hochschule mannheim

Fakultät Informationstechnik
Studiengang Technische Informatik

Bachelorarbeit

Arbeitstitel: Anforderungen und Testen

Florian Brunotte

Abschlussarbeit

zur Erlangung des akademischen Grades

Bachelor of Science

Vorgelegt von	Florian Brunotte
am	30.01.2021
Referent	Prof. Dr. Ing. Mark Hastenteufel
Korreferent	Prof. Dr. Martin Damm

Schriftliche Versicherung laut Studien- und Prüfungsordnung

Hiermit erkläre ich, dass ich die vorliegende Arbeit selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Mannheim, 30.01.2021 _____

Florian Brunotte

Zusammenfassung

Ziel dieser Bachelorarbeit war die Erstellung einer leichtgewichtigen Software mit der Requirements und Testcases aufgenommen werden können und durch Testruns validiert werden. (Unterschied Validierung und Verifikation?) Eingesetzt wird diese Software im Rahmen von kleinen Semesterprojekten an der Hochschule Mannheim. Geschrieben wurde sie in als Webapplikation in Python und Django während die Visualiisierung und das User Interface durch HTML und CSS erstellt wurde.

Inhaltsverzeichnis

Kapitel 1

Motivation - Warum wurde diese Software geschrieben?

Es ist vorzuziehen, dass...

Motivation: Die Zielgruppe für diese Arbeit und für diese Software sind die Hochschulangehörigen, die innerhalb des Studiums Anforderungen und Tests verwalten müssen.

Es soll eine leichtgewichtiges Softwaretool entwickelt werden zum Anforderungs- und Testmanagement. Dabei sollte man sich auf die Kernfunktionalitäten beschränken, damit man sich auf das Wesentliche konzentrieren kann, dem Schreiben von Anforderungen und dem Ausführen von Tests. Andere Softwaretools, die am Markt verfügbar sind, sind für den Einsatz in der Lehre nicht geeignet, da sie zu komplex sind und auch Geld kosten können. Sollte man nicht bezahlen, werden bestimmte Funktionen gesperrt. Insgesamt sind sie zu schwergewichtig und man verliert das eben beschriebene wesentliche aus dem Blick. Eine intuitive und simple Software, wie „Anforderungen und Testen (WIP)" soll hierbei Abhilfe schaffen.

Hier werden dann noch manche Tools aufgezählt. Vielleicht mit Bilder die Limitierung und die Überfülle zeigen.

Kapitel 2

Anforderungsanalyse - Welche Anforderungen hat das Projekt?

Zuerst hat Herr Hastenteufel ein Dokument mit Anforderungen gesendet. Nach Absprache per Mail wurden noch folgende Sachen geklärt:

- Die Kategorie für die Requirements werden erstmal als einfache Kategorien wie 1, 2, 3 realisiert. Später vielleicht Auswahl aus der Art der Requirement: Stakeholderanforderung, aber dazu nochmal in SOE reingucken
- Die Namen Requirement, Testcase und Testrun werden benutzt, um alles einheitlich zu lassen
- Da der Admin und der Professor als Nutzer sehr ähnlich sind, wurden die beiden Rollen mit ihren Anforderungen zusammengelegt als Professor

Ein Klassendiagramm und ein Zustandsdiagramm wurden erstmal nicht gemacht, da sie nicht für sinnvoll gehalten wurden. Aber das wurde nochmal nachgefragt.

Es wurden ein Use Case Diagramm skizziert, mehrere Activity Diagramme, die die Use Cases verfeinern skizziert und die UIs wurden auch skizziert. Daneben wurde auch das ER-Modell schon in Draw.io gezeichnet und die Tabellen daraus wurden transformiert. Somit hat man für den Prototyp bereits ein Datenmodell mit dem man testen kann. Die Diagramme werden sich wahrscheinlich noch ändern im Verlauf des Projekts, aber für den Anfang sind sie in den Abbildungen 1, 2, 3... zu sehen. Das war der Start des Projekts.

Es wurden ebenfalls erste Skizzen zu den UIs vollzogen und abgesprochen mit dem Kunden, also Herrn Hastenteufel. Die Rückmeldungen waren:

> Documentation is like sex: when it is good, it is very, very good; and when it is bad, it is better than nothing.
>
> *Dick Brandon*

Kapitel 3

Implementierung - Wie erstellt man eine Django-Applikation?

