Hochschule Bremerhaven

Fachbereich II
Management und Informationssysteme
Wirtschaftsinformatik B.Sc.

Modul
Qualitätsmanagement

Semesteraufgabe

Entwicklung einer Hausverwaltung

Vorgelegt von:	Junior Lesage Ekane Njoh	MatNr. 40128
	Steve Aguiwo II	MatNr. 40088
	Franck Majeste Dogmo Silatsa	MatNr. 00000
Vorgelegt am:	19. Februar 2025	
Dozent:in:	Prof. Dr. Karin Vosseberg	

Inhaltsverzeichnis

1 Einleitung

2 Anforderungsanalyse

2.1 Review der Anforderungen

2.2 Verbesserung der Anforderungen

3 Testkonzept

3.1 Auswahl von Testverfahren

3.2 Teststufen und Testarten

3.3 Testumgebung und Testdaten

4 Entwicklung der Testfälle

4.1 Ableitung konkreter Testfälle

4.2 Erstellung einer Testfall-Dokumentation

5 Prototypische Umsetzung der Hausverwaltung

5.1 Software-Architektur und Technologien

5.2 Implentierung

5.3 Anwendung des Testkonzepts

6 Qualitätsmanagement-Methoden in der Softwareentwicklung

6.1 Relevanz der Qualitätssicherung

6.2 Anwendung von QS-Methoden im Projekt

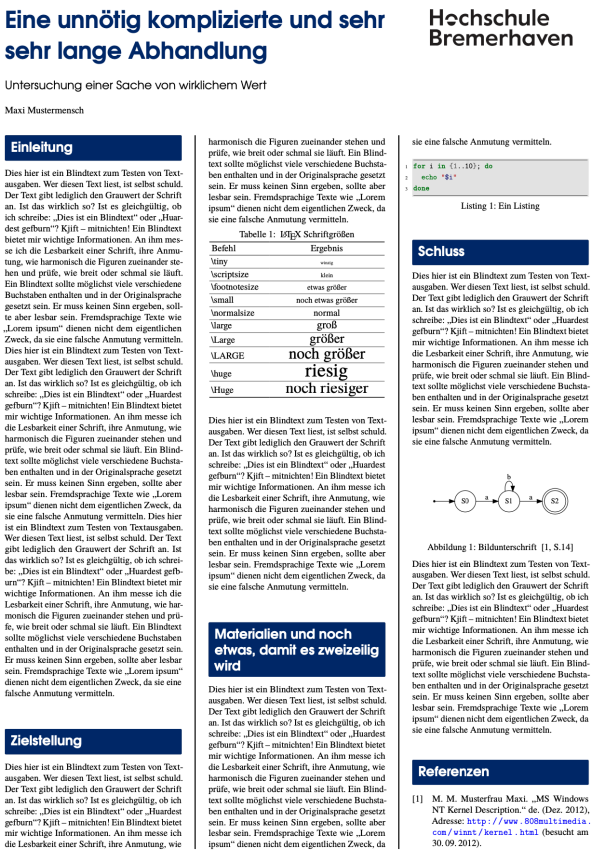
7 Einleitung

Listingverzeichnis

Anhang

I Review der Anforderungen

Review der Anforderungen



Eine unnötig komplizierte und sehr sehr lange Abhandlung

Untersuchung einer Sache von wirklichem Wert

Maxi Mustermensch

Hochschule Bremerhaven

Einleitung

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln. Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Zielstellung

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Tabelle 1: LaTeX Schriftgrößen

Befehl	Ergebnis
\tiny	winzig
\scriptsize	klein
\footnotesize	etwas größer
\small	noch etwas größer
\normalsize	normal
\large	groß
\Large	größer
\LARGE	noch größer
\huge	riesig
\Huge	noch riesiger

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Materialien und noch etwas, damit es zweizeilig wird

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

```
for i in {1..10}; do
  echo "$i"
done
```

Listing 1: Ein Listing

Schluss

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Abbildung 1: Bildunterschrift [1, S.14]

Dies hier ist ein Blindtext zum Testen von Textausgaben. Wer diesen Text liest, ist selbst schuld. Der Text gibt lediglich den Grauwert der Schrift an. Ist das wirklich so? Ist es gleichgültig, ob ich schreibe: „Dies ist ein Blindtext“ oder „Huardest gefburn“? Kjift – mitnichten! Ein Blindtext bietet mir wichtige Informationen. An ihm messe ich die Lesbarkeit einer Schrift, ihre Anmutung, wie harmonisch die Figuren zueinander stehen und prüfe, wie breit oder schmal sie läuft. Ein Blindtext sollte möglichst viele verschiedene Buchstaben enthalten und in der Originalsprache gesetzt sein. Er muss keinen Sinn ergeben, sollte aber lesbar sein. Fremdsprachige Texte wie „Lorem ipsum“ dienen nicht dem eigentlichen Zweck, da sie eine falsche Anmutung vermitteln.

