Technische Hochschule Brandenburg

März 2017

Praktikumsbericht

von

Klaus Schwarz

über ein Praktikum bei der

ikusei GmbH

Zossener Str. 55-58
Aufgang D
10961 Berlin-Kreuzberg

Dauer:	02.01.17 - 26.03.17
Fachbetreuer:	Prof. Dr.-Ing. Sven Buchholz
Firmenbetreuer:	Daniel Molnar, M.Sc.
Studienbereich:	Informatik und Medien
Matrikelnummer:	20140006

Selbstständigkeitserklärung

Hiermit erkläre ich, dass ich die vorliegende Arbeit vollkommen selbstständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt, sowie Zitate kenntlich gemacht habe. Die Arbeit wurde in dieser oder ähnlicher Form noch keiner anderen Prüfungsbehörde vorgelegt.

Brandenburg/Havel, den 24.04.2017

Unterschrift

Inhaltsverzeichnis

Quellcodeverzeichnis **i**

1 Praktikumsstelle **1**

1.1 Firmenprofil 1
1.2 Abteilungsstruktur 2
1.3 Die Entwicklungsabteilung 2
1.4 Betreuung vor Ort 3
1.5 Auswahl des Praktikumsplatzes 3
1.6 Erfahrungen aus der Wahl eines Praktikumsplatzes 3
1.7 Bezahlung und Arbeitszeit 4

2 Aufgaben und Tätigkeiten **5**

2.1 Tätigkeitsumfeld 5
2.2 Aufgaben und Ziele 6
2.3 Tätigkeiten und Arbeitsergebnisse 7
2.3.1 Vorbedingungen zur Aufgabenstellung 7
2.3.2 Konkretisierte Aufgabenstellung 8
2.3.3 Lösungsweg der bestehenden Aufgabe 8
2.3.4 Bestehende Probleme 12
2.3.5 Aufgabenfazit 13

3 Einsichten und Fazit **14**

3.1 Technik 14
3.2 Methodik 15
3.3 Sonstiges 15
3.4 Fazit 16

Quellcodeverzeichnis

Listing 2.1 Installieren des Basissystems . . . 9

Listing 2.2 Erstellen der nötigen Devices . . . 10

Listing 2.3 Erstellen der fstab . . . 11

Listing 2.3 Erstellen der inittab . . . 11

Kapitel 1

Praktikumsstelle

1.1 Firmenprofil

Die ikusei ist auf die Umsetzung webbasierter Softwareprojekte spezialisiert und deckt dabei den gesamten Prozess von der Konzeption, Gestaltung bis zur Umsetzung ab. Die Agentur kümmert sich außerdem auch um den technischen Ausbau, unterstützt die Redaktion des Kunden und berät zu Themen wie Social Media, Suchmaschinen-Optimierung, Online Marketing und E-Mail-Marketing.

Technischen Schwerpunkt bildet dabei die serverseitige Programmierung mit „Ruby on Rails“. Die ikusei nutzt außerdem noch Frameworks wie nodeJS, ReactJS und für Frontend seitige Programmierung JavaScript.

Die Agentur ikusei wurde 2009 gegründet und ist über die Jahre zu einem erfolgreichen Netzwerk aus Spezialisten gediehen. Aus dem Schornstein einer ehemaligen Seifenfabrik in Kreuzberg steigen heute Ideen für z.B. die quirin bank, ulmen.tv, den Tagesspiegel und Mercedes-Benz über den Dächern Berlins auf. In dem offenem Büro in der Dachetage des Bürogebäudes, in dem die Firma ansässig ist, sitzen 5 Entwickler, eine Projektleiterin und Phillip Willimzig, der Chef der Berliner Agentur, die oft durch mehrere Freelancer in Bereichen wie Entwicklung, Werbetextung und Design ergänzt wird. Das Hauptaugenmerk der Firma liegt dabei auf opensource. Wann immer möglich

versuchen die Entwickler ihren Teil zu den genutzten Frameworks beizutragen und veröffentlichen Code, der zur Weiterentwicklung der Recourcen der Allgemeinheit dient. Dazu gehört auch das hauseigene Framework Goldencobra, das als Grundlage der meisten Auftragsarbeiten der ikusei dient.

