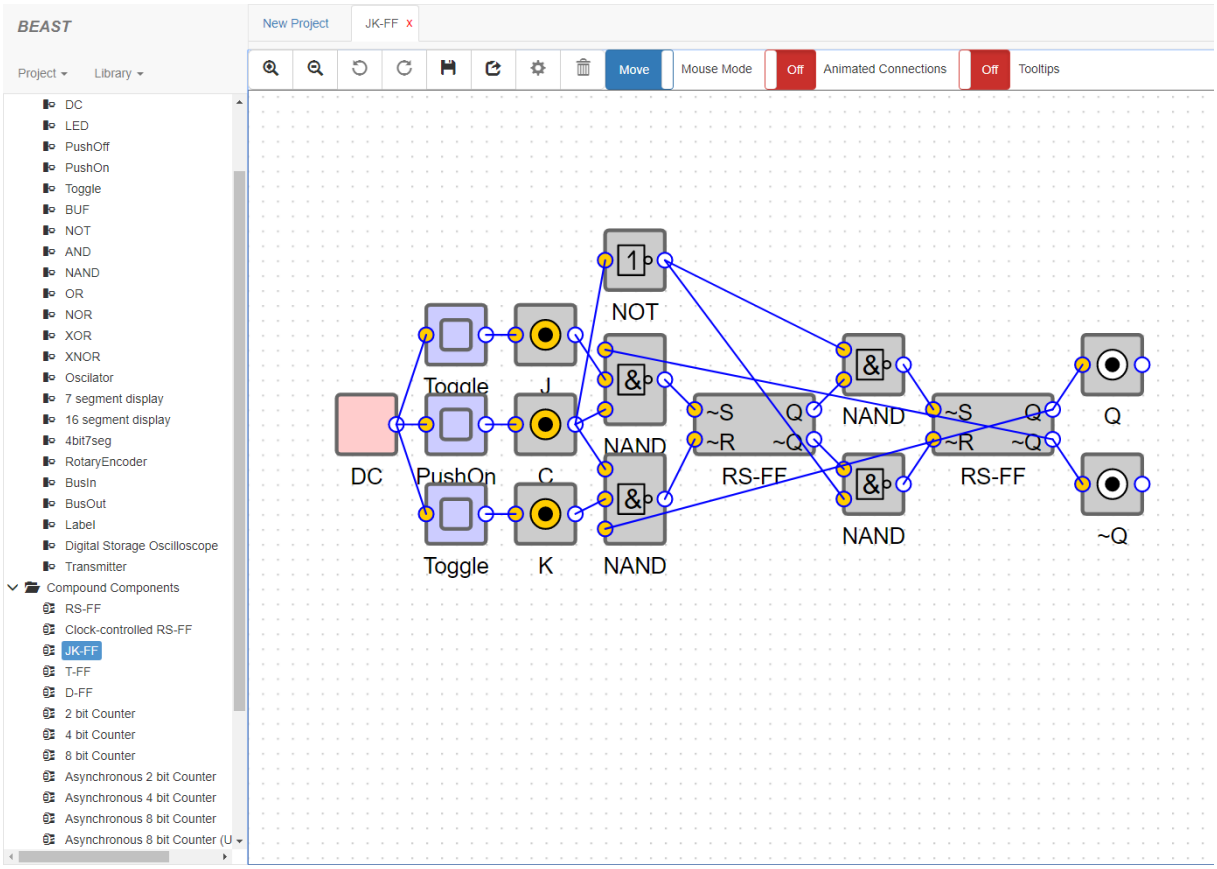

Abbildung 11: BEAST-Oberfläche

Die Projektorganisation, welche ein BEAST-Projekt, die dazugehörigen Bibliotheken und die darin enthaltenen Komponenten umfasst, kann durch eine Menüleiste (siehe Abbildung 11) bearbeitet werden. Hierbei können Projekte angelegt, gespeichert und exportiert werden. Des Weiteren kann der Benutzer Bibliotheken erstellen, die ausgewählte Komponenten enthalten. Bibliotheken können ebenso importiert und exportiert werden. Die Darstellung der Projektstruktur, die durch die Projektorganisation entsteht, erfolgt durch eine Baumansicht. Hierbei werden Bibliotheken als übergeordnete Knoten dargestellt, welche die Komponenten als Elemente enthalten. Änderungen an der Projektstruktur können auch durch Interaktionen mit der Baumansicht erfolgen. Hierzu hat der Benutzer die Möglichkeit Komponenten via Drag&Drop zwischen Bibliotheken zu verschieben und über ein Kontextmenü Komponenten aus dem Projekt zu entfernen.

