Fakultät Informatik

Hochschule Reutlingen
Alteburgstraße 150
72762 Reutlingen

Bachelorarbeit
zur Erlangung des akademischen Grades
Bachelor of Science (B.SC.)
in Kooperation mit Novatec GmbH

API Security Testing mit OWASP Zap und RAML

Mert Yücel

Studiengang:	Wirtschaftsinformatik
Prüfer/in:	Prof. Dr. Martin Schmollinger, Prof. Dr. Muster Mustermann
Betreuer/in:	Anreas Falk, Jan Horak
Beginn am:	1. September 2018
Beendet am:	15. Januar 2019

Kurzfassung

REST-APIs sind heutzutage weit verbreitet und dank ihrer Einfachheit, Skalierbarkeit und Flexibilität werden sie weitgehend als Standardprotokoll für die Web-APIs angesehen. Es scheint plausibel anzunehmen, dass die Ära der Desktop-basierten Anwendungen kontinuierlich zurückgeht und im Zuge dessen, die Benutzer von Desktop- zu Web- und weiteren mobilen Anwendungen wechseln.

Bei der Entwicklung von REST-basierten Web-Anwendungen wird ein REST-basierter Web Service benötigt, um die Funktionalitäten der Web-Anwendung richtig testen zu können, da die gängigen Penetrationstest-Werkzeuge für REST-APIs nicht direkt einsatzfähig sind, wird die Sicherheit solcher APIs jedoch immer noch zu selten überprüft und das Testen dieser Arten von Anwendungen ist eine sehr große Herausforderung. Hauptsächlich für das erstmalige Testen ist es für den Betreiber von Webanwendungen sehr unüberschaubar. Verschiedene Werkzeuge, Frameworks und Bibliotheken sind dazu da, die Testaktivität automatisieren zu können. Die Nutzer wählen diese Dienstprogramme basierend auf ihrem Kontext, ihrer Umgebung, ihrem Budget und ihrem Qualifikationsniveau. Einige Eigenschaften von REST-APIs machen es jedoch für automatisierte Web-Sicherheitsscanner schwierig, geeignete REST-API-Sicherheitstests für die Schwachstellen durchzuführen.

Diese Bachelorarbeit untersucht wie die Sicherheitstests heutzutage realisiert werden und versucht qualitativ-deskriptiv aufzudecken, ob auf solche Sicherheitstests verlässlich sind. Es werden verschiedene existierende Tools verglichen, die das Testen von RESTful APIs unterstützen. Dann wird ihre Vor- und Nachteile herausgefunden und gegeneinander abgewägt. Es wird auch Gewißheit verschaffen, wie die technische Schwachstellen(wie z.B. Konzeptionsfehlern, Programmierfehlern und Konfigurationsfehlern) von Webanwendungen dargelegt. Parallel zu den Schwachstellen werden die jeweiligen Gefahren und Angriffspunkte für eine Webanwendung anhand von ”OWASP Top 10 - Risiken” erörtert.

Im Rahmen dieser Bachelorarbeit wird ein Plugin für das Open Source Werkzeug OWASP Zap entwickelt, um Schwachstellen automatisch zu untersuchen, um ein gewisses Maß an Sicherheit gewährleisten zu können. Dieses Plugin muss so implementiert werden, so dass die RAML-Dokumentationen importiert werden können, um automatisierte Penetrationstests ausführen zu können. Es wird auch eine Möglichkeit geschaffen, dass das Plugin in OWASP Zap sowohl über die UI, aber auch über Kommandozeile ansprechbar ist, um dies auch in CI-Umgebungen einbinden zu können.

Danksagung

Bla bla..

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Verzeichnis der Listings

Verzeichnis der Algorithmen

Abkürzungsverzeichnis

ER error rate. 23

FR Fehlerrate. 23

IKT Informations- und Kommunikationstechnologie. 3

RDBMS Relational Database Management System. 23

1. Einleitung

Bei LaTeX werden Absätze durch freie Zeilen angegeben. Da die Arbeit über ein Versionskontrollsystem versioniert wird, ist es sinnvoll, pro *Satz* neue Zeile im `.java`-Dokument anzufangen. So kann einfacher ein Vergleich von Versionsständen vorgenommen werden.

Die Arbeit ist in folgender Weise gegliedert: In Kapitel 2 werden die Grundlagen dieser Arbeit beschrieben. Schließlich fasst Kapitel 2 die Ergebnisse der Arbeit zusammen und stellt Anknüpfungspunkte vor.

1.1. Motivation

bla bla..

1.2. Zielsetzung

bla bla..

1.3. Aufbau der Arbeit

1. Vorerst werden in Kapitel 2..
2. In einer Anforderungsanalyse in Kapitel 3..
3. Kapitel 4 vermittelt..
4. Es folgt irgendwas in Kapitel 5. In diesem Teil..
5. Kapitel 6 dokumentiert
6. Die Gesamtarbeit..

