UNIKASSEL
VERSITÄT

Fachbereich Elektrotechnik/Informatik
Fachgebiet Software Engineering

Toolgestütztes Event Storming für das Requirements Engineering

fulibWorkflows

BACHELORARBEIT

zur Erlangung des Grades eines Bachelor of Science
im Fachbereich Elektrotechnik/Informatik
der Universität Kassel

Eingereicht von:	Maximilian Freiherr von Künßberg
Anschrift:	Mönchebergstraße 31 34125 Kassel
Matrikelnummer:	33378673
Emailadresse:	maximilian-kuenssberg@uni-kassel.de
Vorgelegt im:	Fachgebiet Software Engineering
Gutachter:	Prof. Dr. Albert Zündorf Prof. Dr. Claude Draude
Betreuer:	M. Sc. Sebastian Copei
eingereicht am:	21. Februar 2022

Inhaltsverzeichnis

Listings

Abbildungsverzeichnis

Tabellenverzeichnis

1 Einleitung

1.1 Motivation

Ach ja das Requirements Engineering in der agilen Softwareentwicklung ist und bleibt ein leidiges Thema. Dafür wurde von Alberto Brandolini das Event Storming ins Leben gerufen. Namensvetter Albert dachte sich: "Ja, geil - Hab ich Bock drauf." So begann er es zu erweitern und nun sind wir alle hier und schauen uns das an.

1.2 Ziele

Noch kurz warum Event Storming eine gute Idee ist und hilfreich in der Anforderungsanalyse sein kann. Natürlich schon mal kurz ansprechen, dass für Albert(o)s Event Storming eine Beschreibungssprache entwickelt wurde und man diese zum einfachen Bedienen mit einem Web-Editor versehen hat.

1.3 Methodik

Expertengespräch zum Prüfen der gesetzten Ziele. Was ist das und warum ist es für die Ziele ausreichend?

1.4 Existierende Konzepte

Oldschool alles auf papier -> keine mockups direkt mit framework Web Editoren wie Miro für das Erstellen von Boards.

1.5 Aufbau der Arbeit

Zuerst grundlegende Ideen und Technologien. Anschließend die Implementierung beschreiben. Evaluation mittels Expertengespräch mit Adam Malik. Fazit bezüglich Ziele und outcome. Ausblick für das Tool.

2 Grundlagen

2.1 Event Storming

Was ist das, warum ist das cool dies das Kontrabass.

2.1.1 Allgemein

Beschreiben was die Grundideen von Brandolini sind. Wo spielt da das DDD (Domain Driven Design) rein?

2.1.2 Erweiterung

Was haben Albert und ich uns dazu gedacht um es nicht nur als Tool für die Wirtschaft, sondern auch in der Lehre verwenden zu können. Auch hier könnte man schon mit hereinbringen, dass Albert das ganze als Ablaufsbeschreibung in Microservices verwendet hat -> Workflow raussuchen und mit youtube video belegen.

2.2 Technologien

2.2.1 fulibWorkflows

Jaaaaa, das ist die Java Library. Ganz geiles Teil eigentlich jetzt wo der Code auch lesbar ist.

2.2.1.1 ANTLR4

Geiles Ding. Aber um Albert noch mal zu zitieren: “ANTLR ist so zickig wie ‘ne Bitch im Jungle Camp”.

2.2.1.2 JSON-Schema

Ziemlich geil um ehrlich zu sein. Man kann auch yaml damit definieren.

Erlaubt es die Eingaben des Nutzers zu beschränken und hints zu geben, was man machen darf und was nicht. Bereitstellen geht easy über schemastore.org. Wird dabei auch von vielen IDEs benutzt. Intellij direkt out-of-the-box, VSCode auch nachdem man die YAML Redhat Extension installiert hat.

2.2.1.3 fulibTools

Dank fulibTools ist auch fulib mit drin. FulibTools ist zur Generierung von Objektidiagrammen genutzt worden. Fulib (Bei FulibTools mit integriert) zur Generierung von Klassendiagrammen.

2.2.2 fulibWorkflows Web-Editor FE

Da brauchte es etwas mehr als beim BE. Die Entscheidungen für die verwendeten Technologien im Frontend wurden aufgrund der Idee der Integrierung vom Editor auf fulib.org getroffen.

2.2.2.1 Angular

Wir kennen es. Wir lieben es.

2.2.2.2 Bootstrap

Alles für den Dackel, alles für den Club unser Leben für die schön gestylten FEs. Simpel, oder? Ja, okay. Man nutzt dann auch ng-bootstrap für Angular Anwendungen. Natürlich

auch noch bootstrap-icons. Will ja niemand traurig machen und von Adrian verdroschen werden.

2.2.2.3 Codemirror

Schönes Ding. Ngx-codemirror ist es dann speziell für eine Angular Anwendung geworden. Eigentlich alles out of the box benutzt.

2.2.2.4 Angular-split

Find ich schon gut zu erwähnen. Ohne die geile dependency wäre das FE nie so pornös geworden.

2.2.2.5 file-saver

Weitere erwähnenswerte dependency. Wird genutzt um Dateien, die vom BE kommen, auch herunterladen zu können.

2.2.3 fulibWorkflows Web-Editor BE

Yeey alle Technologien die ich im Backend benutzt habe.

2.2.3.1 Spring Boot

Framework mit dem man easy mal ein backend generiert bekommt. Durch Java und viele Dependency allerdings alles andere als ein leichtgewicht.

Dennoch musste ein Java backend her, da sonst fulibWorkflows nicht hätte integriert werden können. Jedenfalls nicht ohne noch mehr middle ware.