1.2 Abteilungsstruktur

Die Agentur ikusei besteht grundsätzlich aus 3 Abteilungen. Dem Management, der Marketingabteilung und der Entwicklung. Während das Management die Tagesgeschäfte übernimmt, also die Telefonate entgegen nimmt, sich um die Abrechnungen kümmert und versucht, den schier unendlichen erzeugten Tickets Herr zu werden, ist es außerdem für die vielen Gäste, Meetings und Konferenzen der ikusei zuständig. Das Marketing ist ebenso beschäftigt. Nicht nur die Kundenakquise und die damit verbundenen Termine gehören zu den Aufgaben, sondern auch die teilweise recht aufwendige Kozeptionierung der Kampagnen der Firma und deren Kunden gilt es zu bewältigen. Ich werde Anfangs der Entwicklung unterstellt. Später arbeite ich eng mit der Abteilung zusammen, wobei sich mein Aufgabenbereich klar von dem der Entwicklung unterscheidet. Die Entwicklungsabteilung ist das Herz der Firma. Hier laufen Marketing und Management zusammen, um eine möglichst arbeitsfreundliche Atmosphäre zu schaffen.

1.3 Die Entwicklungsabteilung

Wie oben beschrieben ist dies das Herzstück der Agentur. Sie ist auf den ersten Blick sehr eigenwillig organisiert und die Paradigmen dieser Abteilung entsprechen nicht unbedingt derer, die man im Fach Softwareengeneering an der Hochschule in Brandenburg lernt. Hier wird das Agile Manifest gelebt. Aufgrund der Beschaffenheit der Projekte der ikusei und des recht kleinen Entwicklerteams wird nicht auf Skrum, sondern auf eine Mischung aus Kanban und einem “Standup” genannten Ritual gesetzt, bei dem alle Mitarbeiter einschließlich der Geschäftsleitung zu einem festgelegten Zeitpunkt am Tag

zusammenkommen und berichten, woran sie jeweils arbeiten, wie der Stand in der betreffenden Aufgabe ist und welche Probleme es gibt. Ein durchschnittlicher Standup dauert etwa 30 Minuten.

1.4 Betreuung vor Ort

Zum Zeitpunkt meines Praktikums bin ich bereits fester Bestandteil der Firma mit einer Vollzeitstelle. Mein Betreuer vor Ort ist Daniel Molnar. Er ist einer der Senior Entwickler in der Firma und hat einen hybriden Aufgabenbereich. Er bildet die Schnittstele zwischen Frontend- und Backendentwicklung und kann mich dadurch umfangreich mit seinem Wissen unterstützen. Ich arbeite aber auch mit allen anderen Entwicklern vor Ort eng zusammen, was der recht kleinen Firmenstruktur geschuldet ist.

1.5 Auswahl des Praktikumsplatzes

Nachdem ich ohnehin einen unbefristeten Arbeitsvertrag mit meinem Praktikumsgeber habe, bot es sich für mich an, diese Stelle auch als Praktikumsplatz zu nutzen. Für mich war bei der Wahl eines Arbeitsortes vor allem die Ausrichtung auf quellenoffene Software und ein Arbeitsumfeld im Unix/Linux Umfeld wichtig, auf den ich mich schon während der vorangegangenen Semester an der Technischen Hochschule ausgerichtet habe. Ein Arbeitsplatz an einem Windows-PC oder im Windows-Umfeld ebenso wie im proprietären Software-Bereich wäre für mich nicht in Frage gekommen.

1.6 Erfahrungen aus der Wahl eines Praktikumsplatzes

In der Zeit, in der ich mein Praktikum mache, scheint es zu einer absoluten Fachkräftenot gekommen zu sein. So wurde mir zum Beispiel an einem Tag auf dem Weg zu meiner Arbeitsstelle im Zug von einem Handelsreisenden ein Job im Webentwicklungsbereich in einer konkurierenden Firma angeboten.