2. Grundlagen

Im folgenden Kapitel werden auf die mehrere Grundlagen eingegangen. Zu Beginn wird ein Überblick über das Informationssicherheit geschaffen und Grundlegende Begriffe, Schutzziele, Schwachstellen, Bedrohungen und Angriffe aufgezeigt. Anschließend werden die Sicherheitsrichtlinie und Sicherheitsinfrastruktur vorgestellt. Zuletzt wird ein Überblick über die Technologien des Projektes gegeben.

2.1. Informationssicherheit

> *"Meine Nachricht für Unternehmen, die sie denken, nicht angegriffen worden ist: Sie suchen nicht hart genug."* James Snook

Informationssicherheit ist eine wichtige Herausforderung im Bereich der neuen Informationstechnologien. Sie ist heutzutage in nahezu allen Bereichen von zentraler Bedeutung. Die Sicherheit von Informationen, Daten, Geschäft und Investitionen gehören zu den Schlüsselprioritäten und die Verfügbarkeit von Informationen in der richtigen Zeit und Form ist heutzutage in jeder Organisation ein Muss. IT-Sicherheit hat die Aufgabe, Unternehmen und deren Werte zu schützen, indem sie böswillige Angriffe zu identifizieren, zu reduzieren und zu verhindern.

Das schnelle Wachstum von Webanwendungen hängt zweifellos von der Sicherheit, dem Datenschutz und der Zuverlässigkeit der Anwendungen, Systeme und unterstützenden Infrastrukturen ab. Obwohl IT-Sicherheit heutzutage sehr große Rolle in unserem Leben spielt, liegt das durchschnittliche Sicherheitswissen von IT-Fachkräften und Ingenieuren weit hinter. Das Internet ist jedoch für seine mangelnde Sicherheit bekannt, wenn sie nicht genau und streng spezifiziert, entworfen und getestet werden. In den letzten Jahren war es offensichtlich, dass der Bereich der IT-Sicherheit leistungsfähige Werkzeuge und Einrichtungen mit überprüfbaren Systemen und Anwendungen umfassen muss, die die Vertraulichkeit und Privatsphäre in Webanwendungen wahren, die die Probleme definieren[Fur08, S. 1].

Wegen der Realisierung eine lückenlose Abwehr von Angriffen in der Wirklichkeit nicht möglich ist, inkludieren das Gebiet der IT-Sicherheit hauptsächlich auch Maßnahmen und Vorgehensweise, um die potenziellen Schäden zu vermindern, um die Risiken beim Einsatz von Informations- und Kommunikationstechnologie (IKT)-Systemen zu verringern. Die Angriffsversuche müssen rechtzeitig und mit möglichst hoher Genauigkeit erkannt werden. Dazu muss auf eingetretene Schadensfälle mit geeigneten technischen Maßnahmen reagiert werden. Es sollte klargestellt werden, dass Techniken der Angriffserkennung und Reaktion ebenso zur IT-Sicherheit gehören, wie methodische Grundlagen, um IKT-Systeme so zu entwickeln, dass sie mittels Design ein hohes Maß an Sicherheit bieten. Durch IT-Sicherheit können eine Möglichkeit geschaffen werden, um die Entwicklung vertrauenswürdiger Anwendungen und Dienstleistungen mit innovativen

Geschäftsmodellen beispielsweise im Gesundheitsbereich, bei Automotive-Anwendungen oder in zukünftigen, intelligenten Umgebungen zu verbinden[Eck13, S. 20–21].

2.1.1. Grundlagen der IT-Sicherheit

2.1.1.1. Offene- und geschlossene Systeme

Ein IT-System besteht aus ein geschlossenes und ein offenes System mit der Begabung zur Speicherung und Verarbeitung von Informationen. Die offene Systemen (z. B. Windows) sind vernetzte, physisch verteilte Systeme mit der Möglichkeit zum Informationsaustausch mit anderen Systemen[Eck13, S. 22–23]. Tom Wheeler definiert offene Systeme als *"jene Hard- und Software-Implementierungen, die der Sammlung von Standards entsprechen, die den freien und leichten Zugang zu Lösungen verschiedener Hersteller erlauben. Die Sammlung von Standards kann formal definiert sein oder einfach aus De-facto-Definitionen bestehen, an die sich die großen Hersteller und Anwender in einem technologischen Bereich halten."*[Whe13, S. 4] Ein geschlossenes System (z. B. Datev) baut auf der Technologie eines Herstellers auf und ist zu Konkurrenzprodukten nicht kompatibel. Es dehnt sich auf einen bestimmten Teilnehmerkreis aus und beschränkt ein bestimmtes räumliches Gebiet[Eck13, S. 22–23].

2.1.1.2. Soziotechnische Systeme

IT-Systeme werden in unterschiedliche Strukturen (gesellschaftliche, unternehmerische und politische Strukturen) mit verschiedenem technischen Know-how und für sehr verschiedene Ziele in Betrach gezogen[Eck13, S. 23]. IT-Sicherheit wird den Schutz eines soziotechnischen Systems gewährleistet. Darausfolgend ergibt sich dann auch das Ziel der IT Sicherheit. Die Unternehmen bzw. Institutionen und deren Daten sollen gegen Schaden und Bedrohungen geschützt werden. Insbesondere soll auf den Schutz von IT-Systemen angewiesen sein[STS18].

