DHBW
Duale Hochschule
Baden-Württemberg
Karlsruhe

GUI CODING

Lars Briem

(briem.lars@googlemail.com)

Duale Hochschule Baden Württemberg - Standort Karlsruhe

Verlagswesen

- Konkrete Beschreibung der Gestalt einzelner Elemente
- Förderung eines einheitlichen Erscheinungsbildes
- Corporate Identity
- Inhalt
  - Papierart / -format
  - Druckart
  - Schriftart
  - Layout
  - Gliederung
  - ...

COLORS

ON-SCREEN

Computers, televisions and other electronic displays use combinations of red, green and blue to simulate a large gamut of colors. Many displays are able to show millions of colors and can often display colors more vivid than printers.

ON-PRESS

Standard printing process, from desktop ink-jets to industrial offset lithography use combinations of cyan, magenta, yellow and black inks to simulate a large gamut of colors. This method is not able to produce the full color spectrum and lacks strongly-saturated green and orange.

CUSTOM INKS

The Pantone Matching System by Pantone, Inc. is a proprietary method of mixing inks to exacting standards. It allows highly-accurate color reproduction and is key in unifying corporate communications. PMS numbers, as listed at right, are to be used when designing and printing materials.

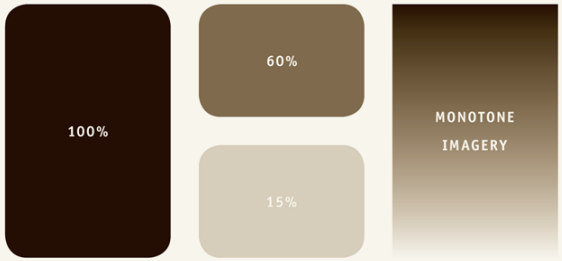
giraffe project

PRIMARY COLOR

The primary color is an off-black that screens to a subtle brownish color. It is to be used in one of the four manners shown at left.

This color allows for adequate contrast in type and photography while retaining approachability and warmth.

PMS 412
C 0
M 30
Y 66
K 98



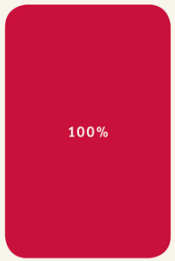
DO NOT

- Mix with other colors
- Screen at percentages other than sixty or fifteen

ACCENT COLOR

The bold and warm accent color provides variation to printed materials that builds hierarchy, captures attention and evokes the bold nature of GIRAFFE HEROES.

PMS 193
C 0
M 100
Y 66
K 13



DO NOT

- Mix with other colors
- Use anything but 100%

Gestaltungsrichtlinien - Softwareentwicklung

- Inhalt
  - Fenster / Dialoge
  - Interaktion
  - Text
  - GUI Komponenten
  - Layout / Farben / Schriften
  - $\Rightarrow$ Im Wesentlichen Blooper Kategorien
- Herstellerspezifisch
  - Microsoft
  - Apple
  - Android (Google)
  - ...

Gestaltungsrichtlinien - Softwareentwicklung

- Plattformabhängigkeit
- Typische Vorgaben
  - Ist das die richtige Komponente?
  - Design Konzepte
  - Verwendungsarten
  - Richtlinien
  - Größe / Abstände / Farben
  - Beschreibung / Dokumentation

Plattformabhängigkeit

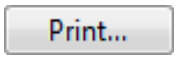
- Unterscheidungsmerkmale
  - Eingabemethode
  - Eingabegenauigkeit
  - Bildschirmgröße
- Desktop
  - Windows / Linux / Mac OS
  - Optimiert für Maus / Tastatur
  - Großer Bildschirm (noch kleine Auflösung)
- Mobile
  - Optimiert für Touch
  - Kleiner Bildschirm (hohe Auflösung)
- Web
  - Kombination aus beidem + eigene Regeln

Ist das die richtige Komponente?