Des Weiteren dient der Baum als Toolbox für den Arbeitsbereich. Der Arbeitsbereich setzt sich aus einer Arbeitsfläche und einer Toolbar zusammen. Er unterliegt einer Tab-Struktur, wobei ein Tab sich aus Toolbar und Arbeitsbereich zusammensetzt.

Auf der Arbeitsfläche können Schaltungen bearbeitet werden. Hierzu kann der Benutzer die Komponenten via Drag&Drop aus dem Baum der Arbeitsfläche hinzufügen und frei anordnen. Durch das Ziehen von Leitungen von Eingängen zu Ausgängen können Verbindungen zwischen Komponenten erzeugt werden. Um die Schaltung zu bearbeiten können mittels Toolbar verschiedene Funktionen aufgerufen werden. Hierzu gehören das Entfernen von Komponenten oder die Änderung von Komponentenparametern.

Die Simulation erfolgt während der Bearbeitung. Dabei werden die logischen Zustände der Ein- und Ausgänge farblich markiert. Eine Toolbaroption bietet dem Benutzer die Möglichkeit auch die Leitungen entsprechend ihrer Start- und Zielknoten farblich zu markieren.

Eine Schnittstelle innerhalb der Quellcode-Dateien der Anwendung bietet die Möglichkeit BEAST als Modul in eine bestehende Webseite einzubetten und Eingänge mit Werten zu belegen, bzw. Ausgänge zu lesen. Sobald die Eingänge der Schaltung über diese Schnittstelle gesetzt werden, wird dem Benutzer dies durch ein Textfeld und eine alternative Darstellung der Ein- und Ausgänge mitgeteilt.

Alle Anforderungen, welche dem Pflichtenheft zu entnehmen sind, wurden in ein umfangreiches Testset überführt. Dieses Testset wird mit der Software testOffice von Spirit Testing Software & Services manuell durchgeführt. Eine in der Planung vorgesehene Testautomatisierung, welche mit testOffice ebenfalls möglich ist, wurde nicht umgesetzt. Ursache dafür ist ein sehr großer Zeitaufwand für die Erstellung der Testautomatisierungsskripte, welche keinen Mehrnutzen für das Projekt gehabt hätten. Sollte das Projekt regelmäßig erweitert und in ständigem Gebrauch finden, macht eine Testautomatisierung im Anschluss Sinn.

Die Protokolle der Tests befinden sich, ebenso wie die Definition der Testfälle, im Testdrehbuch.

5.3 Auswertung der Personalressourcen

Anhand der Zeiterfassung können sowohl für alle Projektphasen, als auch alle Aufgaben die gebuchten Aufwände ausgewertet werden. Die gebuchten Aufwände werden mit den geplanten Aufwänden aus der Aufwandsabschätzung verglichen.

Generell kann festgestellt werden, dass das Projekt in weniger als der geplanten Zeit abgeschlossen werden konnte.

Aufgabe	Geplanter Projektleiteraufwand	IST	Geplanter Entwickleraufwand	IST
IT-Administration	14 (28)	**21**	14 (84)	**15**
Kommunikation	56 (112)	**87**	56 (336)	**211**
Recherche	42 (84)	**16**	42 (252)	**78**
Allgemeine Projektorganisation	28 (56)	**29**	0	**0**
Controlling	14 (28)	**15**	14 (84)	**0**
Allgemeine Entwicklungsaufgaben	14 (28)	**42**	14 (84)	**46**
Review	14 (28)	**45**	14 (84)	**10**
SUMME	182 (364)	**255**	154 (924)	**360**

Abbildung 12: Begleitaufgaben Aufwände

Auffällig ist, dass die tatsächlich gebuchten Zeiten deutlich unter den geschätzten Aufwänden liegen. Ursache dessen ist, dass die Begleitaufgaben auch als Pufferaufwand dienen. Als Beispiel dienen hier die Allgemeinen Entwicklungsaufgaben, welche vom Entwicklerteam kaum benötigt wurden. Da in Folge des engen Zeitfensters vorwiegend an Iterationen innerhalb der Phasen gearbeitet wurde um die gewünschte Funktionalität zu erzielen.