2.1.1.3. Information und Datenobjekte

IT-Systeme haben die Begabung Informationen zu speichern und zu verarbeiten. Die Information wird in Form von Daten bzw. Datenobjekten repräsentiert. **Passiven Objekten** (z.B. Datei, Datenbankeintrag) haben die Fähigkeit, Informationen zu speichern. **Aktive Objekten** (z.B. Prozesse) haben die Fähigkeit, sowohl Informationen zu speichern als auch zu verarbeiten[Eck13, S. 23]. **Subjekte** sind die Benutzer eines Systems und alle Objekte, die im Auftrag von Benutzern im System aktiv sein können[Eck13, S. 24].

Informatik wird als die Wissenschaft, Technik und Anwendung der maschinellen Verarbeitung, Speicherung, Ubertragung und Darstellung von Information identifiziert. **Informationen** sind einen abstrakten Gehalt ("Bedeutungsinhalt", "Semantik") eines Dokuments, einer Aussage, Beschreibung, Anweisung oder Mitteilung[Bro13, S. 5] und sie werden durch die Nachrichten übermittelt[BKRS13, S. 18]. Information wird umgangssprachlich sehr oft für Daten verwendet aber es gibt Untershied zwischen Daten und Information. Der Mensch bildet die Informationen in Daten ab, indem er die Nachrichten übertragen oder verarbeiten. Die Daten, die maschinell verarbeitbare Zeichen sind, werden durch in einer Nachricht enthaltene Information die Bedeutung

der Nachricht darstellen. Auf der Ebene der Daten geschiet die Übertragung oder Verarbeitung und die Resultat wird vom Mensch als Information interpretiert[BSI08].

2.1.1.4. Sicherheit

2.1.1.4.1. Funktionssicher

In den Sicheren Systeme sollen alle korrekten Spezifikationen korrekt funktioniert werden und eine hohe Zuverlässigkeit und Fehlersicherheit gewährleistet werden. Die Isolierung von der Außenwelt weicht konstant durch die stetig zunehmende Vernetzung jeglicher Systeme mit Informationstechnik auf. Die Zweck von Funktionssicherheit (engl. safety) ist, dass die Umgebung vor dem Fehlverhalten des Systems schützen. Bei der Entwicklungsphase müssen systematische Fehler vermieden werden. Durch die Überwachung im laufenden Betrieb müssen die Fehlern erkannt werden und solche Fehlern müssen dominiert werden, um ein funktionsichere Zustand zu erreichen[MHTK15].

2.1.1.4.2. Informationssicher

Das Hauptziel von Informationssicherheit (engl. security) ist, dass die Informationen zu schützen, die sowohl auf Papier, in Rechnern oder auch in Köpfen gespeichert sein. IT-Sicherheit kümmert sich ersten um den Schutz digitalen Informationen und deren Verarbeitung[BSIDI11, S. 81], um unautorisierte Informationsveränderung oder -gewinnung zu verhindern[Eck13, S. 26].

2.1.1.4.3. Datensicherheit und Datenschutz

Datensicherheit bedeutet, dass der Zustand eines Systems der Informationstechnik, in dem die Risiken, die im laufenden Betrieb dieses Systems bezüglich von Gefährdungen anwesend sind, durch Maßnnahmen auf ein bestimmtes Menge eingeschrankt sind[Ebe94, S. 14–15].

2.1.1.4.4. Verlässligkeit

asdasd

2.1.2. Schutzziele

bla bla..

2.1.3. Schwachstellen, Bedrohungen, Angriffe

2.1.3.1. Bedrohungen

2.1.3.2. Angriffs- und Angreifer-Typen

2.1.3.3. Rechtliche Rahmenbedingungen

2.1.4. Sicherheitsrichtlinie

2.1.5. Sicherheitsinfrastruktur

2.2. Technologien des Projektes

2.2.1. Java

2.2.2. Microservice Architekturen

2.2.3. Spring Boot

2.2.3.1. Spring MVC Komponente

2.2.3.2. Spring Rest Docs

2.2.4. RESTful Web Services

2.2.5. RAML

2.2.6. Open Source Werkzeug OWASP Zap

2.2.7. RAML-Parser für Java

2.2.8. Test-Driven mit automatisierten Tests

3. Sicherheitsrisiken von Webanwendungen

Hier wird der Hauptteil stehen. Falls mehrere Kapitel gewünscht, entweder mehrmals `\chapter` benutzen oder pro Kapitel eine eigene Datei anlegen und `ausarbeitung.tex` anpassen.

LaTeX-Hinweise stehen in Anhang A.

3.1. Schwachstellen

3.1.1. OWASP Top 10 Risiken

3.1.2. Weitere Risiken

3.1.3. Common Vulnerability Scoring System

3.1.4. Common Vulnerability Exposures (CVE)

3.2. Technische Schwachstellen

3.2.1. Programmierfehler

3.2.2. Konfigurationsfehler

3.2.3. Konzeptionsfehler

3.2.4. Fehler resultierend aus menschlichem Verhalten

4. Penetrationstest

Hier wird der Hauptteil stehen. Falls mehrere Kapitel gewünscht, entweder mehrmals `\chapter` benutzen oder pro Kapitel eine eigene Datei anlegen und `ausarbeitung.tex` anpassen.