Genauso passiert es des öfteren, dass sogenannte Headhunter gezielt die Profile der einschlägigen Entwicklerplattformen durchtelefonieren, um gewillte Entwickler zu finden. Sogar ein Hosting-Serviceanbieter, mit dem meine Firma eng zusammen arbeitet, machte mir in der Zeit meines Praktikums ein Angebot für einen Jobwechsel innerhalb einer normalen Geschäftskorrespondenz. Das Fazit daraus ist für mich, dass man sich ruhig seine Eigenheiten leisten kann und wie in meinem Fall auf ein Unix/Linux Umfeld und der Veröffentlichung der in der Arbeit geschriebenen Software bestehen kann. Denn es scheint besser zu sein sich weitestgehend frei entfalten zu können, als sich auf einen Markt auszurichten, der morgen schon wieder ein anderer sein könnte.

1.7 Bezahlung und Arbeitszeit

Da ich zum Zeitpunkt meines Praktikums schon über recht umfangreiches Praxis- und Fachwissen in meinem Bereich verfüge, habe ich in der Firma nicht als Junior-Entwickler angefangen. Dadurch bedingt habe ich ein Gehalt von 19€ Brutto je Stunde erhalten. Die Arbeitszeiten in der Agentur richten sich nach den Bedürfnissen der Arbeitnehmer. Als Vater eines Kleinkindes und als Student hat es das für mich sehr einfach gemacht. Es herrscht weitestgehend Vertrauensarbeitszeit. Nach kurzer Kommunikation mit dem Team kann nach Wahl begonnen und getauscht werden. Im Fokus liegt das Schaffen der täglichen Aufgaben und Ziele der Agentur und nicht so sehr die Kernarbeitszeit. Wobei es natürlich nicht möglich ist, selbige dauerhaft zu unterschreiten.

Kapitel 2

Aufgaben und Tätigkeiten

2.1 Tätigkeitsumfeld

Das Tätigkeitsumfeld meines Praktikumsgebers beschreibt sich wie folgt. Die Entwicklungsabteilung ist den Hauptteil der Arbeitszeit damit beschäftigt, die Aufgaben und Tickets der großen Kunden der Agentur abzuarbeiten. Nebenher werden auch noch eigene Werkzeuge entwickelt, die den Arbeitsalltag erleichtern oder bestimmte Lösungswege erst ermöglichen. So sind beispielsweise alle Softwareprojekte der Firma darauf ausgelegt, kontinuierlich ausgeliefert zu werden in einem Meilenstein oder Feature basierten Ansatz. Dies erfordert einer Struktur und Werkzeuge, die es nicht fertig oder schier gar nicht auf dem Markt zu kaufen gibt.
Am Anfang meiner Tätigkeit wurde ich mit der Aufgabe betraut, eines dieser Werkzeuge zu entwickeln. Ziel war es, alle möglichen Stati der vielen Projekte komfortabel und einheitlich überwachen und beobachten zu können. Aus dieser Aufgabe hervorgehend entwickelte sich nach und nach auch mein Tätigkeitsfeld in der Entwicklung abseits der Kundenaufträge. In enger Zusammenarbeit mit der Entwicklung ist es meine Aufgabe, Werkzeuge und Strukturen zu liefern, die den Arbeitsalltag ermöglichen. Dabei bietet mir der Job allerhand gestalterische Freiheit und so kommt es, dass ich mittlerweile in Vollzeit an einem Betriebssystem arbeite, das in Zukunft die Grundlage der Projekte meiner Firma bildet und das wichtige Software, wie die einer

Bank und einer Versicherung, die nächsten Jahre als Heimat dient. Der Weg bis dahin war allerdings lang und ich habe nicht die ganze Zeit an hoch sensiblen Systemen gearbeitet. Andere Projekte, die ich in meiner Zeit bei der ikusei realisieren konnte sind zum Beispiel ein chat-Bot, der einfache Aufgaben wie die Türsteuerung oder auch die Benachrichtigung einzelner Entwickler bei wichtigen Ereignissen übernimmt.

