Ein Werkzeug zum manuellen und automatisierten Erfassen von grafischen Benutzungsschnittstellen für ATOSj

GUIwalker 0.5 - GUIanalyzer

GUIwalker 0.5

Überblick

- Einleitung/Wiederholung
- Modelle
- Algorithmen
- Probleme

GUIwalker 0.5

Überblick

- Einleitung/Wiederholung
- Modelle
- Algorithmen
- Probleme

Einleitung

Aufgabe

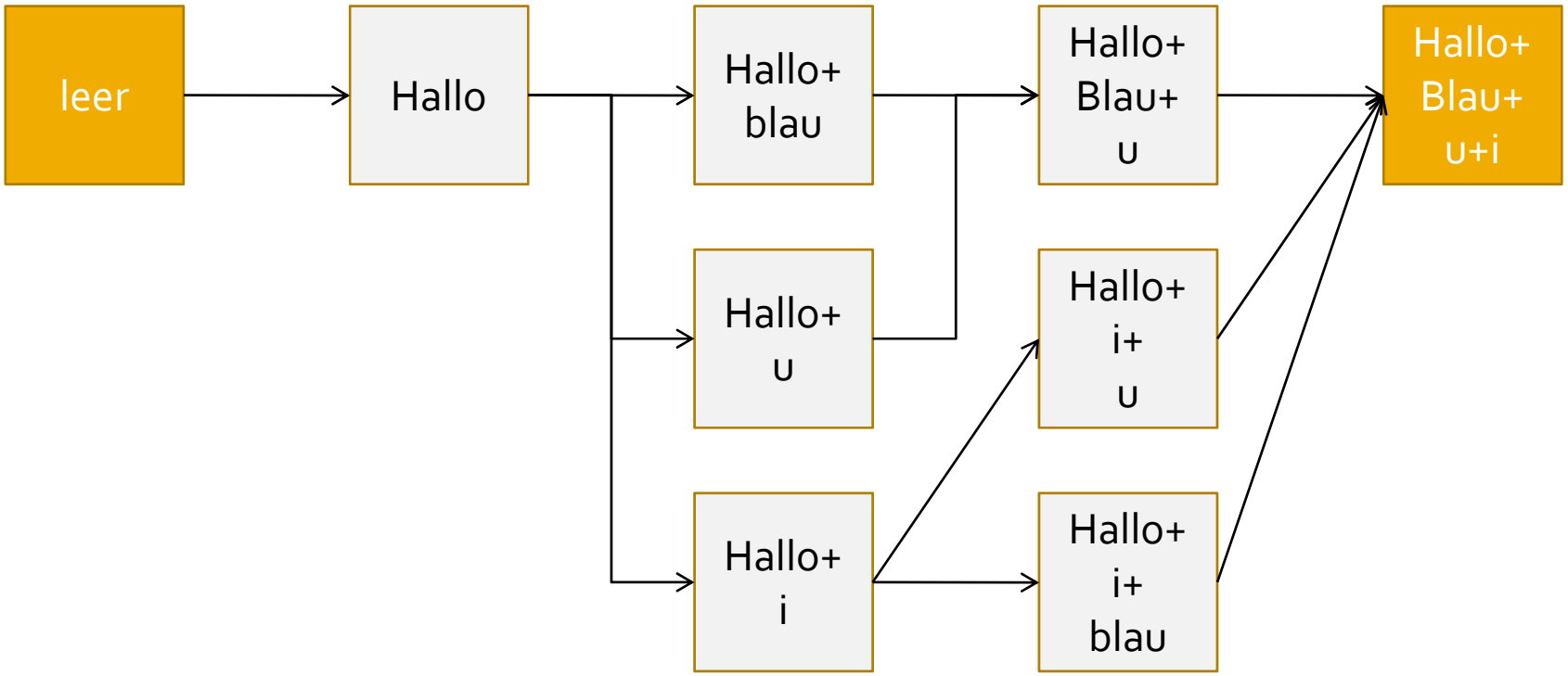
- Entwicklung eines Tools zur automatischen Erfassung von grafischen Benutzungsschnittstellen und automatisiertes Ableiten von relevanten Testfällen

Einleitung

Wie?

- Erfassen der GUI als Modell
  - Struktur der GUI (GUIModell)
  - Zustand der GUI (GUIState)
- Definition von Anfangs- und Endzustand
- Ermittlung der Zwischenzustände durch
  - Anwendung von Ereignissen (Aktionen) auf einen Zustand beginnend mit Anfangszustand

Testfallgenerierung



GUIwalker 0.5

Überblick

- Einleitung/Wiederholung
- Modelle
- Algorithmen
- Probleme

GUIwalker 0.5

Modelle

- GUIModell
  - Aufrufgraph
  - Fenstermodell/Widgets
- GUIState-Modell

GUIModell (Struktur)

Bestandteile

- Fensteraufrufgraph
- Widgets
  - Eigenschaften
  - Vater-/Kindbeziehungen
- Ereignisse
  - Vorbedingungen
  - Effekte