LaTeX-Hinweise stehen in Anhang A.

4.1. Grundlegendes Konzept

4.1.1. Black-Box

4.1.2. White-Box

4.1.3. Gray-Box

4.2. Kriterien für Penetrationstests

4.3. Ablauf eines Penetrationstest

4.4. Penetrationstest-Tools

4.4.1. ABC

4.5. Automatisierte Penetrationstest-Tools

4.5.1. ABC

4.6. Vor- und Nachteile zwischen manuelle und automatisierte Penetrationstest-Tools

5. Entwicklung einer Spring-Boot-Webanwendung mit Sicherheitslücken

Hier wird der Hauptteil stehen. Falls mehrere Kapitel gewünscht, entweder mehrmals `\chapter` benutzen oder pro Kapitel eine eigene Datei anlegen und `ausarbeitung.tex` anpassen.

LaTeX-Hinweise stehen in Anhang A.

6. Entwicklung von Plug-in für das OWASP-ZAP

Hier wird der Hauptteil stehen. Falls mehrere Kapitel gewünscht, entweder mehrmals `\chapter` benutzen oder pro Kapitel eine eigene Datei anlegen und `ausarbeitung.tex` anpassen.

LaTeX-Hinweise stehen in Anhang A.

7. Was ist die Vorteile von diesem Plug-in gegenüber anderen Plug-ins?

Hier wird der Hauptteil stehen. Falls mehrere Kapitel gewünscht, entweder mehrmals `\chapter` benutzen oder pro Kapitel eine eigene Datei anlegen und `ausarbeitung.tex` anpassen.

LaTeX-Hinweise stehen in Anhang A.

8. Fazit

Hier bitte einen kurzen Durchgang durch die Arbeit.

A. Benutzerhandbuch für das OWASP-Zap RAML Plug-in

> Probleme kann man niemals mit derselben Denkweise lösen, durch die sie entstanden sind.
>
> *(Albert Einstein)*

Pro Satz eine neue Zeile. Das ist wichtig, um sauber versionieren zu können. In LaTeX werden Absätze durch eine Leerzeile getrennt.

Folglich werden neue Abstäze insbesondere *nicht* durch Doppelbackslashes erzeugt. Der letzte Satz kam in einem neuen Absatz.

Wörter am besten mittels `\enquote{...}` „einschließen“, dann werden die richtigen Anführungszeichen verwendet.

Beim Erstellen der Bibtex-Datei wird empfohlen darauf zu achten, dass die DOI aufgeführt wird.

A.1. Quellcode

Listing A.1 zeigt, wie man Programmlistings einbindet. Mittels `\lstinputlisting` kann man den Inhalt direkt aus Dateien lesen.

Quellcode im `<listing />` ist auch möglich.

A.2. Abbildungen

Die Abbildung A.1 und A.2 sind für das Verständnis dieses Dokuments wichtig. Im Anhang zeigt Abbildung A.4 auf Seite 24 erneut die komplette Choreographie.

Es ist möglich, SVGs direkt beim Kompilieren in PDF umzuwandeln. Dies ist im Quellcode zu latex-tipps.tex beschrieben, allerdings auskommentiert.

Listing A.1 lstlisting in einer Listings-Umgebung, damit das Listing durch Balken abgetrennt ist

```
<listing name="second sample">
  <content>not interesting</content>
</listing>
```