Referenzen

[1] M. M. Musterfrau Maxi. „MS Windows NT Kernel Description.“ de. (Dez. 2012), Adresse: http://www.808multimedia.com/winnt/kernel.html (besucht am 30. 09. 2012).

Abbildung I.1: Poster

II Verbesserte Review der Anforderungen

Wird ein Projekt mit der Zeit relativ groß, verfügt also über eine Vielzahl von Listings, Tabellen, Abbildungen und Text, dann wird für das Kompilieren eine nicht unerhebliche Zeit benötigt. Um das ganze Dokument, inkl. aller Referenzen und Abkürzungen etc. zu bauen, ist dies auch nötig und lässt sich nicht vermeiden. Oft benötigt man jedoch nur einen kurzen Blick auf Layout oder den Umfang des bisher geschriebenen und möchte nicht das Dokument in seiner Gesamtheit kompilieren.
Für diese Anforderungen gibt es sog. `subfiles`, diese ermöglichen es, nur einen kleinen Teil

des Dokumentes zu kompilieren und als PDF zu betrachten. Dabei kommt es zu gewissen Einschränkungen, so werden Seitenzahlen, Referenzen und Abkürzungen nicht korrekt dargestellt, dennoch erhält man einen guten Blick auf das Layout. Um mit `subfiles` zu arbeiten, muss einiges beachtet werden. So muss statt des `\input{}` das Kommando `\subfile{}` genutzt werden. Darüber hinaus müssen alle `.tex`-Dateien mit einer eigenen `documentclass` ausgestattet werden. In der `hauptdatei.tex` sieht dies dann folgendermaßen aus.

```
\defverbatim[colored]\CodeOne{
  \begin{minted}{python}
    import numpy as np
    import pylab as pl

    def f_x(x):
      return np.exp(x)+x**2-5*x

    def approx_f(x):
      return 1 -4*x +3./2*x**2

    xvals = np.arange(-4,4,0.1)
    fx_vals = [f_x(x) for x in xvals]
    approx_vals = [approx_f(x) for x in xvals]

    pl.plot(xvals,fx_vals)
    pl.plot(xvals,approx_vals)

    pl.show()
\end{minted}
\captionof{listing}{Example Code}
}

\section{Code Example}
\begin{frame}{Code Example}
  Some Text on first Frame~\autocite{gregg2013systems}
  \CodeOne
  \vspace{0.5cm}
\end{frame}
```

Listing II.1: Einbinden von subfiles

Jedes Subfile muss dann mit der passenden `documentclass` beginnen.

```
\defverbatim[colored]\CodeOne{
  \begin{minted}{python}
    import numpy as np
    import pylab as pl

    def f_x(x):
      return np.exp(x)+x**2-5*x

    def approx_f(x):
      return 1 -4*x +3./2*x**2

    xvals = np.arange(-4,4,0.1)
    fx_vals = [f_x(x) for x in xvals]
    approx_vals = [approx_f(x) for x in xvals]

    pl.plot(xvals,fx_vals)
    pl.plot(xvals,approx_vals)

    pl.show()
\end{minted}
\captionof{listing}{Example Code}
}

\section{Code Example}
\begin{frame}{Code Example}
  Some Text on first Frame~\autocite{gregg2013systems}
  \CodeOne
  \vspace{0.5cm}
\end{frame}
```

Listing II.2: Erstellen von subfiles

Durch die eigene `documentclass`, welche wiederum auf die `hauptdatei.tex` verweist und die Angabe von `\begin` bzw. `\end{document}` kann die entsprechende `tex`-Datei separat kompiliert werden. Dafür kann das Skript `quickbild.sh` genutzt werden, welches als Argument den Namen der zu kompilierenden Datei erwartet und ein PDF erzeugt.
Abbildungen, Listings, Tabellen können wie gewohnt genutzt werden. Wird `buildlualatex.sh` ausgeführt, baut nach wie vor das gesamte Dokument inkl. aller Subfiles und Referenzen.