GUIwalker 0.5

Modelle

- GUIModell
  - Aufrufgraph
  - Fenstermodell/Widgets
- GUIState-Modell

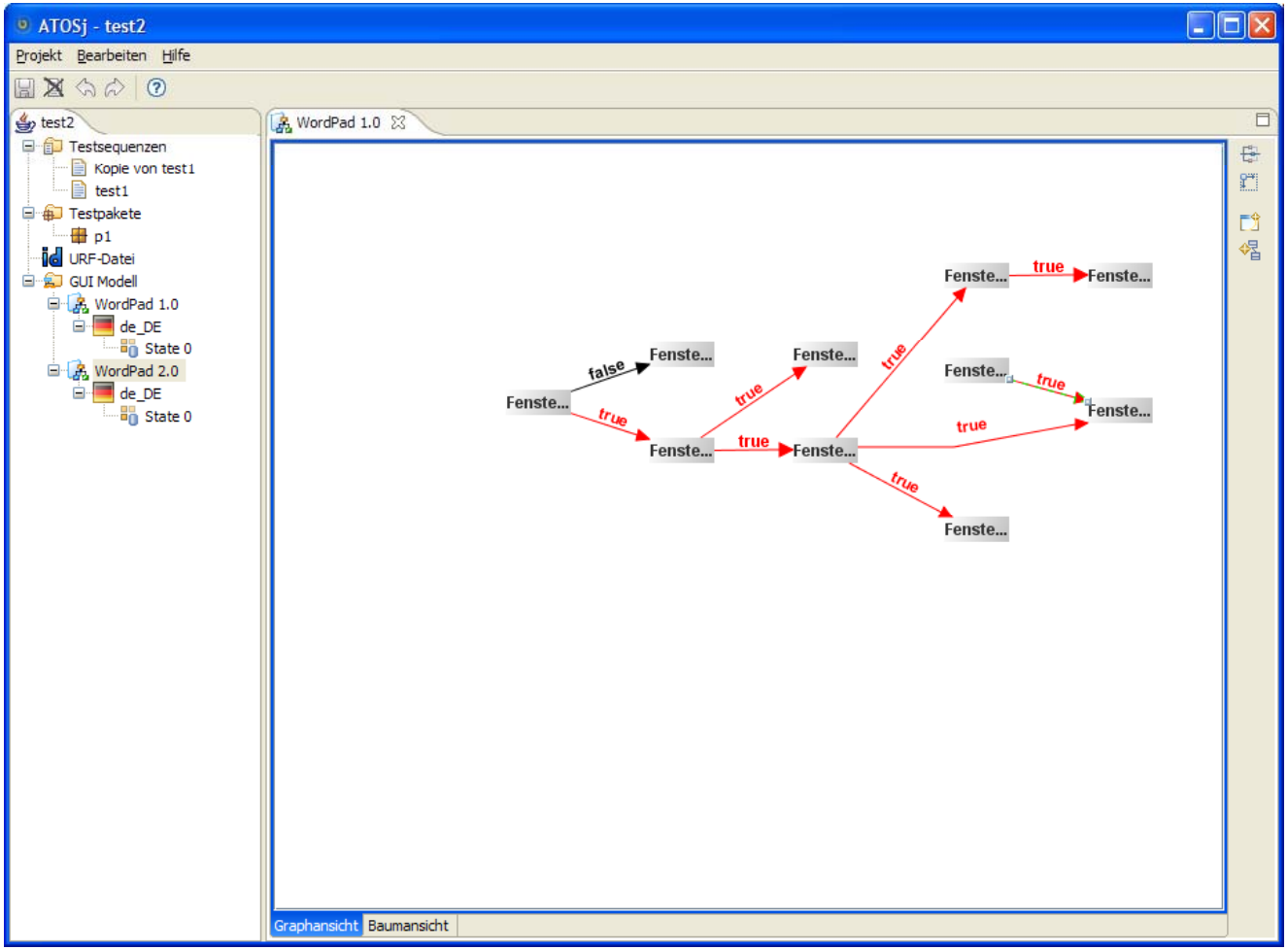
Aufrufgraph

Zweck

- Speichert Informationen über
  - Fenster im Modell (Knoten des Graphen)
  - Fenster, die beim Start des Testobjekts geöffnet sind (Startpunkt) (keine Eingangskante)
  - Zu welchen Fenstern man gelangen kann, wenn man von einem bestimmten ausgeht (Aufrufreihenfolge) (Kanten)
  - Ist der Aufruf des Fensters modal (Eigenschaft der Kante)

GUIModell

Aufrufgrapheditor



GUIModell - Aufrufgraph

Zweck

- Wichtig um Komplexität zu reduzieren
  - Abstrahiert vom Aufbau eines Fensters
  - Oberste Abstraktionsebene
  - Gibt Pfad von einem Fenster zu einem anderen an ohne Ereignisse durchsuchen zu müssen

GUIwalker 0.5

Modelle

- GUIModell
  - Aufrufgraph
  - Fenstermodell/Widgets
    - Eigenschaftsdefinition
    - Ereignisdefinition
- GUIState-Modell

GUIModell - Fenstermodell

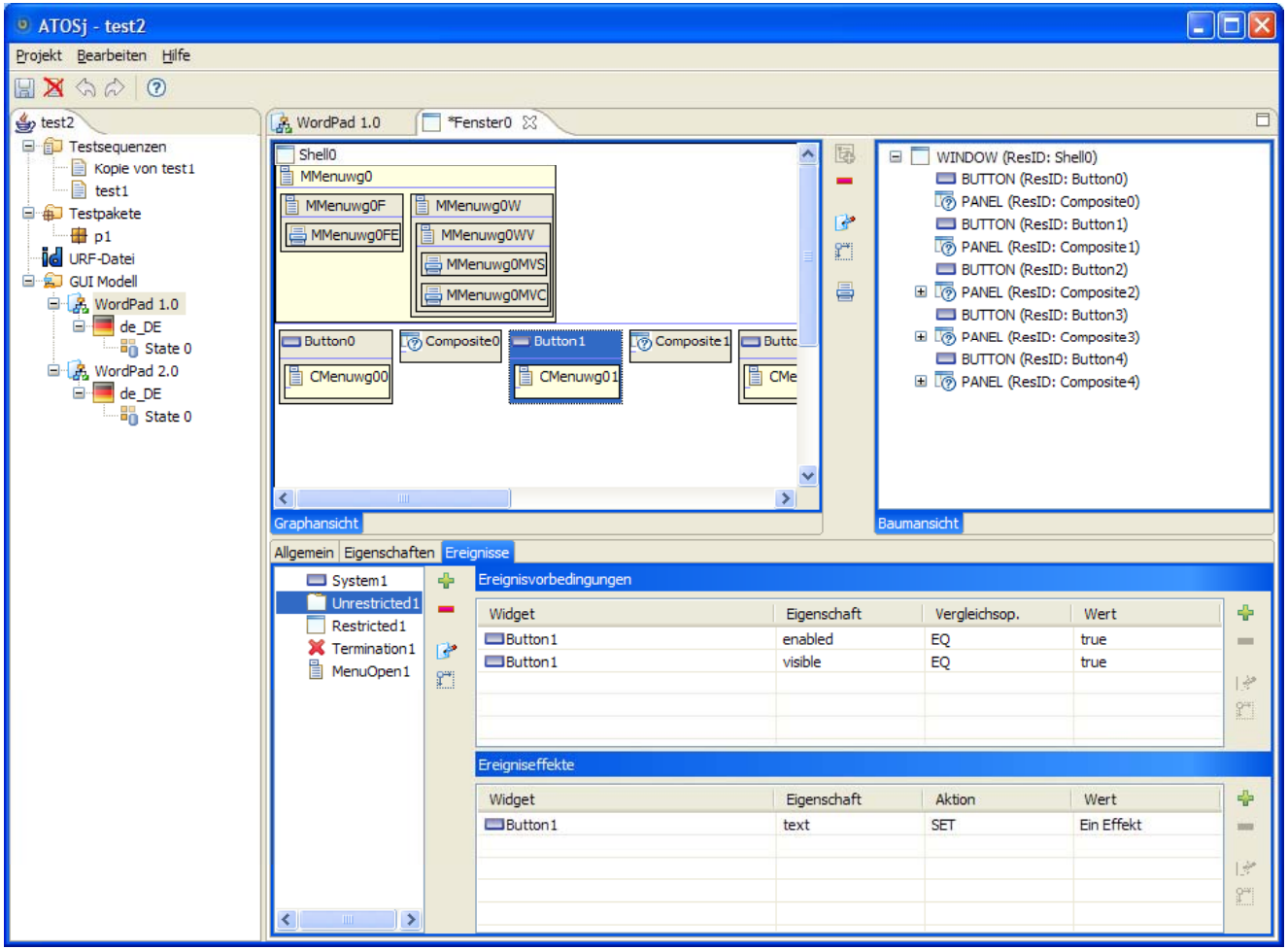
- Hierarchisch
- Ausgehend vom Fensterwidget
- Zusammen mit Aufrufgraph
  - Beschreibung der GUI-Struktur