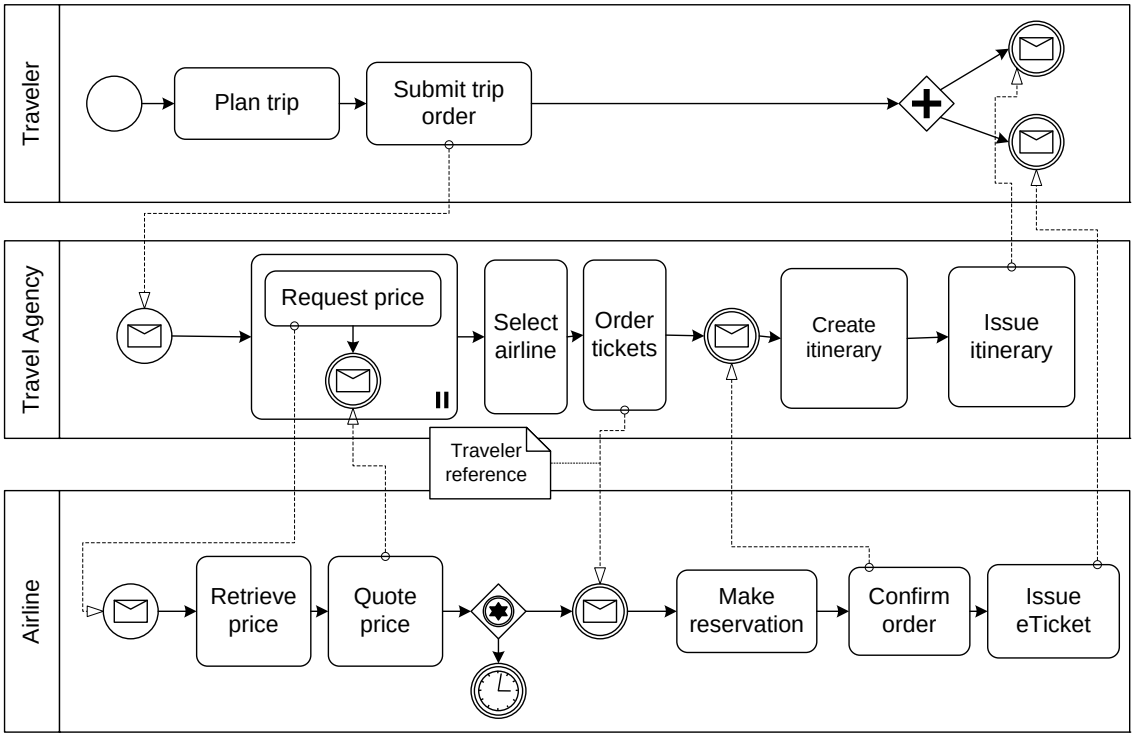


Abbildung A.1.: Beispiel-Choreographie

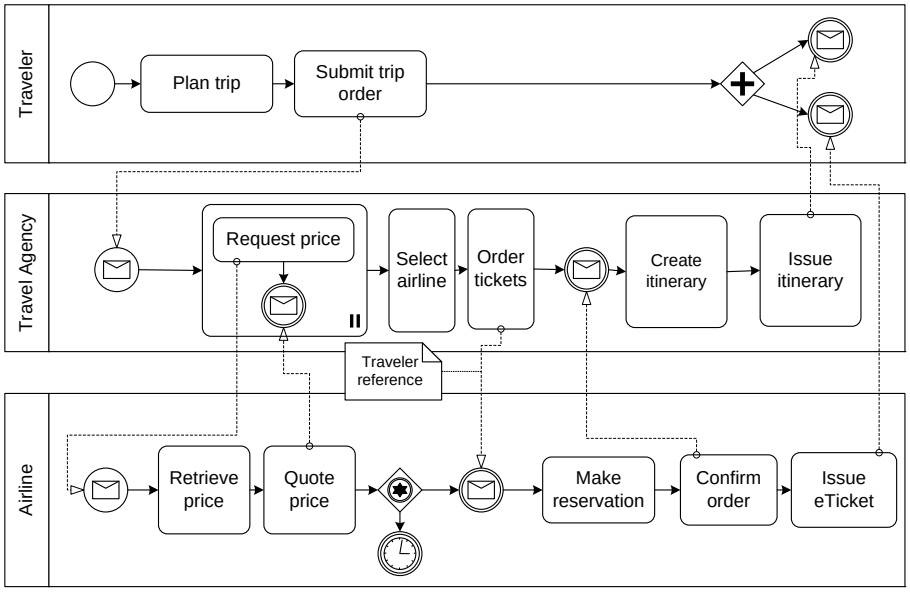
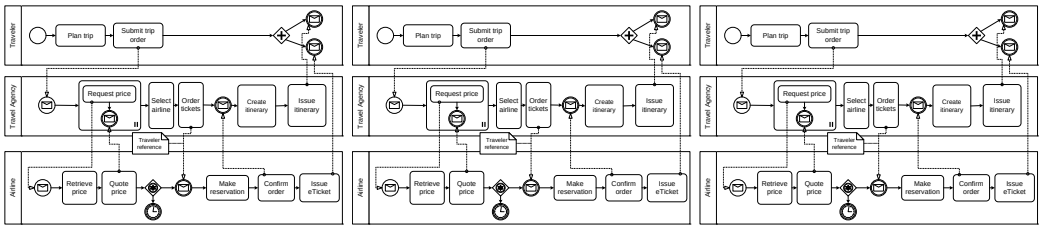


Abbildung A.2.: Die Beispiel-Choreographie. Nun etwas kleiner, damit `\textwidth` demonstriert wird. Und auch die Verwendung von alternativen Bildunterschriften für das Verzeichnis der Abbildungen. Letzteres ist allerdings nur Bedingt zu empfehlen, denn wer liest schon so viel Text unter einem Bild? Oder ist es einfach nur Stilsache?



(a) **(b)** **(c)** Subcaption if needed

Abbildung A.3.: Beispiel um 3 Abbildung nebeneinader zu stellen nur jedes einzeln referenzieren zu können. Abbildung A.3b ist die mittlere Abbildung.

zusammengefasst		**Titel**
Tabelle	wie	in
tabsatz.pdf	empfohlen	gesetzt
Beispiel	ein schönes Beispiel für die Verwendung von „multirow“	

Tabelle A.1.: Beispieltabelle – siehe http://www.ctan.org/tex-archive/info/german/tabsatz/

A.3. Tabellen

Tabelle A.1 zeigt Ergebnisse und die Tabelle A.1 zeigt wie numerische Daten in einer Tabelle representiert werden können.

A.4. Pseudocode

Algorithmus A.1 zeigt einen Beispielalgorithmus.