2.2 Aufgaben und Ziele

Aus der Zeit heraus entwickelt, stellen sich meine Aufgaben also wie folgt dar. Ich entwickle und betreue das von mir entworfene Betriebsystem und ein paar zum Teil oben genannte kleinere Projekte, wie zum Beispiel einen Automatisierer für das Ausstatten von Webseiten mit SSL-Zertifikaten. Weiterhin gibt es noch die Möglichkeit, dass ich exklusiv für Kunden arbeite und seltener, dass ich kleinere Tickets oder Kundenaufträge löse, welche sich zumeist im Support-Bereich ansiedeln. Aufgrund meiner umfangreichen Kenntnisse im Unix/Linux Umfeld genieße ich dabei weitestgehend gestalterische Freiheit, muss mich aber eng an die Bedürfnisse des Hauses halten, um nicht an meinen Kollegen vorbei zu entwickeln.
Zur Zeit des Praktikums habe ich mich weitestgehend um die Entwicklung und Kozeptionierung des firmeneigenen Betriebssystem gekümmert, weswegen für alle weiteren Ausführungen in diesem Bericht zur Spezialisierung und den gemachten Erfahrungen ausschließlich dieses Projekt herangezogen wird.
Ziel für den Verlauf des Praktikums war die Kozeptionierung und Entwicklung eines ersten lauffähigen Prototyps für die Erprobung der weiteren Nutzung innerhalb der Firma. Persönliches Ziel war es für mich vor allem, mein Wissen in dem Bereich zu professionalisieren und für den Entwurf und die Entwicklung eines sonst nur in meinem privaten Bereich stattfindenden Projektes bezahlt zu werden.

2.3 Tätigkeiten und Arbeitsergebnisse

2.3.1 Vorbedingungen zur Aufgabenstellung

Es scheint in der Natur von Softwareagenturen aller Art zu liegen, die eigene Software beständig weiter zu entwickeln und dabei oft die Grundlagen dieser in den Hintergrund oder gar ganz ins Abseits fallen zu lassen. So fließt beispielsweise ein Großteil der Arbeit am Linux-Kernel für Android Systeme nie zurück und bleibt damit verschlossen hinter Tür und Riegel großer Firmen wie Samsung, HTC und Co. Ebenso sieht es auf einer anderen Ebene bei den Webagenturen aus. Alle Agenturen brauchen einen funktionierenden Server mit einem Betriebssystem, das die Funktionalität und die Programme mit sich bringt, die Produkte der Agenturen zum laufen zu bringen. Bezahlt wird von den Auftraggebern jedoch schlicht nur für das Produkt. Ein Server wird meistens innerhalb einer Pauschale abgehandelt, gestellt und in jedem Fall nach kurzer Zeit erst wieder angefasst, wenn er nicht mehr richtig läuft, verändert werden soll oder, wenn, noch viel schlimmer, der Supergau in Form eines Einbruches geschehen ist. Die Server-Betriebssysteme unserer Zeit sind für alle möglichen Aufgaben auf einmal ausgelegt. Selbst die minimale Installation eines Ubuntu-Servers bringt mehrere hundert Megabyte und hunderte Programme mit sich. Viele Programme und Funktionalität, die all zu oft ungenutzt bleiben und dadurch aus dem Fokus der Wartung geraten. Damit verbunden verbergen sie Sicherheitslücken, an die nach kurzer Zeit keiner mehr denkt. Container-basierte Systeme wie Docker können hier eine Lösung bieten, aber was, wenn es die Gesetze einer Branche erfordern, dass es eine physische Trennung zwischen dem eigenen und deren Servern geben muss, sei es um sensible Daten zu schützen oder schlicht weg aus Performance Gründen. An diese Stelle treten Alternativen wie Alpine-Linux und NixOS. Letzteres ist ein System, das in einer speziellen Sprache in Skriptdateien zusammengesetzt werden kann. Es wird dabei für den Anwendungsfall in einer Datei beschrieben und während der Auslieferungszeit gebaut und zusammengesetzt. So erhält man ein System, das nur Komponenten enthält, die auch wirklich benötigt werden und kann beim Erneuern der Software durch das Entwicklungsteam nebenbei gleich ein komplett neues System ausliefern.