Widgets

- Widget (auch Komponente genannt)
  - Grundlegendes grafisches Objekt (z.B. Button) einer GUI
  - 2 Arten
    - Grundlegend
    - Zusammengesetzte (Container / Composite / Panel)
  - Hauptelement des GUIModells
  - Bestimmt durch seine Eigenschaften (z.B. Hintergrundfarbe, angezeigter Text, etc.)

GUIModell - Widgets

Fensterkomponenteneditor



GUIModell - Widgets

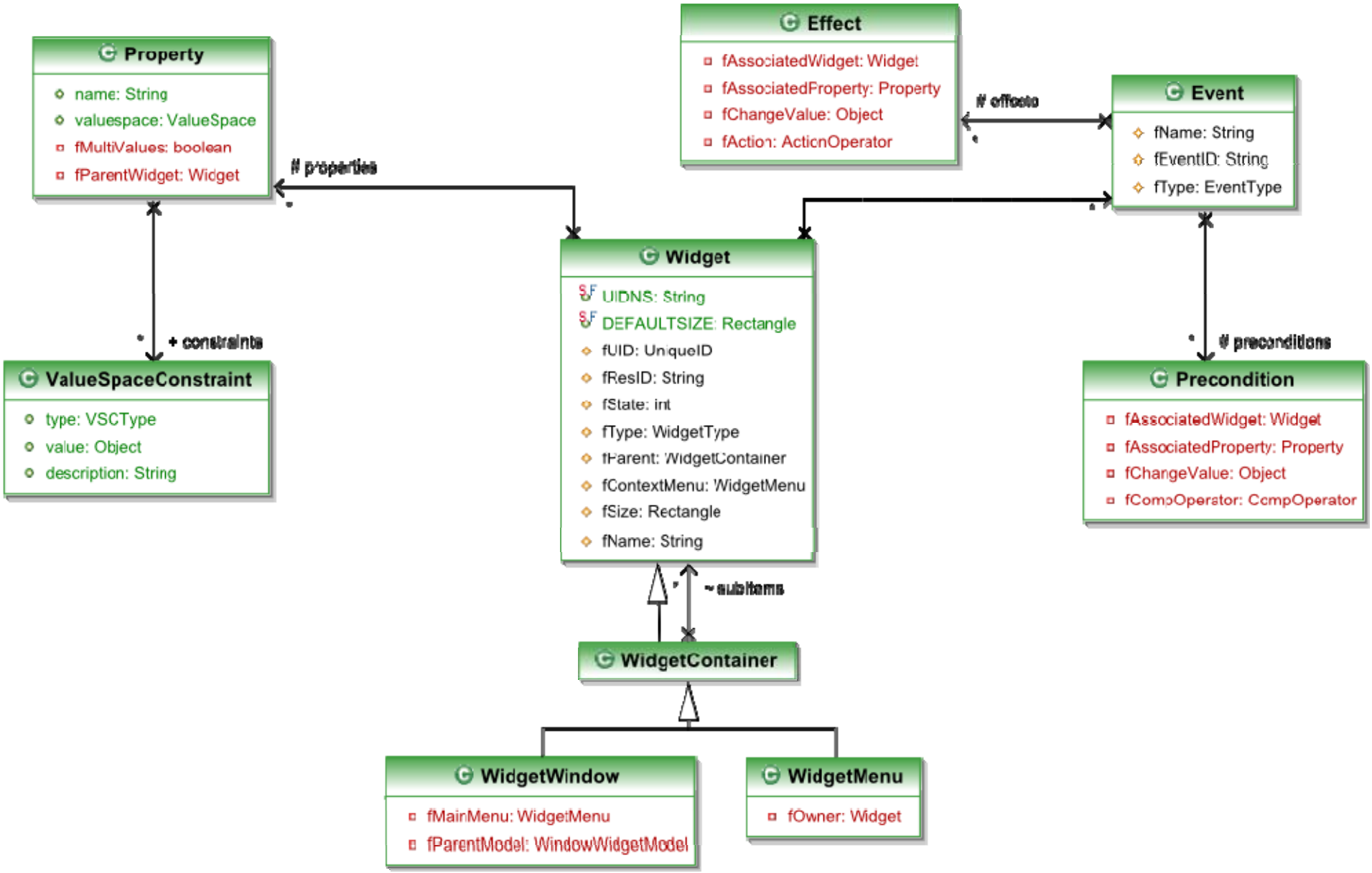
Daten I

- Widgets
  - Sind von bestimmten Typ (WidgetType)
  - Haben eindeutigen Bezeichner
  - Haben Eigenschaften (Properties)
  - Haben ein Vaterwidget (Hierarchie)
  - Können 1 Kontextmenü haben
  - Ereignisse werden zugeordnet
    - Die bei Interaktion mit Widget ausgelöst werden