Eine weitere Ursache liegt in der Buchungsstrategie der Projektmitarbeiter. Während des Projekts konnte beobachtet werden, dass in vielen Fällen keine eindeutige Trennung zwischen Beispielimplementierungen und aufgabenrelevanten Implementierungen erfolgte.

Die fehlenden IST-Zeiten hinsichtlich der Kommunikation sind in der Planungsphase entstanden, da bei Fragen in vielen Fällen auf ein nächstes Treffen gewartet wurde als das Problem nur mit den betreffenden Projektmitarbeitern zu besprechen. Der Kommunikationsaufwand erhöhte sich mit dem Anbieten der Aufenthaltszeiten der Projektleiter, welche zur Problemklärung genutzt werden konnten.

Auch ist das Controlling der Entwickler auffällig, da hierfür keine Aufwände gebucht wurden. Dies ist darauf zurückzuführen, dass in der Entwicklungsphase keine Konzeptänderungen aufgetreten sind, sondern mit der Implementierung bis zur Klärung gewartet wurde.

Phase	Aufgabe	Geplanter Projektleiteraufwand	IST	Geplanter Entwickleraufwand	IST
Planung und Entwurf	Anforderungs- und Risikoanalyse	9 (18)	**37**	15 (90)	**78**
	Feinkonzeption	9 (18)	**27**	15 (90)	**77**
	Projektorganisation	4 (8)	**8**	0	**0**
Implementierung	Umsetzung	22 (44)	**26**	30 (180)	**213**
Abschlussphase	Validierung	12 (24)	**17**	24 (144)	**82**
	Übergabe	2 (2)	**2**	2 (2)	**2**
SUMME		58 (114)	**117**	86 (504)	**452**

Abbildung 13: Phasenaufgaben Aufwände

In den Aufwänden der phasenspezifischen Aufgaben, erklären sich teilweise die niedrigen IST-Aufwände der Begleitaufgaben. Wie bereits beschrieben, haben die Entwickler mehr Zeit für die Umsetzung aufgewendet als vorher eingeplant. Ob dies aus einer falschen Buchungsstrategie der Entwickler oder einem Planungsfehler durch die Projektleitung entstanden ist, kann nicht eindeutig geklärt werden.

Allgemein sind die Aufwände der phasenspezifischen Aufgaben auch geringer als die geplanten Aufwände. Hier sind jedoch nicht so extreme Umstände wie bei den Begleitaufgaben zu beobachten.

Während der Durchführung konnte beobachtet werden, dass viele Aufgaben zwischen Veranstaltungen der Universität durchgeführt wurden, wodurch Aufwände nicht eindeutig festgelegt werden konnten. In einem Unternehmen, in dem die Mitarbeiter feste acht Stunden am Tag arbeiten, fällt dies deutlich leichter.

Insgesamt wurde weniger Zeit für die Umsetzung des Projekts benötigt als zu Beginn geplant. Die von der Universität für das Softwareprojekt zur Verfügung gestellte Zeit wurde vom Projekt-Team nicht vollständig gebraucht. Dies geht auch aus dem GANTT-Diagramm aus dem

5.4 Bugreview

Während des Testens und in Folge der Smoke-Tests konnten in der Anwendung einige Probleme sowohl erkannt, als auch bereits behoben werden. Da die Umsetzung eine längere Zeit als ursprünglich geplant und vorgegeben in Anspruch nahm, konnten die Fehler auch als fehlerhafte Implementierung in Folge der Implementierungstickets behoben werden.

Nebeneffekte von bereits abgeschlossenen Implementierungen wurden als Fehlertickets in BCS aufgenommen. Dabei sind in der Validierungsphase insgesamt 18 Fehlertickets erstellt wurden. Bis zur Abgabe des Projekts werden diese Fehler behoben. Da BCS als Projektmanagement-Tool voraussichtlich nicht vom betreuenden Fachgebiet genutzt wird, werden die zum Projektabschluss offenen Bugs in Schriftform den Kunden mitgeteilt.