Mit PyCharm ein neues Projekt starten. Dabei wird ein neues virtual Enviroment gebildet in dem man alles installieren kann, das hat aber keine Auswirkungen auf andere Installationen, wie die Hauptinstallation von Python. Neuer Ordner wurde gebildet in dem Bachelorarbeit Ordner für den Prototyp. Als nächstes sollte man Django installieren. Im Terminal von PyCharm kann alles gemacht werden, wie Django installieren. Mit Pip install django kann man dann django installieren

Dann einfach das Tutorial machen von Django Documentation.

Django entstand in einem Nachrichtenumfeld

Man startet ein neues Django Project mit django-admin startproject name, dabei sollte der Name nicht wie django oder test heißen, das könnte zu Problemen führen cd ./anforderungenundtesten um in den richtigen Ordner zu kommen, der die manage.py Datei enthält

über python manage.py runserver kann der Developerserver aufgerufen werden und man sieht, dass die Installation geklappt hat. Man kann hierbei auch die IP Adresse so ändern, dass alle Geräte im gleichen Netzwerk Zugriff haben.

Man erhält verschiedene Dateien beim Start eines Projects, die Funktionen der einzelnen Dateien sind in der Tabelle 111 dargestellt.

Es gibt dabei einen Unterschied zwischen einem Projekt und einer App in Django, so kann ein Projekt mehrere Apps enthalten und eine App kann in verschiedenen Projekten enthalten sein. Man sollte auf die Wiederverwendbarkeit achten.

Hierbei kam der Begriff Boiler Code auf, der Codeabschnitte beschreibt, die wenige Änderungen haben und an vielen Stellen auftauchen können. Beispiele sind in den

Listings 1 und 2 dargestellt. Man sieht in dem 1. Lisitng ein Shebang für die Programmiersprache Python, damit wird dem Compiler klar das es sich hier um Python Code handelt. Im 2. Lisiting sieht man die Basis von einem HTML Dokument. Mit diesem Template können viele verschiedene Websites generiert werden, aber diese Tags sind notwendig.

Die App muss unter den installierten Apps angemeldet werden

Danach sollten die URLs definiert werden und zusammengeführt werden. Es gibt eine url Datei für das Projekt und eine für die App. Von dem Projekt kann man auch die Grundseite umleiten lassen auf die Seite der App. Die verschiedenen Routen der Seiten sind in der Abbildung 1 dargestellt.

Es müssen auch die Befehle makemigrations und migrate angewandt werden, da sie sämtliche Änderungen an den Datenbanken und den Modellen ausführen.

Zuletzt wird mit runserver der Development Server erstellt und zum Laufen gebracht unter der Adresse 00000 kann man dann lokal die App testen oder man kann auch für alle Geräte im gleichen Netz die App verfügbar machen. Das ist nützlich, um die Webseiten auf mobilen Geräten oder generell Geräten mit verschiedenen Displaygrößen zu testen.

Die Datenbank, in diesem Fall Postgres muss auch eingerichtet werden, dazu gibt es in settings.py die Einstellungen zu Databases. Diese müssen verändert werden, damit das Django-Projekt mit einer Postgres Datenbank funktioniert.

Danach können die Modelle definiert werden. ORM von Python ist hierbei wichtig. Die Modelle werden aus den Tabellen, die wiederrum aus den ER-Modellen hergeleitet werden, erstellt.

Die ER-Modelle sind in der Abbildung 1 dargestellt....weiter erklären. Die Modelle sind in Abbildung 1 dargestellt... auch weiter erklären.

Nachdem man die Modelle geschrieben hat muss man das Modul noch für den Admin zur Bearbeitung freischalten bzw. registrieren. Der Admin oder Superuser muss auch erstellt werden.

Modelle in Django verfolgen die DRY Regel, also man sollte sich nicht wiederholen.

Nachdem die Modelle erstellt/geschrieben wurden, kann man mit makemigrations und migrate die Tabellen/Relationen in die Datenbank einspielen. Mit makemigrations werden die migrations erstellt als Datei um noch mal drüber zu gucken. Dabei konnte ich auch den Fehler mit dem Default DateTimeField ausbessern. Da die Migrationen nacheinander gemacht werden. Jetzt kann man bereits über den Admin die Daten einpflegen. Dazu ist zu sagen, dass das bis jetzt nur der Admin kann und die Admin Seite zwar Funktional ist, aber im Sinne von UI/Aussehen noch verbessert

werden kann.