GUIModell - Widgets

Daten II

- Containerwidgets
  - Zusätzlich Kinderwidgets
- Fensterwidgets
  - Zusätzlich Hauptmenü
  - Kein Vaterwidget


Property
name: String
valuespace: ValueSpace
fMultiValues: boolean
fParentWidget: Widget
Effect
fAssociatedWidget: Widget
fAssociatedProperty: Property
fChangeValue: Object
fAction: ActionOperator
effects
Event
fName: String
fEventID: String
fType: EventType
properties
Widget
UIDNS: String
DEFAULTSIZE: Rectangle
fUID: UniqueID
fResID: String
fState: int
fType: WidgetType
fParent: WidgetContainer
fContextMenu: WidgetMenu
fSize: Rectangle
fName: String
+ constraints
# preconditions
ValueSpaceConstraint
type: VSCType
value: Object
description: String
Precondition
fAssociatedWidget: Widget
fAssociatedProperty: Property
fChangeValue: Object
fCompOperator: CompOperator
~ subitems
WidgetContainer
WidgetWindow
fMainMenu: WidgetMenu
fParentModel: WindowWidgetModel
WidgetMenu
fOwner: Widget

GUIwalker 0.5

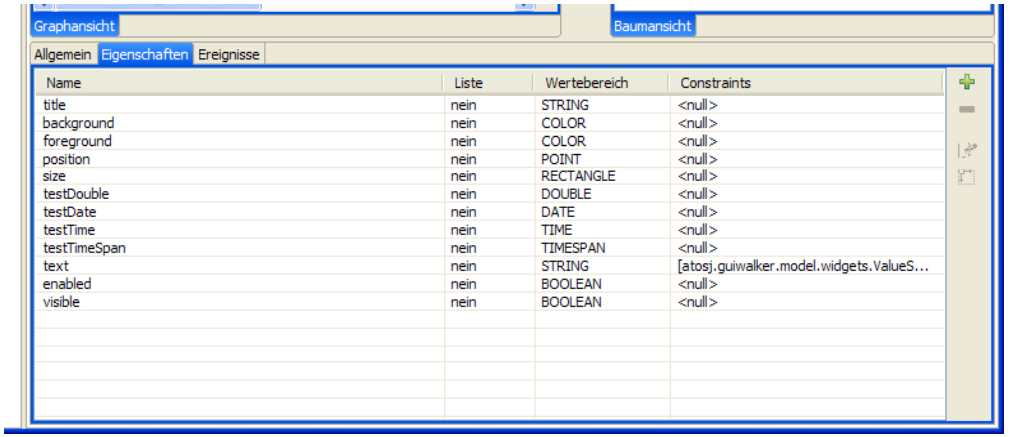
Editoren

- GUIModell
  - Aufrufgraph
  - Fenstermodell/Widgets
    - Eigenschaftsdefinition
    - Ereignisdefinition
- GUIState-Modell

GUIModell - Eigenschaften

Eigenschaftsdefinition

- Eigenschaften definiert durch
  - Widgetweit eindeutigen Namen
  - Wertebereich (int, string, …)
  - Ist Menge von Werten dieses Wertebereichs?
  - Beschränkungen



GUIModell - Eigenschaften

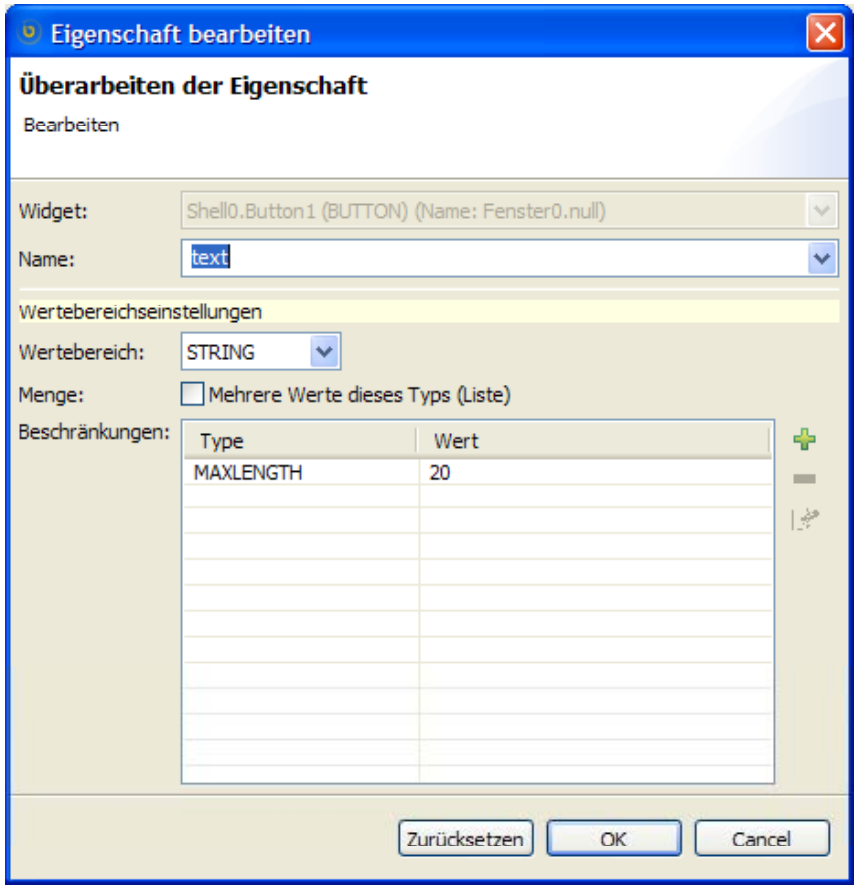
Eigenschaftswertebereich

- Eigenschaftswertebereich
  - Boolean
  - Ganzzahlig (long, int, short)
  - Fliesskomma (float, double)
  - Datumsformate (date, time, timespan)
  - Farbe (color)
  - Koordinaten (point, rectangle)
  - Text (string)
- Nutzen:
  - Je genauer der Wertebereich bestimmt ist, desto besser können Grenzfälle in generierten Testfällen getestet werden