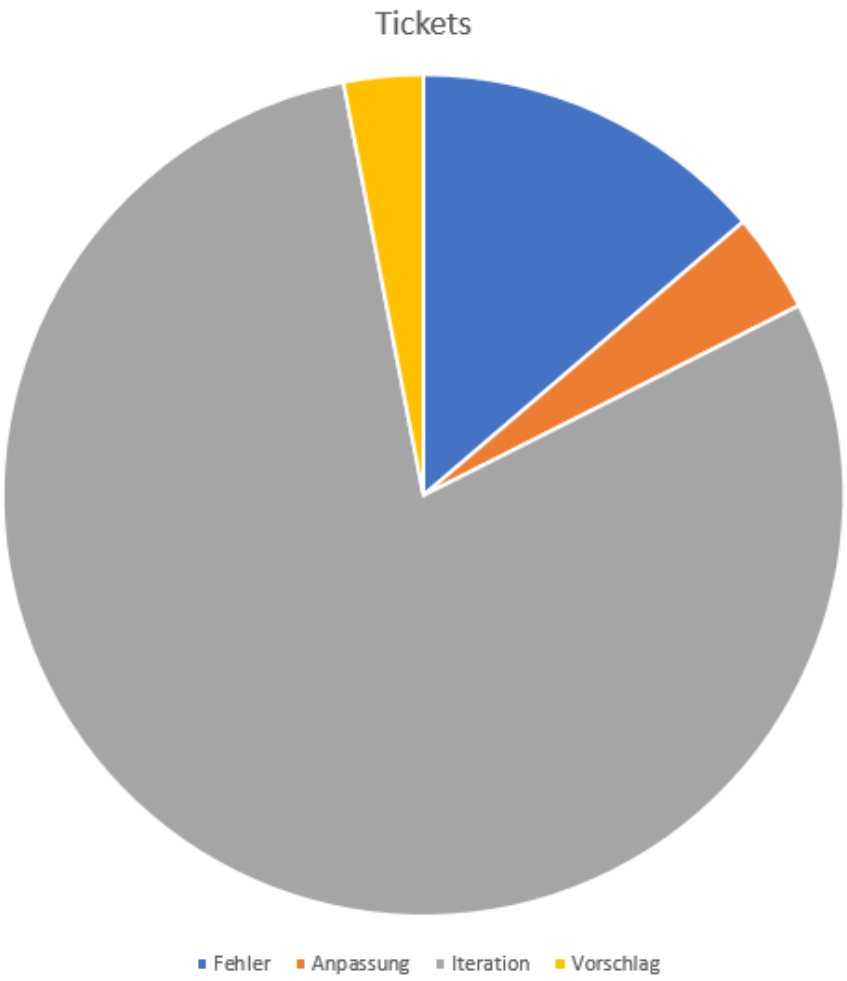

Abbildung 14: Ticketstatistik

Unter insgesamt 131 Tickets sind lediglich 18 Fehlertickets aufgetreten. Das bedeutet eine niedrige Fehlerquote und hohe Qualität während der Implementierung. Dies lässt sich auf die regelmäßigen Tests während der Implementierungsphase zurückführen.

Des Weiteren existieren bereits vier Vorschlag-Tickets zur Erweiterung der entstandenen Webanwendung.

5.5 Fazit

Im Fazit werden sowohl das Vorgehen innerhalb des Projektes, als auch der Projekterfolg kritisch bewertet. Im GANTT-Diagramm im Anhang (Abbildung 16, Stand 02.07.2017) sind noch nicht alle Aufgaben abgeschlossen. Die Auswertung der Personalressourcen kann auch anhand dieses Diagramms dargestellt werden und verdeutlicht die Ergebnisse des vorangegangenen Kapitels. Auch ist ersichtlich, dass alle Meilensteine bis zum aktuellen Stand erreicht und auch abgeschlossen wurden.

5.5.1 Bewertung des Vorgehens

Für das Projekt „Digitale Schaltungssimulation im Browser“ wurde das Vorgehensmodell des Unified Process verwendet. Innerhalb der Projektgruppe waren dabei wenig Erfahrungen mit einem solchen Vorgehensmodell vorhanden. So wurde als Grundlage ein Wasserfallmodell gewählt, welches das Projekt in Planung- und Entwurf, Implementierung und Validierung geteilt wurde. Zusätzlich wurden Begleitaufgaben definiert, welche phasenübergreifend Aufwände aufnehmen können.