- Fragen / Beschreibung zur Entscheidung
- Sortiert nach Wichtigkeit
- Beispiel Button
  - Wird eine Aktion direkt angestoßen?
  - Wäre ein Link eine bessere Alternative?
  - Wäre eine Kombination aus Auswahlelementen und generischen Buttons besser?

Design Konzepte

- Unterschiede der Aktion
- Beispiel Button
  - Auslassungszeichen kennzeichnet zusätzlich benötigte Informationen

  - Bei direkter Aktion, keine Auslassungszeichen

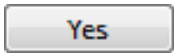
- Beispiel Fortschrittsanzeige
  - Länge und aktueller Fortschritt bekannt

    
  - Unbekannte Länge

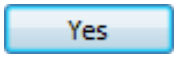
    

Verwendungsarten

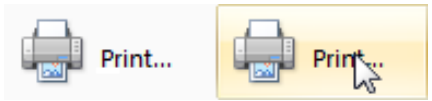
- Unterschiede bei der Bedienung
- Unterschiede im Aussehen
- Auswahl hinterlegter Aktion
- Beispiel Button
  - Normaler Button

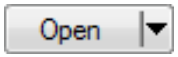
  - Standard / Fokusierter Button

  - Leichtgewichtiger Button

  - Split Button

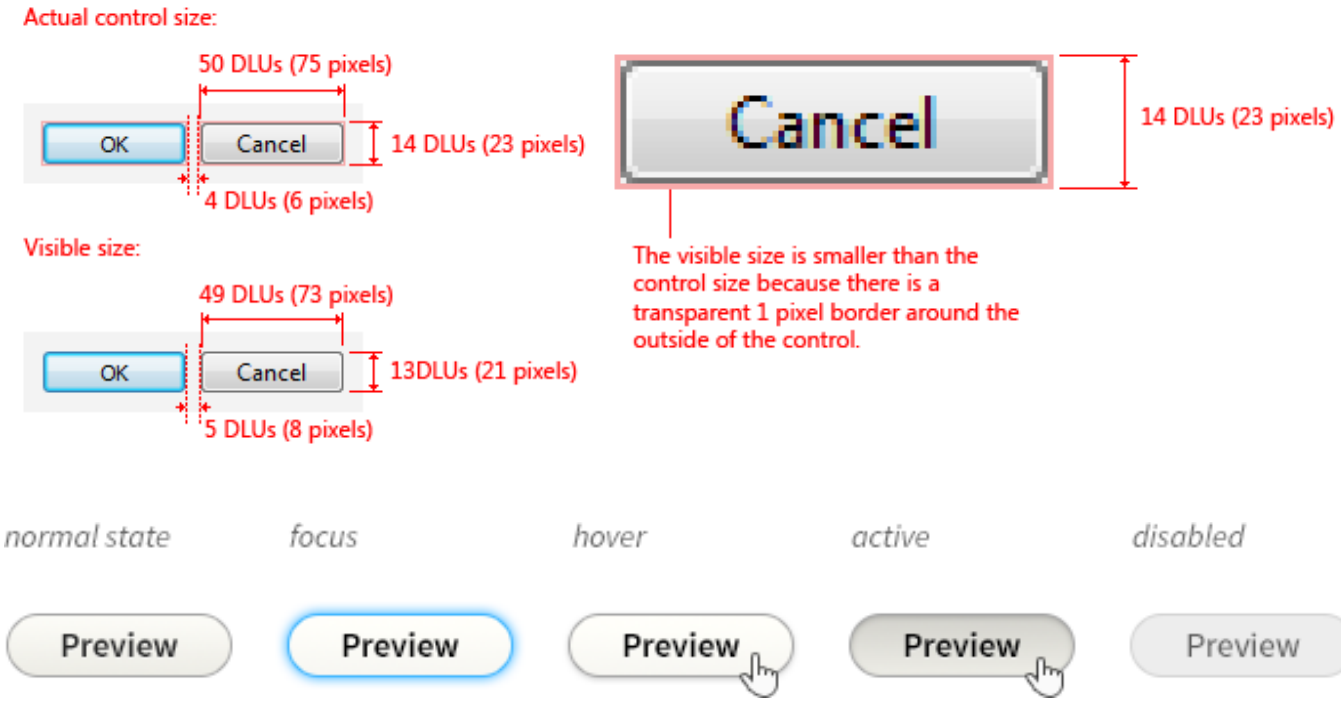
    

Richtlinien

- Vorgaben zur Verwendung
- Anzuzeigende Informationen
- Beziehungen zwischen Komponenten
- Animationen / Positionierung / Sichtbarkeit
- Auflistung / Sortierung von Kindelementen
- Detaillierte Anweisungen zu unterschiedlichen Verwendungsarten

Größe / Abstände / Farben

- Detaillierte Beschreibung
  - Größen
  - Abstände
- Beispiel Button



Beschreibung / Dokumentation

- Art und Weise der Beschriftung
- Umfang der Beschriftung
- Kombination aus Grafik und Text
- Terminologie in der Dokumentation / Hilfe

Motivation

Warum soll GUI getestet werden?

- Software muss getestet werden
- GUI ist Teil der Software
- GUI kann Fehler enthalten

$\Rightarrow$ GUI sollte getestet werden

Ziele

- Funktion von GUI Komponenten gewährleisten
- Zusammenspiel verschiedener Komponenten gewährleisten
- Test automatisieren (Continuous Integration)