GUIModell

Beschränkungen für Eigenschaft

- Speziell für jeden Wertebereich
- Mögliche Typen
  - MIN/MAXLENGTH
  - MIN/MAX (Wert)
  - ENUMERATION
  - DEFAULT
- Nutzen:
  - Je genauer die Beschränkungen desto bessere Grenzfalltestung



GUIwalker 0.5

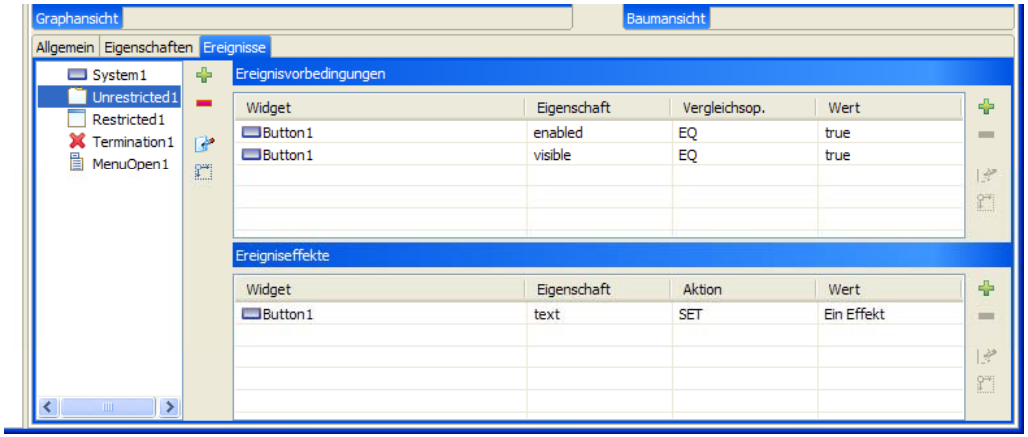
Modelle

- GUIModell
  - Aufrufgraph
  - Fenstermodell/Widgets
    - Eigenschaftsdefinition
    - Ereignisdefinition
- GUIState-Modell

GUIModell - Ereignisse

Ereignis - Bestandteile

- Ereignisse mit
  - Namen (nur für Nutzer interessant)
  - Vorbedingungen
  - Effekte
  - Typ des Ereignisses (Restricted Focus, Unrestricted Focus, Termination, Menu Open, System Interaction)
- Typ des Ereignisse ergibt sich aus Kontext der Effekte
  - Bei manueller Eingabe aber als Kontrolle zur Effektdefinition angebbar



GUIModell - Ereignisse

Vorbedingungen - Bestandteile

- Vorbedingungen
  - Widget auf das Bezug genommen wird
  - Eigenschaft des Widgets auf die Bezug genommen wird
  - Vergleichoperator (EQ, LEQ, LT, GEQ, GT, SET, NOTSET, ENABLED, VISIBLE, DEFINED)
  - Vergleichswert

GUIModell - Ereignisse

Effekte - Bestandteile

- Effekte
  - Widget auf das Bezug genommen wird
  - Eigenschaft des Widgets auf die Bezug genommen wird
  - Aktionsoperator (SET, UNSET, OPEN, CLOSE, OPENMENU)
  - Zu setzender Wert