Das UI wird aber später kommen, da man erstmal einen Prototyp haben möchte der die Grundlegenden Funktionen einer Datenbank hat: Daten eintragen, löschen, ändern und anzeigen.

Kapitel 4

LaTeX-Sachen

4.1 Latex Sachen als Section

Die Ziele des Templates sind wie folgt:

- Beispiele der typischen Verwendung von LaTeX und dessen Erweiterungen geben, die viele im Rahmen von Abschlussarbeiten üblichen Anforderungen abdeckt.
- Nahe am LaTeX-Standard halten mit wenigen weit verbreiteten Erweiterungen, um problemlosen Einsatz und Erweiterbarkeit sicher zu stellen.
- Die Einhaltung der Formalien an der Hochschule Mannheim in der Fakultät Informationstechnik vereinfachen.

~~durchgestrichen~~ angezeigt.

Nehmen Sie Kapitel 1 nicht mit dazu. Schreiben Sie Inhalte und keine Leerphrasen. Verwenden Sie nicht das „nächste“ oder „folgende“ Kapitel sondern immer als Zahl das wievielte Kapitel. Verlassen Sie sich auf LaTeX und nummerieren Sie nie selbst sondern referenzieren Sie. Jedes Kapitel außer das Erste muss vorkommen. In Kapitel 4 führen wir in die LaTeX-Umgebung kurz ein und geben eine Übersicht über die Tools, die notwendig sind ein Dokument zu erstellen. In Kapitel **??** stellen wir das Layout sowie einige Idiome zum Textsatz mit LaTeX vor. In Kapitel **??** besprechen wir das Einbinden und Erstellen von Fließobjekten wie Bilder, Tabellen und Listings. Hinweise zum Schreibstil, mathematischem Formelsatz und zur Literatur sind in Kapitel **??** gesammelt. Abschließend fassen wir in Kapitel 5 die Vorteile und Features, hervorzuheben sind die gute Qualität und Satz, von LaTeX für Ihre Abschlussarbeit noch einmal zusammen.

Plattform	LaTeX-Distribution	Editor
Linux/Unix	TeX Live	Texmaker, Emacs
MacOSX	TeX Live, MacTex	Texmaker, TeXShop
Windows	TeX Live, MiKTeX	Texmaker, TeXstudio

Tabelle 4.1: LaTeX-Distributionen und Editor je Plattform

in Tabelle 4.1.

(zum Beispiel `thesis.tex`)

(Times New Roman) When Apollo Mission Astronaut Neil Armstrong first walked on the moon, he not only gave his famous "one small step for man, one giant leap for mankind" statement but followed it by several remarks, usual communication traffic between him, the other astronauts and Mission Control. Just before he re-entered the lander, however, he made this remark *Good luck Mr. Gorsky*.

(Helvetica) Many people at NASA thought it was a casual remark concerning some rival Soviet Cosmonaut. However, upon checking, there was no Gorsky in either the Russian or American space programs. Over the years many people questioned Armstrong as to what the *Good luck Mr. Gorsky* statement meant, but Armstrong always just smiled. On July 5, 1995 in Tampa Bay FL, while answering questions following a speech, a reporter brought up the 26 year old question to Armstrong. This time he finally responded. Mr. Gorsky had finally died and so Neil Armstrong felt he could answer the question.

(Palatino) When he was a kid, he was playing baseball with a friend in the backyard. His friend hit a fly ball, which landed in the front of his neighbor's bedroom windows. His neighbors were Mr. & Mrs. Gorsky. As he leaned down to pick up the ball, young Armstrong heard Mrs. Gorsky shouting at Mr. Gorsky. *Oral sex! You want oral sex?! You'll get oral sex when the kid next door walks on the moon!*

Machen Sie Fußnoten[1] immer ohne einleitendes Leerzeichen und innerhalb des Satzes, also nie nach einem Punkt. Fußnoten sind ganze Sätze mit Satzzeichen. Fußnoten sind Inhalte, die nicht für das Verständnis notwendig sind[2]. Juristen verwenden Fußnoten zur Quellenangabe. Wir sind keine Juristen und distanzieren uns (nicht nur) von dieser Praxis deutlich. Setzen Sie Fußnoten sehr sehr sparsam ein.