Vorteil dieses Vorgehens sind die klare Strukturierung der Aufgaben und eindeutige Zuordnung von Aufwänden. Die Iterationen sind dabei als Tickets definiert und bieten die Möglichkeit jede Implementierung zu kommentieren und im Nachhinein auszuwerten. Nachteilig kann die oft parallele Bearbeitung von Iterationen gesehen werden.

Auch die fehlende Erfahrung einiger Projektmitarbeit in Bezug auf die Arbeit in Projektgruppen, hat sich zeitweise nachteilig ausgewirkt. So kam es zu Überschneidungen von Aufgaben, welche den Schnittbereich zwischen zwei Iterationen bildeten. Die klare Aufteilung in Feinkonzeption und Umsetzung bzw. Implementierung stellte die Projektmitarbeiter ebenfalls vor Probleme. Da die Feinkonzeption oft einem Grobentwurf ähnelten, kam es in der Implementierungsphase zu Verzögerungen, da kein einheitliches Konzept vorlag.

Alle Probleme, die jedoch mit dem Vorgehensmodell Unified Process einhergingen, konnten durch die Flexibilität behoben und das Vorgehen entsprechend angepasst werden. So wurde beispielsweise die Feinkonzeption entgegen der Projektplanung eher abgeschlossen, damit die Umsetzung und Implementierung früher beginnen kann. So wurden Konzeptanpassungen auch noch in der Implementierungsphase durchgeführt.

5.5.2 Bewertung des Projekterfolges

Das Projekt wurde mit den entsprechenden Anforderungen innerhalb des Zeitfensters umgesetzt. Daher kann von einem erfolgreichen Projektabschluss ausgegangen werden.

Zusätzlich zu den Anforderungen aus dem Pflichtenheft sind während der Kundengespräche weitere Anforderungen aufgetreten, welche erst mit der Umsetzung des Projekts deutlich wurden. So wurde der Navigationsbaum, obwohl nicht im Pflichtenheft aufgeführt, bereits mit einem Kontextmenü und Shortcut-Unterstützung ausgestattet.

Neben dem funktionalen Erfolg des Projekts, ist auch ein Lernerfolg der Projektgruppe vorhanden. Dies betrifft sowohl die bereits Projekt-erfahrenen Projektleiter, als auch die Entwickler. Beide Projektleiter konnten durch das Projekt erstmals Erfahrungen mit Projektmanagement und der Organisation von einem Entwicklungsteam sammeln. Des Weiteren konnten sie ihre bisherigen Erfahrungen als Mitarbeiter in Projekten an die noch unerfahrenen Projektmitarbeiter weitergeben.

Die Entwickler konnten neben dem implementierungsspezifischem Know-how auch die Zusammenarbeit mit anderen Entwicklern erlernen.

So kann das Softwareprojekt „Digitale Schaltungssimulation im Browser“ als erfolgreich für alle Projektmitarbeiter betrachtet werden.

6 Glossar

Begriff	Erläuterung/Erklärung
GOLDi	Grid of Online Labs Ilmenau
BEAST	Block Diagram Editing And Simulating Tool
simcir.js	Frei verfügbare JavaScript-Bibliothek, welche es ermöglicht Komponenten zu Schaltungen zu verknüpfen und diese zu simulieren.
Komponente	Ein Baustein zur Erstellung von Blockschaltbildern, z.B. AND, OR, etc.
Leitung	Verbindung zwischen zwei Komponenten
Bibliothek	Übergeordnete Struktur, welche Komponenten beinhaltet
Projekt	Struktur aus einem Blockschaltbild, Basisbibliotheken und spezifischen Bibliotheken
Basisbibliotheken	Bibliothek mit Grundkomponenten, Bibliothek mit zusammengesetzten Grundkomponenten und eine Bibliothek zur Ablage von neu-erstellten Komponenten, welcher keiner spezifischen Bibliothek zugeordnet sind.
responsive design	Die Oberflächenanordnung und Größenverhältnisse verhalten sich relativ zur Größe des übergeordneten Elementes
Shortcut	Tastenkürzel oder Mausaktion zur schnellen Betätigung von Funktionen
FSM	Finite State Machine – Zustandsautomat
Tortoise	Anwendung zur Versionsverwaltung von Dateien
Slack	Gruppenorientierte Kommunikationsanwendung
Webstorm	Von Jetbrains entwickelte JavaScript-Entwicklungsumgebung
CSS	*Cascading Style Sheets*, Sprache zur einheitlichen Darstellung von Oberflächenelementen.
fancytree.js	Freie Bibliothek, welche eine Baumansicht implementiert.
MockUp	Vorlage zur Erstellung von Benutzeroberflächen, skizzenhafte GUI-Darstellung
SVG	Skalierbare Vektorgrafik