GUIwalker 0.5

Modelle

- GUIModell
  - Aufrufgraph
  - Fenstermodell/Widgets
- GUIState-Modell

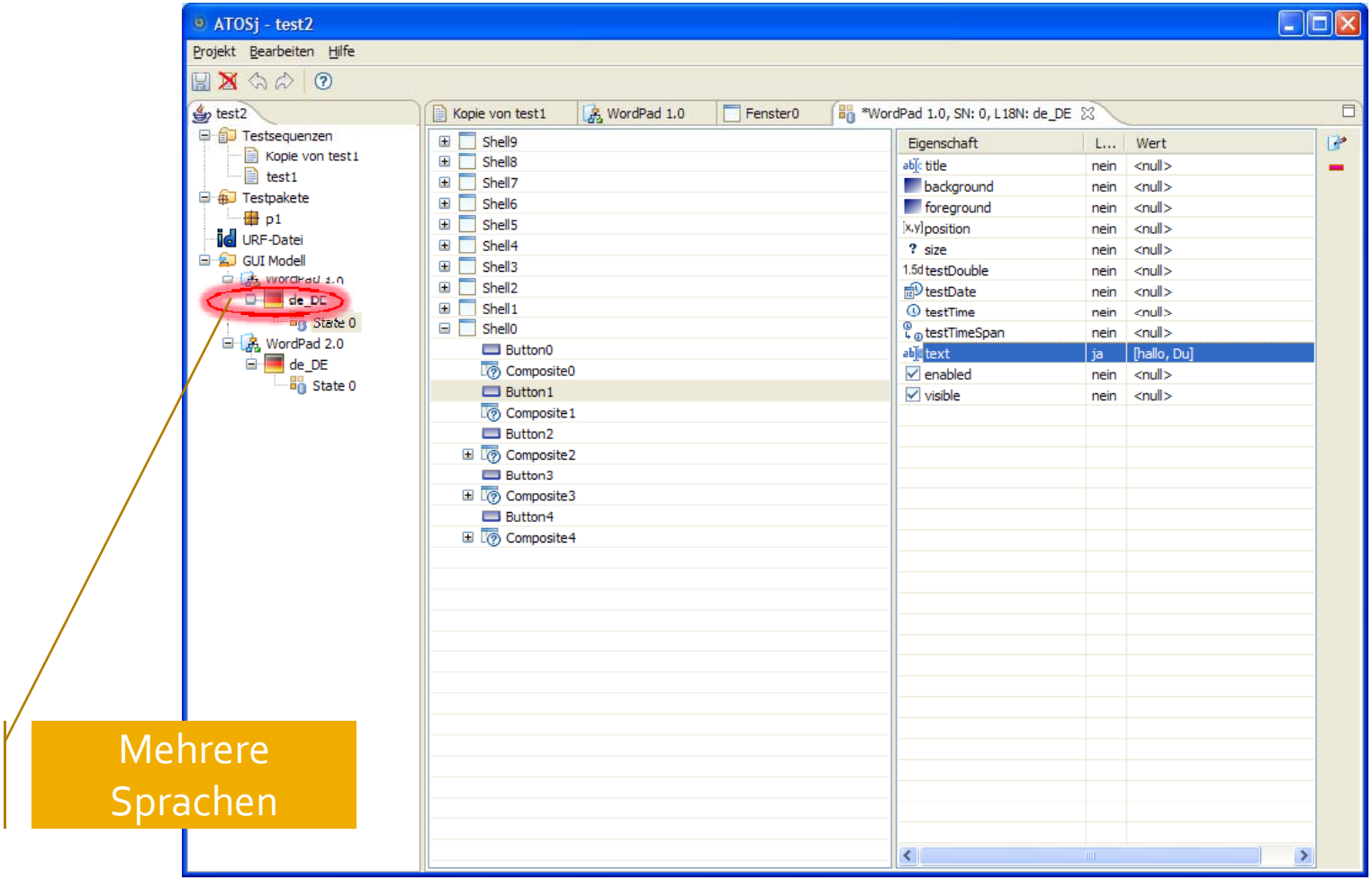
GUIState

Definition

- Menge von
  - Widget <-> WidgetState Zuordnung für das GUIModell
- Jedem relevanten Widget wird ein WidgetState zugeordnet
  - Nicht relevante Widget haben keine Zuordnung, d.h. sind nicht im GUIState
- Ein GUIState kann mehrere Fenster umfassen
- Pro Sprache ein separater Satz von GUIStates

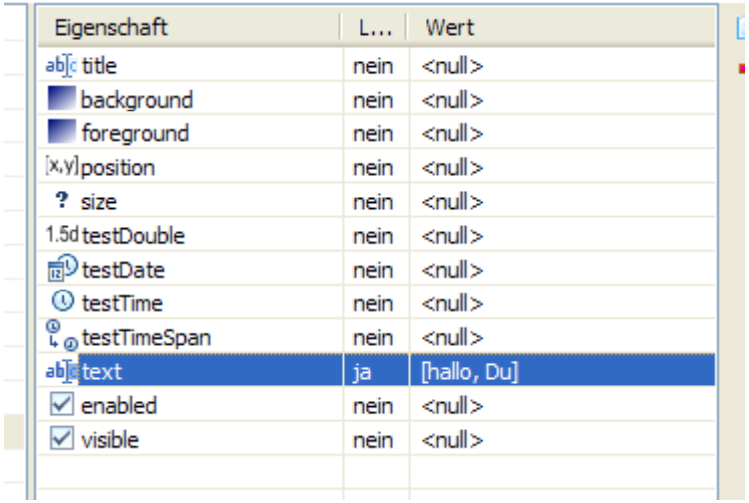
GUIState

GUIState-Editor



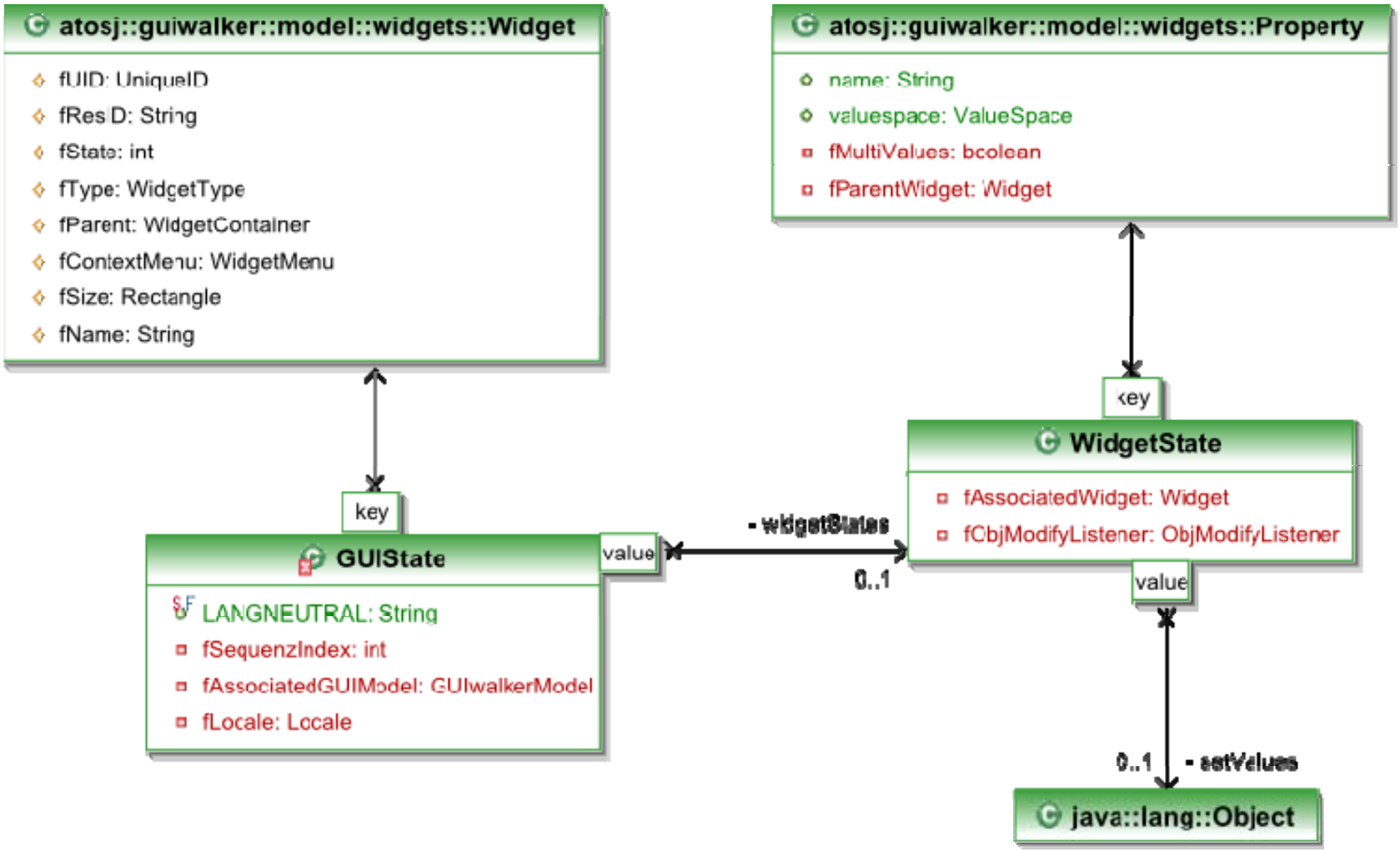
GUIState - WidgetState

- WidgetState:
  - Eigenschaft <-> Wert Zuordnung für ein Widget
  - Jeder relevanten Eigenschaft wird genau ein entsprechender Wert zugeordnet
  - Alle nicht relevanten Eigenschaften haben keine Zuordnung, d.h. sind nicht im WidgetState („<null>“)