Zudem hatte ich im Praktikum mit Spring Boot erfahrungen gesammelt. Die Verwendung von Annotations und dem aufsplitten zwischen Controller und Service ist mir bereits durch Nest.js bekannt gewesen.

Dennoch muss man sagen, dass durch die von Spring Boot bereits integrierten libraries nichts weiter außer fulibWorkflows hinzugefügt werden musste. Immerhin umfasst der Code vom Backend vielleicht 300 Lines of Code.

2.2.4 Deployment

Ein Web Editor will natürlich für alle erreichbar sein. Und fulibWorkflows muss auch irgendwo bereitgestellt werden, damit es das Backend und alle anderen interessierten benutzen können.

2.2.4.1 MavenCentral

MavenCentral ein wirklicher Hussarones was das publishen angeht. Glücklicherweise ist fulibWorkflows Teil der Fujaba Tool Suite, wodurch die benötigten Zugriffsrechte bereits vorhanden und andere Libraries bereits gepublished wurden. Hierdurch war es recht schnell möglich mit dem zuvor erworbenem Wissen fulibWorkflows zu publishen.

2.2.4.2 Heroku

Der Web-Editor soll immer erreichbar sein. Dies ist durch Heroku nur bedingt möglich. Heroku bietet allerlei Möglichkeiten verschiedenste Anwendungen bereitzustellen. Auch mit einem kostenlosen Plan ist es ohne Probleme möglich solch kleine Anwendungen bereitzustellen.

FE Deployment war easy, auch wenn ich erstmal wieder in eins meiner früheren Projekte gucken musste. BE Deployment war kniffliger, doch man ist nie der erste der eine Spring Boot application auf Heroku deployen will. Daher Tutorial reingefahren und ab ging der gebutterte Lachs.

3 Implementierung

Hier bin ich mir tatsächlich unsicher was alles reinsoll. Ich teile es auch erstmal in drei auf.

3.1 fulibWorkflows

Was gibt es hier so interessantes zu sehen? Gehen Sie weiter.

3.1.1 fulibWorkflows Grammatik

Beschreiben der ANTLR4 Grammatik für fulibWorkflows

Und natürlich auch wie mir das beim parsen der yaml geholfen hat. Dat wird ne lange Sektion.

3.1.2 Generierung dank fulibTools

FulibTools gibt gute Anbindung an Graphviz, wenn man sowieso schon mit fulib arbeitet.

3.1.3 schema

Da hab ich lange dran gesessen. Und ich glaube, selbst jetzt ist es kein gutes Schema. Allerdings tut es was es soll.

3.1.3.1 Mockups

Eigenes Datenmodell gebaut. Daraus die wichtigsten Infos gezogen. Dank StringTemplates von antlr richtig easy zu bauen. Gilt für Html als auch Fxml.

3.2 editor-frontend

Alles was es zum FE so gibt.

3.2.1 iframes

Naja der editor basiert einfach darauf, dass es iframes gibt. Könnte man auch weg lassen?

3.2.2 codemirror

Eingerichtet und los ging es. Noch einen eigenes kleines Hint Add on geschrieben. Feddig. Musste es erstmal simpel halten. Gibt noch genügend zukünftige Ideen.

3.2.3 angular split

Danke an Adrian, der mit dazu geraten hat.

Nachdem ich mit purem css da grids erstellt habe, stand ich vor dem Problem der Veränderung von Größen. Angular split löst das Problem auf eine wunderbare Art und Weise.

Die dreiteilung der View war damit wirklich einfach. Auch wenn man aufpassen musste beim Verändern der größen, wenn man iframes benutzt. Da musste ein kleiner Fix rein, der aber auch bereits von den machern vorgegeben war.

3.3 editor-backend

Das wird wieder kurz.

3.3.1 Spring booterino

Gabelstablinski hier drüben war dank ein paar Annotations schnell erstellt und macht bisher keine Probleme.

Die zip Datei wird im BE zusammengebaut. Dafür gibt es schon alles fertig in java.util.zip. Da hab ich nach gegoogelt und alles lief wie von selbst.

Na gut, ich muss zugeben, dass die zip vom BE ans FE schicken und dann richtig runterladen können, ohne das mir alles um die Ohren fliegt schon einen Arbeitstag gebraucht hat. Manchmal muss man lediglich seine Dummheit für einen Moment ablegen.

4 Evaluation

Eigentlichen Anwendugszweck beschreiben

Event Storming nachvollziehbar für alle machen. Htmls einfach noch mal öffnen.

Besser mit Kunden über das Layout sprechen können -> Mockups mit grundlegendem Styling Kunden können einen Workflow mal durchklicken -> Next prev page

4.1 Expertengespräch

Gespräch mit Adam. Fragenkatalog erstellen. Alles aufzeichnen, wenn Adam zustimmt auch Bild und Ton -> Youtube bereitstellen nicht gelistet

4.2 Auswertung

Was ist die Meinung des Experten zu der Anwendung?

Kann ihm das helfen? (Begründung durch seine Vorerfahrungen)

5 Fazit

Machen die Erweiterungen Sinn?

Kann der Editor von Leuten verwendet werden, die nur wenig Programmier erfahrung haben?

Hilft es beim RE in der Wirtschaft?

Füllt diese Anwendung eine bestehende Lücke im RE? Wurde das zuvor gesetzte Ziel erreicht? Ist die grundlegende Idee gut, aber die Umsetzung nicht ausreichend durchdacht?

Alberts Drang immer wieder neues zu haben xD

6 Ausblick

Was hatten die Leute zu meckern und sollte daher umgesetzt werden?

Was ist mir selbst an Ideen gekommen?

fulibWorkflows ist Teil von Fulib und sollte daher auch über fulib.org erreichbar sein. Alles zusammen an einem Ort macht mehr Sinn als alles verstreut zu haben.