Alpine-Linux hingegen setzt auf einen wirklich kleinen Auslieferungsumfang. Ein komplettes System kann weit unter zehn Megabyte haben und ist trotzdem komfortabel bedienbar. Besonders bemerkenswert ist jedoch das hohe Sicherheitsaugenmerk, das die Entwickler der Distribution auf es legen. Standardmäßig kommt es bereits mit einem gehärteten Kernel und Regeln für Anwendungen sowie einer Randomisierung der Hauptspeicheradressen und einer Stacksmasching Protektion. All diese Dinge erfordern viel mehr Arbeit als eine kleine Agentur aufbringen kann um sie aktiv auf einem Standard Server einzurichten und zu warten. Wobei die Distribution Alpine auch noch auf eine nicht von der NSA entwickelte Alternative des bekannten SELinux setzt und so für mehr Sicherheit sorgt. Nach dieser Einführung könnte man die Idee ein eigenes System zu entwerfen durch einen einzelnen Praktikanten für eine kleine Agentur für abwegig halten, allerdings ist eine Verbindung aus den oben genannten Systemen derart wünschenswert, dass sich schnell ein Sponsor fand und der Idee ein Konzept folgen musste.

2.3.2 Konkretisierte Aufgabenstellung

Ziel der Entwicklung eines agentureigenen Systems war nicht das oft zitierte "Not invented here" also ein individualistischer Akt der Selbstüberschätzung, sondern der Versuch, aus gegebenen Systemen und Paradigmen ein lauffähiges System zu schaffen, das einfach zu Warten und schnell zu erneuern ist, dabei aber nur den jeweiligen und keinen weiteren Zweck erfüllt als eben jenen, für den es erschaffen worden ist. Es sollte ein Weg gefunden werden, Alpine-Linux, NixOS artig für den individuellen Zweck zusammenzusetzen, wobei gleichzeitig die Orchestrierbarkeit und die Wartbarkeit im Auge behalten werden sollten. Es sollte, wenn möglich, zur Auslieferung der Software automatisch erstellt werden und die fertige Software ohne Konfigurationsaufwand entgegennehmen.

2.3.3 Lösungsweg der bestehenden Aufgabe

Erfreulicherweise gibt es die Distribution Alpine-Linux in etlichen Ausbaustufen. Ich habe mich gezielt für eine Version, die lediglich aus dem Root-

Filesystem besteht, entschieden. Da ich während der Arbeit in der Firma recht passable Ruby Kenntnisse erlernen konnte und ich die Paradigmen der bedingungslosen Objektifizierung, sowie die Fähigkeit der Live Interpretierbarkeit und der Sparachstabilität über Alternativen wie zum Beispiel Python hinaus zu schätzen gelernt habe, fiel mir die Wahl der Sprache für das Framework zur Erstellung meines künftigen Systems leicht. Ein stets aktuelles Filesystem lässt sich aus den Paketquellen der Distribution genauso einfach erlangen wie ein frischer Paketmanager. Die Struktur der Paketquellen ist dabei einfach zu syntetisieren, da auch der Paketmanager selbst darauf zugreifen muss. So ist es recht leicht zu sagen, ich möchte stets Software der "latest-stable" Branch, der Version "v3.5" et cetera, zu gewünschter Architektur und das von derzeit 22 Spiegelservern. Aufgrund der geringen Größe des Systems kann es komplett im Ram zusammengesetzt werden. Nach der Wahl einer Version der Distribution und eines Spiegelservers sowie einer Architektur muss noch die gewünschte Version des Paketmanagers festgelegt werden. Vorzugsweise sollte das Framework von alleine herausbekommen, welche die aktuelle Version ist. Sind diese Wahlen getroffen, kann ein erstes System zusammengesetzt werden. Ab diesem Punkt beginnt allerdings die gestalterische Freiheit, was dem Projekt jedoch ermöglicht das fertige Ergebnis als eigenes System zu benennen. Schließlich unterscheidet sich Justin-Bieber-Linux von einem Standard Ubuntu nur darin, dass es ein anderes Hintergrundbild hat. Ich werde mit dieser Aufgabe etwas weiter gehen. Nach der Wahl eines Arbeitsordners muss der Paketmanager der Distribution über die wie oben festgelegte Adresse geholt werden. Da das System noch nicht lauffähig ist, muss dafür ein Gastsystem dienen. Auf eben diesem System wird mittels des heruntergeladenen und entpackten Paketmanagers über die Kommandozeilenoption

```
./sbin/apk.static --arch x86 -X ${mirror}/${branch}/main -U \
--allow-untrusted --root ${chroot_dir} --initdb add alpine-base
```