Referenzieren Sie innerhalb des Dokuments, zum Beispiel auf das Kapitel **??** in dem es unter anderem um Bilder geht und das auf Seite **??** los geht, mit `\ref` (meistens) oder `pageref` (sehr selten). Verwenden Sie vor dem Befehl zum Referenzieren immer ein ~. Das ist ein nicht umbrechbares Leerzeichen und ~~Kapitel~~
~~1~~, also der Zeilenumbruch vor der Nummerierung, wird vermieden.

~~Es macht keinen Sinn aus irgendwelchen Gründen~~

[1] Das ist eine Fußnote.

[2] Fußnoten haben übrigens nichts mit Noten oder Musik zu tun.

~~erscheinen sie noch so sinnvoll~~
~~Zeilenbrüche im Fließtext einzuführen.~~
~~Sie~~
~~wollen eigentlich etwas anderes.~~

Sie sollten Abkürzungen (AKÜ) bei ersten Vorkommen definieren. Schreiben Sie das Wort zuerst aus und dann die Abkürzung in Klammern. Danach können Sie die AKÜ verwenden. Meistens sollten Sie jedoch auf Abkürzungen verzichten. Schreiben Sie lieber *beispielsweise, zum Beispiel, und so weiter, beziehungsweise* statt *bspw., z.B., usw., bzw.*.

Setzen Sie die drei verschiedenen Bindestriche -, – und — richtig ein. Der einfache Bindestrich - wird bei Worttrennungen, wie AKÜ-Fimmel, eingesetzt (im Quelltext mit `-`). Der etwas längere Streckenstrich – wird bei Streckenangaben, wie die Strecke Mannheim–Karlsruhe oder von 10:00–11:45 eingesetzt (im Quelltext mit `--`). Der Gedankenstrich — ist bei Einschüben — wie zum Beispiel hier — einzusetzen (im Quelltext mit `---`). ~~Es ist auf keinen Fall ein Leerzeichen um einen Binde - strich oder einen Strecken -- strich und immer ein Leerzeichen um einen Gedanken --- strich~~.

Name	Adresse	Wohnort	Telefon
Susi Sinnlos	Eichenstrasse 5	12345 Unterstadt	24927749242
Horst Kurz	Schnellweg 17	42420 Rapid	999
Jochanaan Leuchtentrager	Hochstraße zu	666 Hell	1-800-33845

Tabelle 4.2: Adressliste

Die entsprechenden vordefinierten Umgebungen heißen `table` für Tabellen und `figure` für Abbildungen. Mit den optionalen Argumenten `htbp`, das steht für *here, top, bottom, page*, geben Sie LaTeX den Tipp, dass Sie am liebsten das Fließobjekt *hier* an dieser Stelle haben möchten. Wenn das nicht geht, dann eben am *Anfang* der Seite, und wenn das nicht geht (weil es zum Beispiel ein Kapitelanfang ist) ans *Ende* der Seite. Wenn das alles nicht klappt, dann halt auf eine Extra-*Seite*. Beherzigen Sie folgende Tipps zu Fließobjekten:

- Jede Tabelle, jedes Bild und jedes Listing ist ein Fließobjekt.
- Zentrieren Sie Bilder und Tabellen.
- Jedes Fließobjekt hat eine Bildunterschrift (Caption) mit einem Label und wird im Text passend referenziert.

Schreiben Sie nie, ~~wie man unten in der Tabelle sehen kann~~, da Sie nie wissen und auch nicht wissen sollen, ob die Tabelle wirklich „weiter unten“ ist. Verwenden Sie statt relativer Positionsangaben Referenzen mit Zahlen, die Sie durch das Label

erhalten, wie zum Beispiel „wie sie in Tabelle 4.2 sehen können“. Verwenden Sie kurze und prägnante Bildunterschriften, die nicht länger als eine Zeile lang sind. Alles was mehr als eine Zeile hat gehört in den Fließtext. Sie sollten für die Fließobjekt Caption keinen Satz bilden und daher auch keinen Punkt am Ende haben. Die Caption ist eine Unterschrift und gehört unter das Fließobjekt.

Es ist möglich, wenn auch nicht empfohlen, Bilder an den Rand einer Seite zu klatschen, wie wir das mit dem GNU-Logo in Abbildung 4.1 gemacht haben. Das ganze ist ein netter Effekt für Graphiken, wie zum Beispiel ein Logo, die nicht zum Verständnis des Texts gebraucht werden und wenig Details aufweisen. Der Effekt sollte aber nicht überstrapaziert werden, 1–2 Mal je Abschlussarbeit sollte, wenn überhaupt, reichen. Außerdem funktioniert `wrapfigure` nicht immer sehr stabil.