  - Bei allen Änderungen
  - Auf allen Plattformen

GUI Tests

Funktionaler Test von GUI Komponenten

- Komponenten Tests
  - Testet Komponente einzeln
- Integrationstests
  - Testet Integration verschiedener Komponenten
- Akzeptanztests

$\Rightarrow$ Vergleichbar mit Unittests für Quellcode

$\Rightarrow$ Ähnliche Anforderungen wie an Unittests

Arten

- Manuelles Testen
- Record und Replay
- Skriptbasiertes Testen
- Automatisiertes Testen

Manuelles Testen

Reale Person testet das Programm von Hand

- Protokoll mit Abfolge der Aktionen
- Ausführen einzelner Aktionen
- Überprüfen der Vorgaben
- Fehler oder Erfolg wird protokolliert

Manuelles Testen

+ Erkennung „sinnvoller“ Abweichungen der Vorgaben
+ Plausibilitätsprüfung der Vorgaben
+ „Gutes Aussehen“ bzw. Konsistenz überprüfbar
+ Schnell bei einmaliger Ausführung

− Extrem zeitaufwändig
− Extrem teuer
− Evtl. keine Wahrnehmung von Details
− Erfüllt Anforderung „automatisch“ nicht

Probleme skriptbasierter / automatischer Tests

- Suchen / Finden der GUI Elemente
- Interaktion mit den GUI Elementen
- Überprüfen der Ergebnisse
- Feststellen von Änderungen / Ereignissen
- Protokollierung von Fehlern

Suchen / Finden der GUI Elemente

- Position auf dem Bildschirm
- Beschriftung
- Elternelemente
- Attribute (z.B. CSS Klassen)
- Eindeutige Bezeichnung (z.B. fx:id)

Suchen / Finden der GUI Elemente

- Position auf dem Bildschirm
- Beschriftung
- Elternelemente
- Attribute (z.B. CSS Klassen)
- **Eindeutige Bezeichnung** (z.B. fx:id)

Interaktion mit den GUI Elementen - Robot

Zentrale Komponente jedes GUI Testtools

- Führt Benutzeraktionen aus
- Maussteuerung
  - Meist absolute Bildschirmkoordinaten
  - Move(x,y)
- Tastatureingaben
  - Einzelne Tastendrücke
  - Sequenzen von Tastendrücken
  - Tastenkombinationen
  - Press(„T“)
  - Press(Enter)
- Aufnahme von Screenshots