Das Root-Filesystem installiert und hat so eine Basis für alle weiteren Dinge geschaffen. Danach müssen die benötigten "Devices" eingerichtet werden. Die Natur der Aufgabe verlangt es, nur die nötigsten Geräte zu initialisieren. Da kein System für alle möglichen Dinge gebaut werden soll, erstellen wir

mit den folgenden Bashbefehlen alles, was das System an Hardware benötigt. Dank der Kompatibilität von Ruby ist das Einbinden von Bashscripten oder Befehlen kein Problem und so sieht der Abschnitt, der Hardwarekompatibilität nach Unix Paradigmen herstellt, wie folgt aus.

```
create_devices () {
    echo "creating devices..."
    mknod -m 666 ${chroot_dir}/dev/full c 1 7
    mknod -m 666 ${chroot_dir}/dev/ptmx c 5 2
    mknod -m 644 ${chroot_dir}/dev/random c 1 8
    mknod -m 644 ${chroot_dir}/dev/urandom c 1 9
    mknod -m 666 ${chroot_dir}/dev/zero c 1 5
    mknod -m 666 ${chroot_dir}/dev/tty c 5 0
    rm -rf  ${chroot_dir}/dev/null
    mknod -m 666 ${chroot_dir}/dev/null c 1 3
    mknod -m 666 ${chroot_dir}/dev/1 c 4 1
    mknod -m 666 ${chroot_dir}/dev/2 c 4 2
    mknod -m 666 ${chroot_dir}/dev/tty2 c 4 2
    rm -rf ${chroot_dir}/dev/tty1
    echo "done"
}
```

Wie man sieht, werden durch das Installieren des Basis-Systems schon ein paar Devices erstellt, welche wir aber löschen und zum Teil sogar mit anderen Rechten neu erstellen. Nach dem Erstellen der Geräte benötigt ein System mit dem Ziel lauffähig zu sein noch eine funktionierende fstab. Diese dient dem Einhängen der Partitionen und ist von nachhaltiger Wichtigkeit für das System. Fehler an dieser Stelle führen schlichtweg zu Datenmüll. Auch ist es wichtig, dass sich das System dabei an die Struktur der Server des Hosters hält. Es ist nicht sonderlich einfach an diese Informationen zu gelangen, auch weil derartige Dinge heute oft automatisch konfiguriert werden und die Hoster sie schlicht selbst nicht genau beschreiben können. Das vorliegende System erstellt seine Partitionstabelle wie folgt.

```
create_fstab() {
    echo "create fstab..."
cat >${chroot_dir}/etc/fstab <<EOFSTAB
# START ----------------------------------
none /dev/pts devpts rw,gid=5,mode=620 0 0
none /dev/shm tmpfs defaults 0 0
# END ------------------------------------
EOFSTAB
    read -p '<ENTER> to edit/verify the inittab'
    $EDITOR ${chroot_dir}/etc/fstab
    echo "done"
}
```