III Testfälle

Wie im Listing gezeigt, wird das Listing außerhalb des Frames definiert und in Zeile 27 eingefügt. `CodeOne` ist dabei der zugewiesene Name unter dem das Listing eingefügt werden kann. Die Definition erfolgt in Zeile 1, der Name ist frei wählbar. Das Listing so einzubinden ist nicht optimal, jedoch bleibt aktuell keine andere Möglichkeit, wenn mit `minted` gearbeitet wird.
Das Inhaltsverzeichnis und die Referenzen werden automatisch erstellt, alle Einstellungen können natürlich innerhalb der Vorlage angepasst werden.
Obiges Listing würde mit der Vorlage folgendermaßen aussehen. Wie immer finden sich alle Beispiele auch in den jeweiligen Vorlagen.

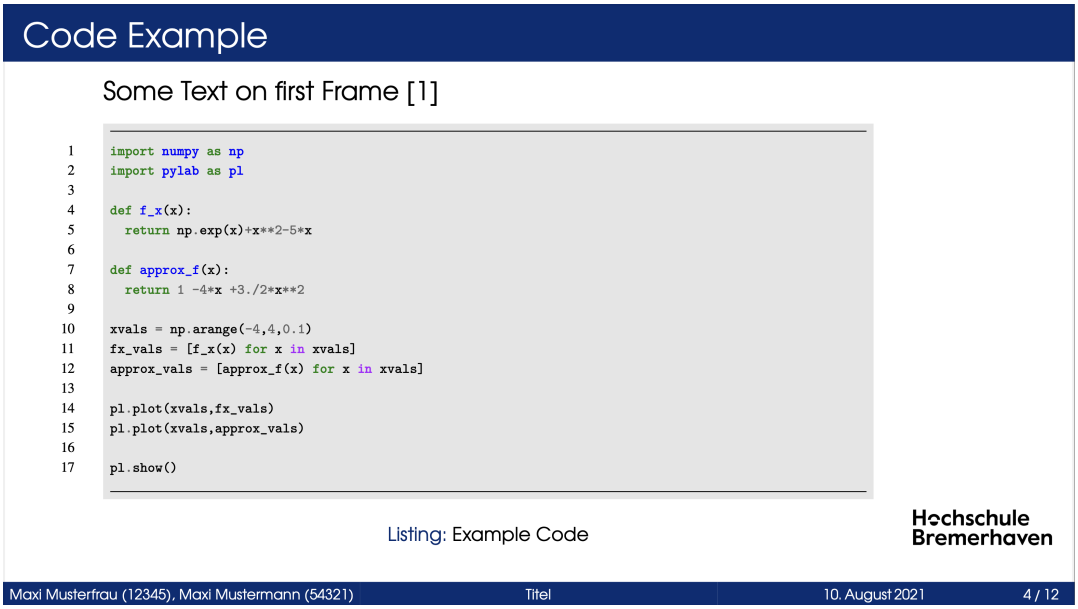


Abbildung III.1: Frame mit Listing

IV Quickbuild

Wird ein Projekt mit der Zeit relativ groß, verfügt also über eine Vielzahl von Listings, Tabellen, Abbildungen und Text, dann wird für das Kompilieren eine nicht unerhebliche Zeit benötigt. Um das ganze Dokument, inkl. aller Referenzen und Abkürzungen etc. zu bauen, ist dies auch nötig und lässt sich nicht vermeiden. Oft benötigt man jedoch nur einen kurzen Blick auf Layout oder den Umfang des bisher geschriebenen und möchte nicht das Dokument in seiner Gesamtheit kompilieren.
Für diese Anforderungen gibt es sog. `subfiles`, diese ermöglichen es, nur einen kleinen Teil des Dokumentes zu kompilieren und als PDF zu betrachten. Dabei kommt es zu gewissen Einschränkungen, so werden Seitenzahlen, Referenzen und Abkürzungen nicht korrekt dargestellt, dennoch erhält man einen guten Blick auf das Layout. Um mit `subfiles` zu arbeiten, muss einiges beachtet werden. So muss statt des `\input{}` das Kommando `\subfile{}` genutzt werden. Darüber hinaus müssen alle `.tex`-Dateien mit einer eigenen `documentclass` ausgestattet werden. In der `hauptdatei.tex` sieht dies dann folgendermaßen aus.

```
\defverbatim[colored]\CodeOne{
  \begin{minted}{python}
    import numpy as np
    import pylab as pl

    def f_x(x):
      return np.exp(x)+x**2-5*x

    def approx_f(x):
      return 1 -4*x +3./2*x**2

    xvals = np.arange(-4,4,0.1)
    fx_vals = [f_x(x) for x in xvals]
    approx_vals = [approx_f(x) for x in xvals]

    pl.plot(xvals,fx_vals)
    pl.plot(xvals,approx_vals)

    pl.show()
\end{minted}
\captionof{listing}{Example Code}
}

\section{Code Example}
\begin{frame}{Code Example}
```

```
  Some Text on first Frame~\autocite{gregg2013systems}
  \CodeOne
  \vspace{0.5cm}
\end{frame}
```