7 Anhang / Ressourcen

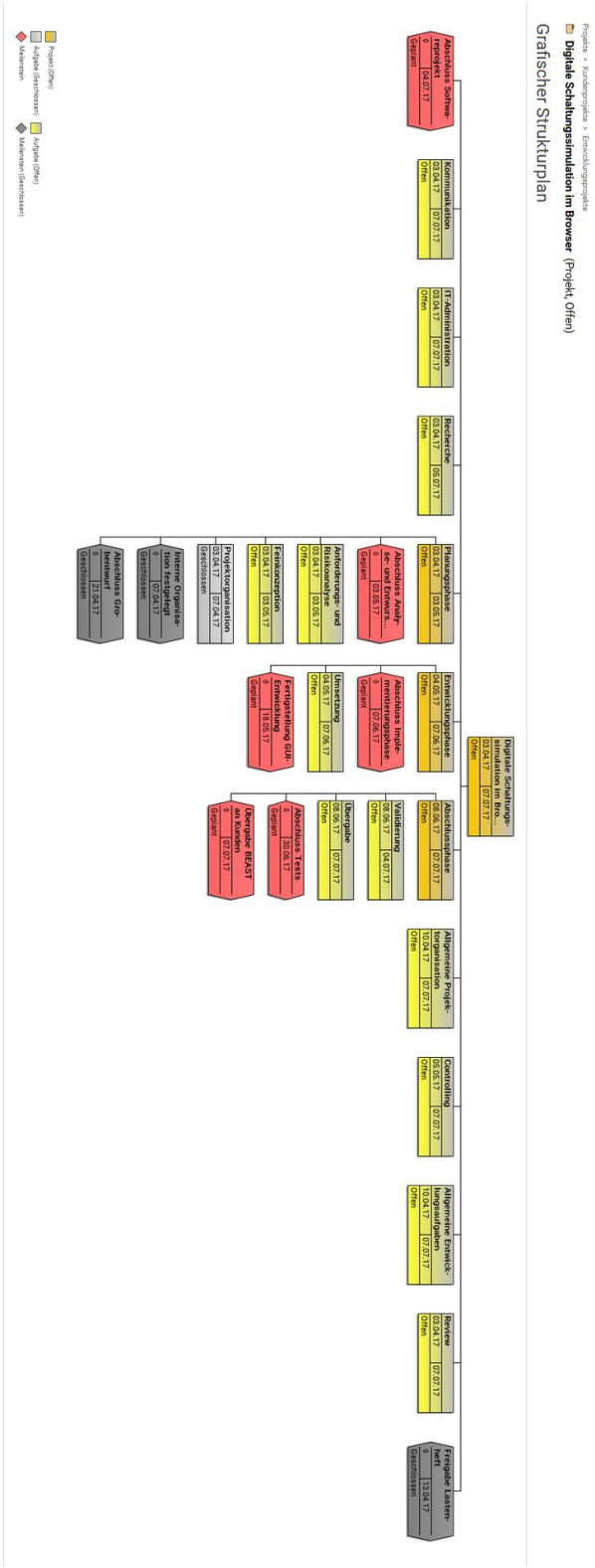

Abbildung 15: grafischer Strukturplan (Stand 01.05.2017)