An dieser Stelle benötigt das System eine funktionierende “inittab”. Ohne diese startet das System erst gar nicht und erzeugt ebenfalls nur Daten-Abfall. Eine “inittab” liegt durch die Installation des Basis-Systems bereits vor ist aber zu offen für den vorliegenden Anwendungsfall und benötigt einige Komponenten schlichtweg nicht. Andere müssen aufgrund der eigenen Gestaltung der Devices und der inittab hinzugefügt werden. Sie wird wie folgt erzeugt.

```
create_inittab() {
    echo "create inittab"
    TAB=${chroot_dir}/etc/inittab
    sed -i 's/^tty/#tty/g' $TAB
    sed -i 's/^::ctrl/#::ctrl/g' $TAB
cat >>${chroot_dir}/etc/inittab <<EOINITTAB
1:2345:respawn:/sbin/getty 38400 console
2:2345:respawn:/sbin/getty 38400 tty2
EOINITTAB
    read -p '<ENTER> to edit the inittab'
    $EDITOR $TAB
    echo "done"
}
```

An dieser Stelle müssen zur Lauffähigkeit des Systems nur noch Kleinigkeiten hinzugefügt werden. Es fehlt die Installation des Paket-Managers innerhalb des Systems, bisher ist er ja nur von ausserhalb benutzt worden. Er wird mit einem ähnlichen Befehl installiert wie das Basis System oben. Es müssen dafür aber noch ein paar Verzeichnisse angelegt werden. Das System braucht einen Hostnamen und natürlich muss es wissen, auf welche Paketquellen es in Zukunft zurückgreifen darf. Von Vorteil ist es außerdem, dem System den Nameserver des Hosters mitzuteilen, dies macht die Namensauflösung besonders schnell und ist dem Zweck später dienlich. Bis zu diesem Punkt ist bis auf das Wählen der Paketquellen und der Version des Betriebssystems die Abhandlung der Befehle eher statisch. Ab diesem Punkt erhält man ein System mit einem Paketmanager und einem Komandozeilenemulator. Das System ist an dieser Stelle in etwa 7 Megabyte groß. Nun müssen Paketumfänge für Anwendungszenarien gefunden werden. Zur Zeit des Praktikums wurden etwa 160 Programme installiert, die das System auf eine Gesamtgröße von 161 Megabyte gebracht haben.

2.3.4 Bestehende Probleme

Viele der Anforderungen konnten auf die oben genannte Art und Weise gelöst werden. Das Erstellen von Paketkollektionen zu Anwendungszenarien wird noch einiges an Zeit in Anspruch nehmen, stellt aber kein ernsthaftes Problem dar. Innerhalb der Agenturarbeit ist die Anzahl der möglichen Szenarien begrenzt und Pakete lassen sich leicht nachinstallieren, wobei die Orchestrierbarkeit nicht zwingend kaputt gehen muss. Leider wird es durch das neue System nicht möglich, eine Downtime zu umgehen. Allerdings lässt sie sich durch das anwendungsspezifische Betriebssystem und die Angepasstheit an die agenturspezifische Software stark verkürzen. Es besteht also nur ein kleines Problem, das vielleicht in Zukunft durch eine intelligente Struktur seitens des Hosters weiter minimiert werden kann.

2.3.5 Aufgabenfazit

Hier soll noch kein vollständiges Fazit stehen, sondern nur mit den am Anfang vorhandenen Zielen verglichen werden. Die Aufgabe wurde vollends erfüllt und es wird der Firma in Zukunft möglich sein, einfach aber effizient ein vollständiges System nebenbei zu warten und zu pflegen. Wartungsbedingte downtimes werden sich verkürzen, weil bereits in der Agentur gewartet und weiterentwickelt werden kann. Das Ziel eines live bei Auslieferung erzeugten Systems konnte erfüllt werden und wird die Arbeit der ikusei nachhaltig verändern.