	Parameter 1		**Parameter 2**		**Parameter 3**		**Parameter 4**	
Bedingungen	**M**	**SD**	**M**	**SD**	**M**	**SD**	**M**	**SD**
W	1,1	5,55	6,66	,01				
X	22,22	0,0	77,5	,1				
Y	333,3	,1	11,11	,05				
Z	4444,44	77,77	14,06	,3				

Tabelle A.2.: Beispieltabelle für 4 Bedingungen (W-Z) mit jeweils 4 Parameters mit (M und SD). Hinweiß: immer die selbe anzahl an Nachkommastellen angeben.

Algorithmus A.1 Sample algorithm

procedure SAMPLE(a,v_e)
  parentHandled $\leftarrow (a = \text{process}) \vee \text{visited}(a'), (a', c, a) \in \text{HR}$
    // $(a', c'a) \in \text{HR}$ denotes that a' is the parent of a
  if parentHandled $\wedge (\mathcal{L}_{in}(a) = \emptyset \vee \forall l \in \mathcal{L}_{in}(a) : \text{visited}(l))$ **then**
    visited$(a) \leftarrow$ true
    $$\text{writes}_{\circ}(a, v_e) \leftarrow \begin{cases} \text{joinLinks}(a, v_e) & |\mathcal{L}_{in}(a)| > 0 \\ \text{writes}_{\circ}(p, v_e) & \exists p : (p, c, a) \in \text{HR} \\ (\emptyset, \emptyset, \emptyset, false) & \text{otherwise} \end{cases}$$
    if $a \in \mathcal{A}_{basic}$ **then**
      HANDLEBASICACTIVITY(a,v_e)
    else if $a \in \mathcal{A}_{flow}$ **then**
      HANDLEFLOW(a,v_e)
    else if $a =$ process **then** // Directly handle the contained activity
      HANDLEACTIVITY(a',v_e), $(a, \perp, a') \in \text{HR}$
      writes$_{\bullet}(a) \leftarrow$ writes$_{\bullet}(a')$
    end if
    for all $l \in \mathcal{L}_{out}(a)$ **do**
      HANDLELINK(l,v_e)
    end for
  end if
end procedure

Und wer einen Algorithmus schreiben möchte, der über mehrere Seiten geht, der kann das nur mit folgendem **üblen** Hack tun:

Algorithmus A.2 Description

code goes here
test2

A.5. Abkürzungen

Beim ersten Durchlauf betrug die Fehlerrate (FR) 5. Beim zweiten Durchlauf war die FR 3. Die Pluralform sieht man hier: error rates (ERs). Um zu demonstrieren, wie das Abkürzungsverzeichnis bei längeren Beschreibungstexten aussieht, muss hier noch Relational Database Management Systems (RDBMS) erwähnt werden.

Mit `\gls{...}` können Abkürzungen eingebaut werden, beim ersten Aufrufen wird die lange Form eingesetzt. Beim wiederholten Verwenden von `\gls{...}` wird automatisch die kurz Form angezeigt. Außerdem wird die Abkürzung automatisch in die Abkürzungsliste eingefügt. Mit `\glspl{...}` wird die Pluralform verwendet. Möchte man, dass bei der ersten Verwendung direkt die Kurzform erscheint, so kann man mit `\glsunset{...}` eine Abkürzung als bereits verwendet markieren. Das Gegenteil erreicht man mit `\glsreset{...}`.

Definiert werden Abkürzungen in der Datei *content ausarbeitung.tex* mithilfe von `\newacronym{...}{...}{...}`.

Mehr Infos unter: http://tug.ctan.org/macros/latex/contrib/glossaries/glossariesbegin.pdf

A.6. Definitionen

Definition A.6.1 (Title)
Definition Text

Definition A.6.1 zeigt …

A.7. Fußnoten

Fußnoten können mit dem Befehl `\footnote{...}` gesetzt werden[1]. Mehrfache Verwendung von Fußnoten ist möglich indem man zu erst ein Label in der Fußnote setzt `\footnote{\label{...}...}` und anschließend mittels `\cref{...}` die Fußnote erneut verwendet[1].

[1] Diese Fußnote ist ein Beispiel.