Abbildung 4.1: GNU-Logo [9, 13]

Bitte nehmen Sie **nie** JPG oder PNG für Vektorgrafiken, also Zeichnungen mit Linien oder anderen geometrischen Objekten, sondern ausschließlich PDF. Binden Sie also **nie** Vektorgraphiken verpixelt ein.

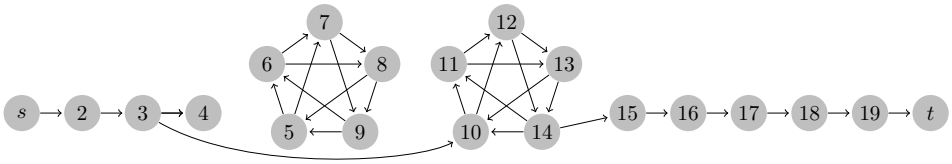

Abbildung 4.2: Automaten mit tikz [8]

Neben langen Listings sind natürlich kurze prägnante Listings in Pseudocode (oder Python ;-) viel angenehmer, wie in Listing 4.1 der effiziente GGT.

```
1 def ggt(x, y):
2     while x != 0:
3         x,y = y%x, x
4     return y
```

Listing 4.1: ggT — kurz und gut

Die Parameter für Listings sollte man für das ganze Dokument gleich lassen. Wenn man mal unbedingt wechseln will, dann ist das auch möglich, wie zum Beispiel bei Listing 4.2, das den ggT in Java mit einem anderen Zeichensatz zeigt.

```
public static int ggt(int x, int y) {
    while (x != 0) {
        int h = x;
        x = y%x;
        y = h;
    }
    return y;
}
```

Listing 4.2: ggT — Java

Der verwendete serifenlose Zeichensatz sieht vielleicht schöner aus, aber der variable Zeichenabstand kann bei Listing störend sein. Der in den Beispielen Listing **??** und Listing 4.1 verwendete Zeichensatz mit festem Zeichenabstand ist für Quellcode meist zu bevorzugen. Wir können natürlich auch C++-Quellcode setzen, bei Listings LaTeX in den Kommentaren verwenden und Listings aus Dateien einlesen wie in 4.3.

```
1 int gcd(int x, int y) { // greatest common divisor
2    while (x != 0) { // x ≠ 0
3       int h = x; // prepare swap
4       x = y%x;
5       y = h;
6    }
7    return y;
8 }
```

Listing 4.3: gcd — C/C++

Idealerweise verwenden Sie spätestens jede zweite Seite ein Bild. Ein Bild lockert auf und „sagt mehr als tausend Worte“. Vermeiden Sie Aufzählungen.

Eine Formel kann im Fließtext integriert sein, wie zum Beispiel $\sum_{i=1}^{n} i = \frac{n\cdot(n+1)}{2}$ oder separat und referenzierbar gesetzt werden wie die Folgende:

$$\sum_{i=1}^{n} i = \frac{n \cdot (n+1)}{2} \tag{4.1}$$

Im LaTeX-Quelltext werden beide Arten von Formeln gleich geschrieben aber anders gesetzt. Achten Sie zum Beispiel auf das Summenzeichen und die Positionen des Index. Die Gleichung 4.1 ist natürlich vom Fließtext aus referenzierbar. Man kann auch schreiben: (4.1) ist natürlich vom Fließtext aus referenzierbar. Im Fließtext kann man auch gerne auf den Bruch verzichten und $\sum_{i=1}^{n} i = (n \cdot (n+1))/2$ schreiben, was meist etwas lesbarer ist. Alternativ geht auch $\sum_{i=1}^{n} i = ½ \cdot n \cdot (n+1)$. Achten Sie

bei Formeln darauf als Multiplikationszeichen $\cdot$ zu verwenden und nicht $*$. Ich kenne einen Kollegen, der ansonsten dadurch sehr erregt wird.