Eigenschaft	L...	Wert
title	nein	<null>
background	nein	<null>
foreground	nein	<null>
position	nein	<null>
size	nein	<null>
testDouble	nein	<null>
testDate	nein	<null>
testTime	nein	<null>
testTimeSpan	nein	<null>
text	ja	[hallo, Du]
enabled	nein	<null>
visible	nein	<null>

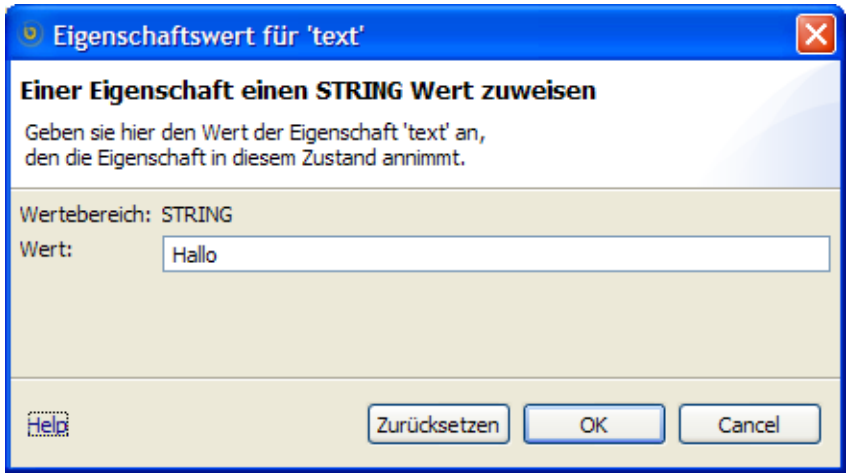
GUIState - UML

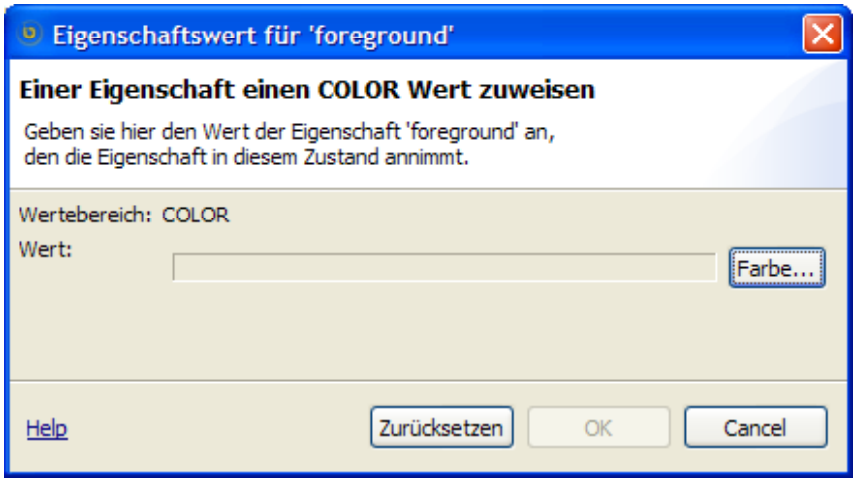


GUIState

Eigenschaftswert-Editor

- Verschiedene Editoren für unterschiedliche Wertebereiche
- Überprüfung der Beschränkungen bei Eingabe (noch nicht implementiert)





Editoren

- Wofür Editoren, wenn Modell automatisch erfasst werden soll?
  - GUI des Testobjekts fehlerhaft
  - Betrachten des erfassten Modells
  - Nicht alles korrekt erfassbar
    - Z.B. Verfeinerung des Wertebereich
  - Nicht nur Regressionstesten
    - Mit Editor Wunschmodell erzeugen
    - Daraus Testfälle generieren
    - Anpassung des Testobjekts an die Testfälle (== Wunschmodell)

GUIwalker 0.5

Überblick

- Einleitung/Wiederholung
- Modelle
- Algorithmen
- Probleme

GUIwalker 0.5

Algorithmen zur automatischen GUIModell-Erfassung Überblick

- Algorithmen
  - Erkennung GUI-Widgettyp
  - Erkennung Wertebereich Texteingabefeld
  - Erkennung Ereignisdefinition
  - Wiederherstellung eines Zustands nach Absturz des Testobjekts

Erkennung GUI-Widgettypen

- Anhand des Klassentyps (instanceof)
- Unterstütztung nur für Standardkomponenten

Erkennung GUI-Widgettyp

Widgettypen

- Texteingabefelder
- Comboboxen
- Listen
- Buttons
- Labels
- Menüeinträge
- Fenster
- Composites/Panels
- Gruppe
- Registerkarten
- Tabellen
- Bäume
- (Unter-)Menüs

GUIwalker 0.5

Algorithmen zur automatischen GUIModell-Erfassung Überblick

- Algorithmen
  - Erkennung GUI-Widgettyp
  - Erkennung Wertebereich Texteingabefeld
  - Erkennung Ereignisdefinition
  - Wiederherstellung eines Zustands nach Absturz des Testobjekts

Erkennung Wertebereich

- Texteingabefeld kann stehen für
  - Zeit/Datum
    - Spanne
    - Punkt
  - Zahlen (Gleitkomma/Ganzzahlen)
  - Ganz normalen Text

Erkennung Wertebereich

Algorithmus I

- Annahmen
  - Nicht zulässige Werte können nicht eingegeben werden
- Eingabe von
  - Grenzwerten für einen Wertebereich (z.B. Maximum für Integer-Werte)
  - Eingabe typischer Werte für Wertebereich

Erkennung Wertebereich

Algorithmus II

- Annahmen
  - Eingabefelder sind beschriftet
  - Beschriftung liefert Hinweis auf Wertebereich
- Zuordnung von Beschriftung zu Wertebereich durch festgelegtes/einstellbares Mapping