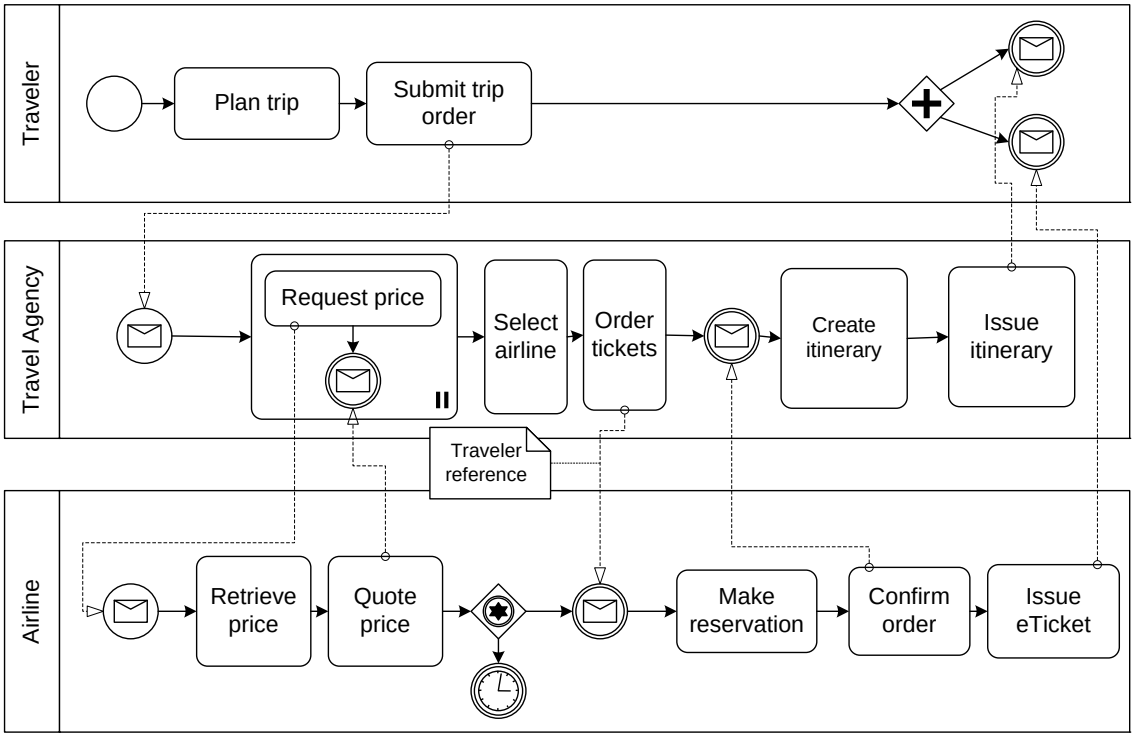


Abbildung A.4.: Beispiel-Choreographie I

A.8. Verschiedenes

Ziffern (123 654 789) werden schön gesetzt. Entweder in einer Linie oder als Minuskel-Ziffern. Letzteres erreicht man durch den Parameter `osf` bei dem Paket `libertine` bzw. `mathpazo` in `fonts.tex`.

Kapitälchen werden schön gesperrt...

I. Man kann auch die Nummerierung dank paralist kompakt halten
II. und auf eine andere Nummerierung umstellen

A.9. Weitere Illustrationen

Abbildungen A.4 und A.5 zeigen zwei Choreographien, die den Sachverhalt weiter erläutern sollen. Die zweite Abbildung ist um 90 Grad gedreht, um das Paket `pdflscape` zu demonstrieren.

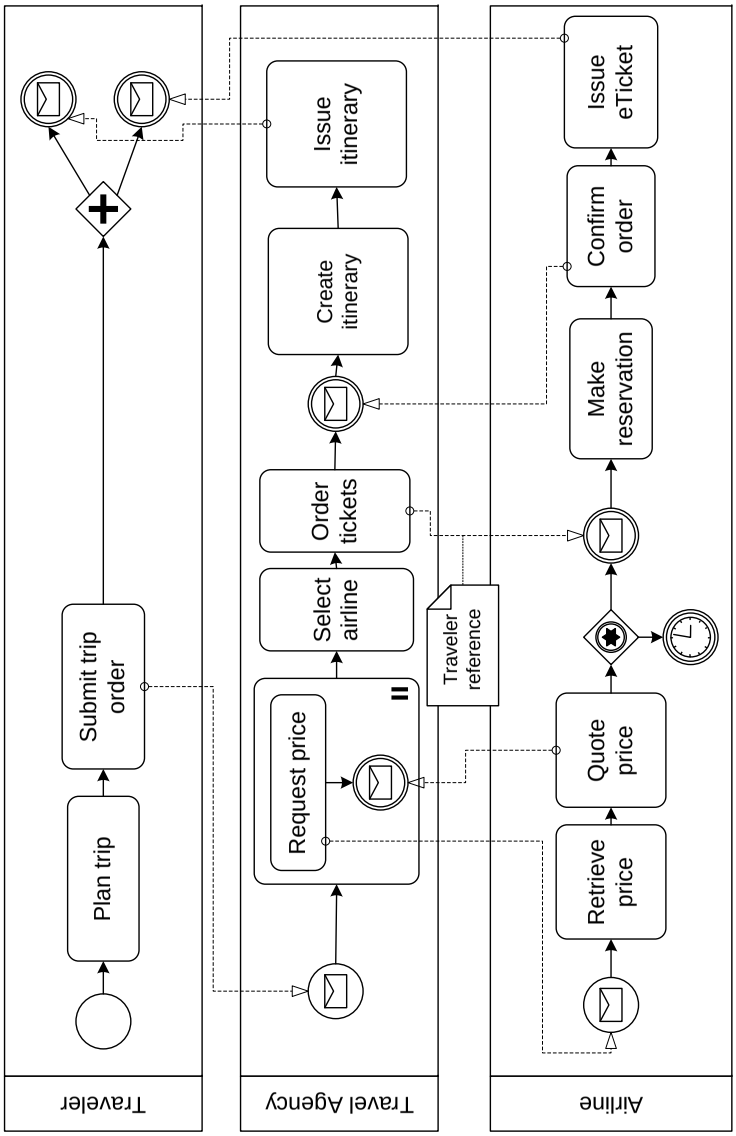


Abbildung A.5.: Beispiel-Choreographie II

A.10. Plots with pgfplots

Pgfplot ist ein Paket um Graphen zu plotten ohne den Umweg über gnuplot oder matplotlib zu gehen.