Listing IV.1: Einbinden von subfiles

Jedes Subfile muss dann mit der passenden `documentclass` beginnen.

```
\defverbatim[colored]\CodeOne{
  \begin{minted}{python}
    import numpy as np
    import pylab as pl

    def f_x(x):
      return np.exp(x)+x**2-5*x

    def approx_f(x):
      return 1 -4*x +3./2*x**2

    xvals = np.arange(-4,4,0.1)
    fx_vals = [f_x(x) for x in xvals]
    approx_vals = [approx_f(x) for x in xvals]

    pl.plot(xvals,fx_vals)
    pl.plot(xvals,approx_vals)

    pl.show()
\end{minted}
\captionof{listing}{Example Code}
}

\section{Code Example}
\begin{frame}{Code Example}
  Some Text on first Frame~\autocite{gregg2013systems}
  \CodeOne
  \vspace{0.5cm}
\end{frame}
```

Listing IV.2: Erstellen von subfiles

Durch die eigene `documentclass`, welche wiederum auf die `hauptdatei.tex` verweist und die Angabe von `\begin` bzw. `\end{document}` kann die entsprechende `tex`-Datei separat kompiliert werden. Dafür kann das Skript `quickbild.sh` genutzt werden, welches als Argument den Namen der zu kompilierenden Datei erwartet und ein PDF erzeugt.
Abbildungen, Listings, Tabellen können wie gewohnt genutzt werden. Wird `buildlualatex.sh` ausgeführt, baut nach wie vor das gesamte Dokument inkl. aller Subfiles und Referenzen.

V Prototyp-Dokumentation

Passend zu diesem Template existieren noch zwei weitere Vorlagen für Poster. Diese unterscheiden sich lediglich durch die Ausrichtung der Seite und sind ansonsten identisch. Der Aufbau der Dateien und Projektstruktur ist weitestgehend identisch zu dieser Vorlage, es gilt lediglich auf einige kleine Besonderheiten zu verweisen.
In der sog. `multicols` Umgebung ist es nicht möglich, mit Floatumgebungen zu arbeiten, dies ist für fast alle Beispiele aus dieser Anleitung irrelevant, lediglich Grafiken müssen anders eingebunden werden. Dazu finden sich jedoch Beispiele in den Vorlagen. Ebenso ist ein Teil der Präambel in eine separate Datei ausgelagert, in dieser sollten auch nicht ohne weiteres Änderungen vorgenommen werden. Wie in diesem Template, müssen innerhalb der Präambel Zitierstil, sowie Autor:innen und Titel gesetzt werden.
Die Vorlagen können auf meiner Webseite sowohl angeschaut, als auch heruntergeladen werden, alle Hinweise zum Kompilieren gelten für die Poster ebenso. Ergebnis dieser Vorlage ist z. B. folgendes Poster:

Selbstständigkeitserklärung

Wir versichern, die von uns vorgelegte Arbeit selbstständig verfasst zu haben. Alle Stellen, die wörtlich oder sinngemäß aus veröffentlichten oder nicht veröffentlichten Arbeiten anderer entnommen sind, habe ich als entnommen kenntlich gemacht. Sämtliche Quellen und Hilfsmittel, die wir für die Arbeit benutzt haben, sind angegeben. Die Arbeit haben wir mit gleichem Inhalt bzw. in wesentlichen Teilen noch keiner anderen Prüfungsbehörde vorgelegt.

Bremerhaven, den 19. Februar 2025 Unterschrift:

Junior Leage EKane Njoh, Franck Majeste
Silatsa Dogmo, Steve Aguiwo II