Kapitel 3

Einsichten und Fazit

3.1 Technik

In der Zeit meines Praktikums habe ich gelernt, dass es möglich ist, auch in kleinen Agenturen und mit beschränkten Resourcen viel zu erreichen. Die Situation nicht hinzunehmen scheint in der gegenwärtigen Zeit eine mögliche Alternative, sei es bei der Wahl des Praktikumsplatzes oder der Wahl der Entwicklung einer eigenen Alternative zu den gängigen Server-Betriebssystemen. Ich habe gelernt, dass die Hochschule nur eine Richtlinie und keine allumfassende Institution ist. Dass Technologien immer dynamisch sind und ich nicht stehen bleiben darf. Java zu programmieren, die Sprache der Hochschule, ist in der Welt, in der ich lebe, absolut in den Hintergrund gerückt. Habe ich in der Hochschule noch streng typisiert und mir erst überlegt, wie der Datenfluss meiner Applikation sein könnte, ist diese Überlegung zu gunsten der immer stäkeren Abstraktion von Objekten zum Opfer gefallen. Es gibt nur wenige Situationen, in denen ich bei Sprachen wie Ruby oder JavaScript noch darüber nachdenken muss, von welchem Datentyp mein Opjekt nun gerade sein könnte. Aber auch die damit einhergehenden Nachteile bei beispielsweise Rechengenauigkeit von Fließkommazahlen habe ich dadurch kennengelernt. Ich bin bis heute überrascht und beeindruckt, wie schier gigantische Software entsteht und hätte mir damit einhergehend mehr Einblicke in Softwarengeneering gewünscht. Auch finde ich sollte es eine Veranstaltung

zum Thema Versionskontrolle geben. Ich habe zum Glück ein paar Erfahrungen mitgebracht, habe mich aber dennoch nicht immer leicht getan im Umgang mit Teams und echter Arbeit.

3.2 Methodik

Als ich mit dem Studium der Informatik angefangen habe, dachte ich, dass Software von klugen Menschen entworfen wird, die wie in den vielen Filmen einfach nur die Zeilen lang runter tippen müssen. Nachdem ich nun eine Weile in der Praxis arbeite habe ich gelernt, dass für jeden der Augenblick kommt in dem er auch bei dem, was er täglich tut, mal nachsehen muss. Ich weiß jetzt, dass man nur lernt, wenn man sich an der Arbeit der anderen orientiert und sie zu neuen Dingen verknüpft. Alles in allem zeigen die bisherigen Einblicke nur wenig Vorreiter.

3.3 Sonstiges

Nach 10 Jahren Arbeit in anderen Bereichen habe ich gedacht, dass auch die Welt der Technik und der Informationen ähnlich funktioniert. Viele der in der Hochschule erlernten Paradigmen zum Beispiel, dass es agiles Schätzen wirklich gibt, habe ich nicht für möglich gehalten bis ich es gesehen habe. Ich hätte mir in meinem früheren Beruf etwas ähnlichen gewünscht. Auch, dass für die Beseitigung von Bugs seitens der Kunden gezahlt wird finde ich interessant. Am stärksten schockiert hat mich allerdings die Bürokommunikation. An meinem ersten Tag hat man mich darauf hingewiesen, dass ich aufgrund des Signals einer der Bürolampen, die die Farbe wechseln kann, im Moment meine Kollegen, auch die, die physisch neben mir sitzen, nur über den firmeneigenen Chat anschreiben kann, nicht aber ansprechen. Ein Kommunikationsmodell an das ich mich erst noch gewöhnen musste.

3.4 Fazit

Mein persönliches Ziel habe ich erreicht. Ich habe einen festen Platz, an dem ich gefunden gerne arbeite und bin mit dem Umfeld sehr zufrieden. Ich gehe in meinen Aufgaben auf und konnte die mir gesteckten Ziele erreichen und übertreffen. Ich habe einen umfassenden Einblick in die Arbeitswelt in einem für mich neuen Bereich gewonnen und deren Paradigmen kennen gelernt. Ich weiß heute, dass und welche Eigenheiten ich mir erlauben kann und wie ich mit den Eigenheiten dieses Umfeldes umzugehen habe. Ich hätte mir seitens der Hochschule einerseits sehr einen direkteren Einstieg in Technologien wie Git oder in SQL gewünscht, ausserdem wäre es von Vorteil, mehr Algorithmieren zu lehren als es bisher der Fall ist ansonsten bin ich nicht zuletzt durch viel Eigenleistungen zu dem Stand gekommen, an dem ich heute bin.