Überprüfen der Ergebnisse

- Vergleich der GUI mit Screenshot
  - Test bei identischem Screenshot bestanden
  - Aufwendig
  - Fehlerbehaftet
- Attribute der GUI Komponente auslesen
  - Möglichkeit zum Auslesen der Attribute notwendig
  - Einfache und präzise Möglichkeit
- Direkter Zugriff auf Businessmodell
  - Möglichkeit für Zugriff auf Businessmodell
  - Überprüfen der Effekte
  - Fehler in Businessmodell führt zu Fehler in GUI Test

Probleme skriptbasierter / automatischer Tests

Feststellen von Änderungen / Ereignissen

- Basierend auf Ereignissen
  - Beobachter an GUI Komponente registrieren
- Polling
  - Mehrfaches Abfragen eines Wertes
  - Abbruch bei Timeout

Protokollierung von Fehlern

- Wie bei Unit Tests
  - Bestanden
  - Fehlgeschlagen
- Screenshot

Record und Replay

- Älteste und einfachste Art der automatischen Tests
- Testerzeugung
  1. Aufzeichnung starten
  2. Programm starten
  3. Aktionen ausführen (Maus, Tastatur)
  4. Programm beenden
  5. Aufzeichnung beenden
  6. Tests / Überprüfungen definieren
- Erzeugter Test
  - Testskript
  - Grafischer Testablauf

Record und Replay - Beispiel Test

```
For i = 1 to 10000
    'VERIFY + AND * OPERATIONS ON THE CALCULATOR
    Window("Calculator").WinEdit("Edit").Set(i)
    Window("Calculator").WinButton("+").Click
    Window("Calculator").WinEdit("Edit").Set(i)
    Window("Calculator").WinButton("=").Click
    intResult_1 =   Window("Calculator").WinEdit("Edit").GetROProperty("text")

    Window("Calculator").WinEdit("Edit").Set(i)
    Window("Calculator").WinButton("*").Click
    Window("Calculator").WinEdit("Edit").Set("2")
    Window("Calculator").WinButton("=").Click
    intResult_2 =   Window("Calculator").WinEdit("Edit").GetROProperty("text")

    If intResult_1 <> intResult_2 Then
        Reporter.ReportEvent micFail, "VERIFY", "RESULT INCONSISTENT FOR DATA :" &i
    End If
Next
```

Record und Replay

Bestandteile

- Schrittweise Anleitung
- Angabe der Aktionen mit
  - Absolute Pixelkoordinaten
  - GUI Komponenten (Objekte suchen)
- Manuell eingefügte Überprüfung
- Ausgabe im Fehlerfall

Record und Replay

- Testenablauf
  1. Testsoftware starten
  2. Test laden
  3. Ausführung der gespeicherten Schritte
  4. Aktionen und Fehler protokollieren
  5. Ergebnis abspeichern
- Fehlerfall
  - Viele Informationen sammeln (Screenshot, . . .)
  - Test abbrechen
  - Test weiter ausführen

Record und Replay

+ Einfach zu bedienen
+ Schnell für kleine Tests
+ Wenig Programmierkenntnisse notwendig

− Unübersichtlich bei komplexen Tests
− Nur manuelle Wiederverwendung
− Viel Redundanz
− Änderungen evtl. schwierig

Skriptbasiertes Testen

- Ähnlich wie Record und Replay
- Programmierer erstellt Skript

\+ Einfacher zu Warten

\+ Zusammenfassen verschiedener Aktionen

\+ Wiederverwenden von Standardaktionen

\+ Gleiche Programmiersprache wie GUI

− GUI bei Erstellung nicht unbedingt sichtbar

Automatisiertes Testen

- Matrix Tests
- Monkey Tests
- Künstliche Intelligenz

Matrix Tests

Einsatzbereich

- Oft großer Datenbereich für Eingabe
  - Zahlen beim Taschenrechner
  - Alle Eingaben testen nicht / nur schwer möglich
- Kombinationen verschiedener Eingaben

Ablauf

- Auswahl der Parameter
- Definition Parameterwerte
- Werte und Kombinationen in Tabelle zusammenfassen