Sie können viele Symbole, wie die griechischen Buchstaben $\alpha, \beta, \gamma, \ldots$; logische Symbole wie $\forall, \exists, \nexists, \wedge, \vee, \neg, \Rightarrow, \Leftrightarrow$; Mengensymbole wie $\in, \cup, \cap, \subseteq, \not\supset, \uplus, \ldots$; andere Symbole $\rightarrow, \sqsubseteq, \sim, \models, \vdash, \infty, \emptyset, \mathbb{N}, \mathbb{R}, \ldots$; oder zusammengesetzte Gleichungen wie die Definition der 91er-Funktion [4] verwenden.

$$f(x) = \begin{cases} x - 10 & \text{gdw} \quad x > 100 \\ f(f(x+11)) & \text{sonst} \end{cases}$$

Definition 4.1 *Sei $\varepsilon = 0$.*

Satz 4.1 *Für alle positiven ganzen Zahlen n gilt $\sum_{i=1}^{n} i = \frac{n \cdot (n+1)}{2}$.*

Beweis Vollständige Induktion:

- *Induktionsanfang ($n = 1$):* Es gilt

$$\sum_{i=1}^{1} i \;=\; 1 \;=\; \frac{1 \cdot (1+1)}{2}.$$

- *Induktionsschritt ($n \rightarrow n+1$):* Es gelte die Induktionsvoraussetzung (IV):

$$\sum_{i=1}^{n} i = \frac{n \cdot (n+1)}{2}$$

  Wir zeigen, dass dann auch

$$\sum_{i=1}^{n+1} i = \frac{(n+1) \cdot (n+2)}{2}$$

  gilt wie folgt:

$$\begin{aligned} \sum_{i=1}^{n+1} i \quad &= \quad (\sum_{i=1}^{n} i) + (n+1) \\ &=_{\text{IV}} \quad \frac{n \cdot (n+1)}{2} + (n+1) \\ &= \quad \frac{n \cdot (n+1)}{2} + \frac{2 \cdot (n+1)}{2} \\ &= \quad \frac{n \cdot (n+1) + 2 \cdot (n+1)}{2} \\ &= \quad \frac{(n+2) \cdot (n+1)}{2} \end{aligned}$$

□

Sie müssen den Dreisatz *Definition*, *Satz* und *Beweis* nicht verwenden, wenn Sie kein sehr formales Thema haben. Eine sehr formale Aufarbeitung von bekanntem Inhalt gefolgt von einem nicht so formalen eigenen Anteil sollte man meist vermeiden.

`http://www.csie.ntu.edu.tw/~cjlin/libsvm/`, sondern einen passenden publizierten Artikel zitieren.

```
$ bibtex thesis1
$ bibtex thesis2
```

Kapitel 5

Zusammenfassung und Ausblick

Abschließend kann man sagen, dass...

Literaturverzeichnis

[1] Chih-Chung Chang and Chih-Jen Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[2] Tilo Gockel. *Form der wissenschaftlichen Ausarbeitung*. Springer, Berlin, Heidelberg, 2008.

[3] Helmut Kopka. *LaTeX: Eine Einführung*. Addison-Wesley, Bonn, 4 edition, 1992.

[4] Zohar Manna and Amir Pnueli. Formalization of properties of functional programs. *J. ACM*, 17:555–569, July 1970.

[5] Oren Patashnik. BIBTEXing, 1988.

[6] Peter Rechenberg. *Technisches Schreiben: (Nicht nur) für Informatiker*. Hanser, München, Wien, 2006.

Online-Quellen

[7] ACM, Digital Library. `https://dl.acm.org/`. [letzter Zugriff: 9. Jan. 2018].

[8] TeXample.net, Automata and Petri nets examples. `http://www.texample.net/tikz/examples/automata-and-petri-nets/`. [letzter Zugriff: 9. Jan. 2018].

[9] The GNU Logo.png. `http://en.wikipedia.org/wiki/File:The_GNU_logo.png`. [letzter Zugriff: 9. Jan. 2018], Optimager commonswiki, 21:48, 20 October 2005, unter Free Art Licence.

[10] IEEE Xplore, Digital Library. `http://ieeexplore.ieee.org/Xplore/dynhome.jsp`. [letzter Zugriff: 9. Jan. 2018].

[11] Free Art Licence. `http://artlibre.org/licence/lal/en`. [letzter Zugriff: 9. Jan. 2018].

[12] Wikipedia. Citing wikipedia, the free encyclopedia. [letzter Zugriff: 9. Jan. 2018].

[13] Wikipedia. Wikipedia. [letzter Zugriff: 9. Jan. 2018].