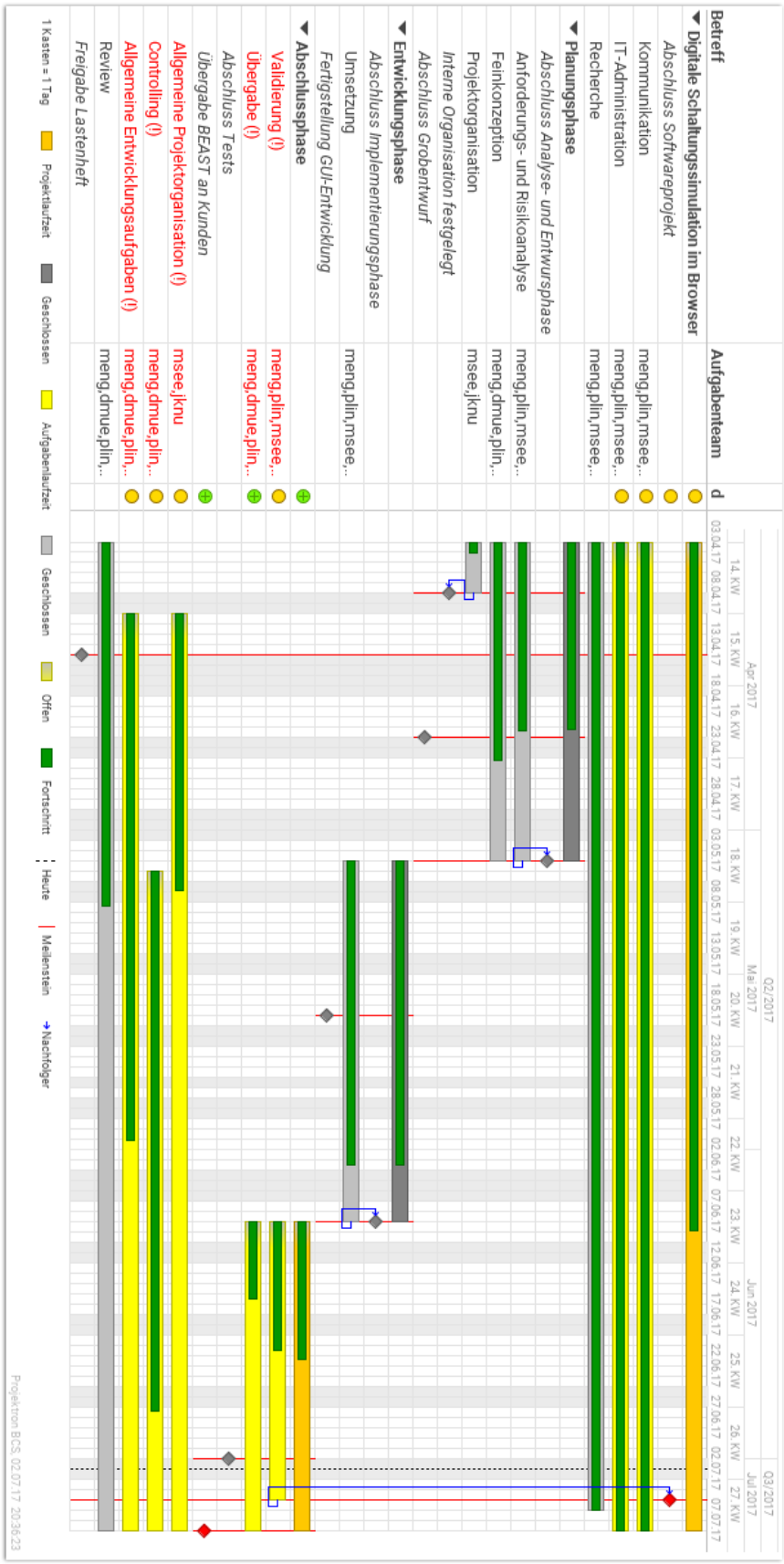

Abbildung 16: GANTT Diagramm

8 Abbildungsverzeichnis

Abbildung 1: Begleitaufgaben Aufwände 10
Abbildung 2: phasenspezifische Aufwände 10
Abbildung 3: Workflow einer Iteration 12
Abbildung 4: Meilensteine 13
Abbildung 5: simcir.js vorhandene Funktionalität 15
Abbildung 6: MockUp Benutzeroberfläche 17
Abbildung 7: Implementierungsansätze 19
Abbildung 8: Klassendiagramm BEAST-Daten 20
Abbildung 9: Datenstruktur BEAST 21
Abbildung 10: MVC-Struktur BEAST 23
Abbildung 11: BEAST-Oberfläche 25
Abbildung 12: Begleitaufgaben Aufwände 27
Abbildung 13: Phasenaufgaben Aufwände 28
Abbildung 14: Ticketstatistik 29
Abbildung 15: grafischer Strukturplan (Stand 01.05.2017) 32
Abbildung 16: GANTT Diagramm 33