Matrix Tests - Beispiel Taschenrechner

Addieren	**0**	**1**	**2**	**Große Zahl**
0	0	1	2	Große Zahl
1		2	3	Große Zahl + 1
2			4	Überlauf
Große Zahl				Überlauf

Integer: Große Zahl = $2^{31} - 2$

Long: Große Zahl = $2^{63} - 2$

$\Rightarrow$ Erweiterung auf negative Zahlen möglich

Matrix Tests - Beispiel Taschenrechner

+ Einfache Abdeckung vieler Kombinationen
+ Reduktion der Redundanz

− Exponentielle Laufzeit
  - k Parameter
  - n Werte pro Parameter
  - n^k Kombinationen

Monkey Tests

- Simulation realer Anwender
- Kontrollierter Zufallsgenerator
- Zufällige Aktionsreihenfolge / -ausführung
  - Menü
  - Shortcut
  - Kontextmenü
- Suche nach Fehlern
- Aktionsfolge abspeichern

$\Rightarrow$ Bei gefundenem Fehler: Aktionsfolge als dauerhaften Test hinzufügen

- Vergleichbar mit Robotik
  - Menge von Zuständen
  - Menge von Aktionen
  - Startzustand
  - Endzustand
  - Graphsuche von Start- zu Endzustand
- Erzeugung von Testfällen
  - Graphsuche (Tiefen-/Breitensuche, A*, . . .)
  - Evolutionäre Algorithmen
  - . . .

Künstliche Intelligenz

+ Generierung vieler Testfälle möglich
+ Viele Wege zum Ziel testbar

− Komplexe Definition von Zuständen / Aktionen
− Komplexe Suche / Optimierung

$\Rightarrow$ Bisher eher selten verwendet

Vor- und Nachteile von GUI Tests

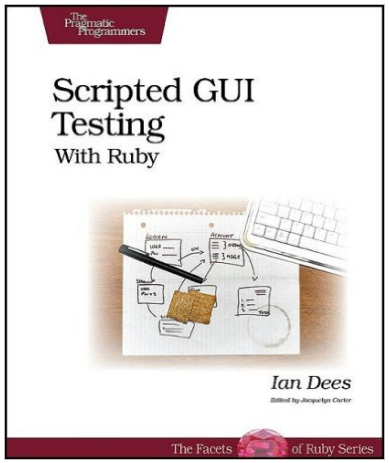
+ GUI wie Code während dem Buildvorgang testen
+ Einmal Aufwand für kontinuierliche Tests
+ 1 Test für unterschiedliche Plattformen
+ Monkey Test vergleichbar mit normalem Benutzer
+ Screenshots

− Ergebnis genau spezifizieren
− Einmalige Überprüfung durch Mensch schneller

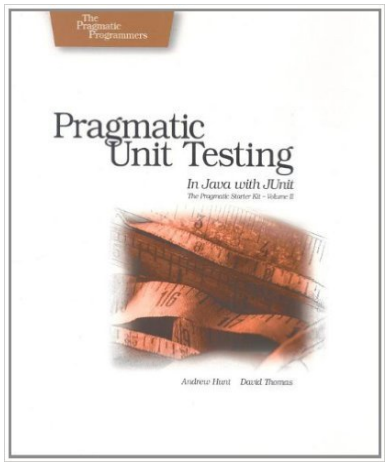
Probleme verglichen mit Unittests

- Deutlich längere Laufzeit
- Asynchrone Abarbeitung
  - Immer Multithreading
  - Timeouts verwenden
- Grafischer Desktop sinnvoll / notwendig
- Eingabe während Ausführung blockiert

Literatur



- Scripted GUI Testing
  - Ian Dees
  - The Pragmatic Programmers
  - ISBN: 978-1934356180



- Pragmatic Unit Testing
  - Andrew Hund und David Thomas
  - The Pragmatic Programmers
  - ISBN: 978-0974514017

Weitere Quellen

- Internet
  - amazon.de
  - gettyimages.de
  - github.com/TestFX/TestFX
  - microsoft.com
  - wikipedia.org