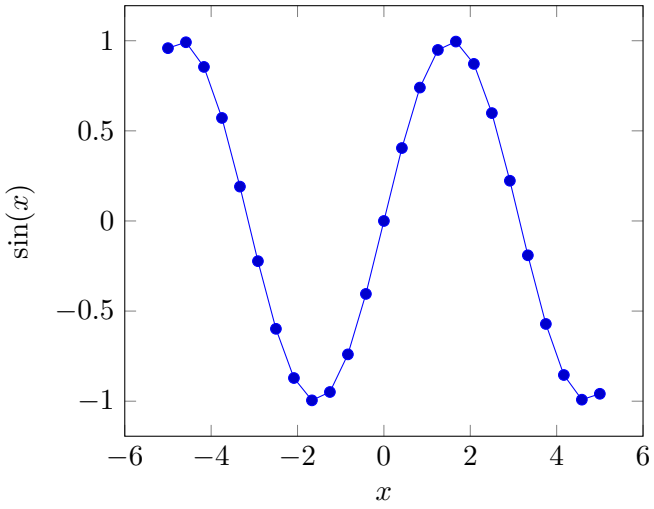


Abbildung A.6.: $\sin(x)$ mit pgfplots.

A.11. Figures with tikz

TikZ ist ein Paket um Zeichnungen mittels Programmierung zu erstellen. Dieses Paket eignet sich um Gitter zu erstellen oder andere regelmäßige Strukturen zu erstellen.

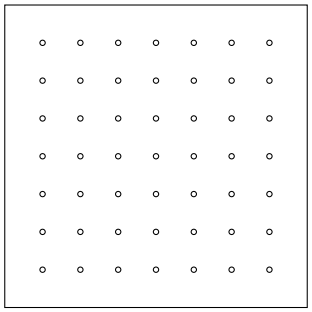


Abbildung A.7.: Eine tikz-Graphik.

Literaturverzeichnis

[BKRS13] J. Blieberger, J. Klasek, A. Redlein, G.-H. Schildt. *Informatik*. Springer-Verlag, 2013 (zitiert auf S. 4).

[Bro13] M. Broy. *Informatik Eine grundlegende Einführung: Band 1: Programmierung und Rechnerstrukturen*. Bd. 1. Springer-Verlag, 2013 (zitiert auf S. 4).

[BSI08] Bildungsstandards Informatik. *Information und Daten*. 2008. URL: https://www.informatikstandards.de/index.htm?section=standards&page_id=10 (zitiert auf S. 5).

[BSIDI11] „Internet-Sicherheit: Einführung, Grundlagen, Vorgehensweise“. In: *Bundesamt für Sicherheit in der Informationstechnik* (2011) (zitiert auf S. 5).

[Ebe94] J. Eberspächer. „Sichere Daten, sichere Kommunikation“. In: *Secure Infor-mation, Secure Communication. Datenschutz und Datensicherheit in Tele-kommunikations- und Informationssystemen. Privacy and Information Security in Communication and Information Systems, Berlin, Heidelberg, New York* (1994) (zitiert auf S. 5).

[Eck13] C. Eckert. *IT-Sicherheit: Konzepte-Verfahren-Protokolle*. Walter de Gruyter, 2013 (zitiert auf S. 4, 5).

[Fur08] S. Furnell. *Securing information and communications systems: Principles, technologies, and applications*. Artech House, 2008 (zitiert auf S. 3).

[MHTK15] N. Menz, P. Hoepner, J. Tiemann, F. Koußen. „Safety und Security aus dem Blickwinkel der öffentlichen IT“. In: *Kompetenzzentrum Öffentliche IT* (2015) (zitiert auf S. 5).

[STS18] Eric Weis. *Was ist der Unterschied zwischen IT Sicherheit und Informationssicherheit*. 2018. URL: https://www.brandmauer.de/blog/it-security/unterschied-it-sicherheit-und-informationssicherheit (zitiert auf S. 4).

[Whe13] T. Wheeler. *Offene Systeme: ein grundlegendes Handbuch für das praktische DV-Management*. Springer-Verlag, 2013 (zitiert auf S. 4).

Alle URLs wurden zuletzt am 10.01.2019 geprüft.

Erklärung

Ich versichere, diese Arbeit selbstständig verfasst zu haben. Ich habe keine anderen als die angegebenen Quellen benutzt und alle wörtlich oder sinngemäß aus anderen Werken übernommene Aussagen als solche gekennzeichnet. Weder diese Arbeit noch wesentliche Teile daraus waren bisher Gegenstand eines anderen Prüfungsverfahrens. Ich habe diese Arbeit bisher weder teilweise noch vollständig veröffentlicht. Das elektronische Exemplar stimmt mit allen eingereichten Exemplaren überein.

Ort, Datum, Unterschrift