GUIwalker 0.5

Algorithmen zur automatischen GUIModell-Erfassung Überblick

- Algorithmen
  - Erkennung GUI-Kompententyp
  - Erkennung Wertebereich Texteingabefeld
  - Erkennung Ereignisdefinition
  - Wiederherstellung eines Zustands nach Absturz des Testobjekts

Erkennung Ereignisdefinition

Erkennung der Ereignistyp

- Erkennung der Ereignistypen
  - Menu Open
  - Restricted Focus
  - Unrestricted Focus
  - Termination
  - System Interaction

Erkennung Ereignisdefinition

Erkennung der Ereignistyp

- Restricted/Unrestricted Focus
  - Komponentenbeschriftungen (Buttons,Menüs), die mit „…“ enden, geben meist Hinweis auf folgenden Dialog
  - Stupides „try and error“ (Fenster öffnet oder nicht)
  - Unterscheidung Restricted/Unrestricted durch Auswertung des Fensterstiles des neu geöffneten Fensters
- Termination
  - Komponentenbeschriftungen ähnlich „Exit“, „Beenden“, „Schließen“, etc.
  - Standardbuttons wie das „x“ im Fensterrahmen
  - Position des Buttons in einer Toolbar (Annahme: alle Programme orientieren sich an einem Muster (Guidelines))
  - Try and error

Erkennung Ereignisdefinition

Erkennung der Ereignistyp

- Menu Open
  - Alle Menüs mit Untermenüs sind‘s automatisch
  - Alle Toplevelmenüs in Hauptmenüs (z.B. „Datei“)
  - Pfeil drücken bei Drop-Drown-Buttons (Toolbar)
- System Interaction
  - Alle übrigen Ereignisse

Erkennung Ereignisdefinition

Algorithmus - Effekte

1. Zustand der GUI wird gemerkt
2. Ereignis wird ausgelöst
3. Neuer Zustand wird gemerkt
4. Alter Zustand wird mit neuem verglichen
5. Werte, die sich geändert haben, werden als Effekte des ausgelösten Ereignisses vermerkt

Erkennung Ereignisdefinition

Algorithmus – Vorbedingungen I

1. In jedem Zustand der GUI wird vermerkt, welche Widgets
   1. Aktiv (sichtbar & enabled) bzw.
   2. Inaktiv sind
2. Für jedes Ereignis werden alle Zustände, in denen das Ereignis ausführbar ist, verglichen
   1. Alle Widgets in diesen Zuständen sind erstmal als Vorbedingung für dieses Ereignis gesetzt
   2. Widgets, bei denen das Ereignis ausführbar ist, wenn dieses Widget sowohl inaktiv als auch aktiv ist, werden von der Vorbedingung entfernt

Erkennung Ereignisdefinition

Algorithmus – Vorbedingungen II

3. Für alle noch vorhandenen Widgets werden deren Zustände über allen relevanten GUI-Zuständen verglichen
   1. Alle Eigenschaften, die über diese GUI-Zustände gleich bleiben, gelten als irrelevant und werden von Vorbedingung entfernt
   2. Alle Eigenschaften, die eindeutig für den Zustand aktiv oder inaktiv des Ereignisses sind (d.h. immer der gleiche Wert für (in)aktiv und beim anderen Zustand ist ein anderer Eigenschaftswert), werden mit dem jeweiligen Wert beim Zustand aktiv als Vorbedingung angesehen
   3. Alle anderen Eigenschaften und Eigenschaften mit mehreren eindeutigen Werten (Punkt 2) müssen in Zusammenhang mit anderen Widgets gebracht werden

GUIwalker 0.5

Algorithmen zur automatischen GUIModell-Erfassung Überblick

- Algorithmen
  - Erkennung GUI-Widgettyp
  - Erkennung Wertebereich Texteingabefeld
  - Erkennung Ereignisdefinition
  - Wiederherstellung eines Zustands nach Absturz des Testobjekts

Wiederherstellung des Testobjekts

Protokollierung

- Ständige Protokollierung
  - der ausgeführten Befehle
  - Der dauerhaften Veränderungen durch das Testobjekt
- Vor Ausführung des Befehls wird geloggt welcher Befehl ausgeführt wird
- Nach Ausführung wird der Log-Eintrag um die Veränderungen an der GUI ergänzt

Wiederherstellung des Testobjekts

Protokollierung - Probleme

- Persistente Veränderungen können überall stattfinden
  - Lokales Dateisystem (Festplatte, USB-Sticks, etc.)
  - Daten im Internet
  - Systemregistrierung
- Nicht alles kann im Protokoll erfasst werden
  - Speicherplatzproblem
  - Ermittlungsproblem
    - Remoteveränderungen
    - Veränderungen von außen

Wiederherstellung des Testobjekts

Protokollierung – Probleme – Lösungsansätze

- Persistente Veränderung
  - Remote
    - Remoteübertragungen mit protokollieren und bei Anforderung durch TO das ursprüngliche anstatt des Veränderten zurückgeben (zu aufwendig)
    - Remoteveränderungen nicht erfassen
  - Lokal
    - Lokale Dateisystemveränderungen durch Kopieren der Originale rückgängig machen
    - Veränderungen durch Dateisystemmonitoring aufspüren (zu aufwendig)
    - Zu sicherende Verzeichnisse durch Benutzer angeben lassen
    - Die Festplatte kopieren (zu umfangreich und langsam)
    - Lokale Registrierungsänderungen durch Mitprotokollieren und Zurücksetzen rückgängig machen

Wiederherstellung des Testobjekts

Algorithmus

- Nach Beendigung des Testobjekts wird der Ursprungszustand wiederhergestellt (Kopieren von veränderten Dateien, etc.)
- Die JVM mit dem Remoteprozess wird beendet und eine neue gestartet
  - Einflüsse durch veränderte Zustände der JVM werden vermieden
- Das Protokoll wird bis zu dem Befehl vor dem Befehl, der die Beendigung auslöste, abgespielt
- Jetzt kann ein anderer (neuer) Befehl ausgeführt werden

GUIwalker 0.5

Überblick

- Einleitung/Wiederholung
- Modelle
- Algorithmen
- Probleme

Probleme

- Alle Ereignisse ausführen führt zur Zustandsexplosion
- Vorbedingungsermittlung ist extrem Zeit- und Speicheraufwendig (alle Zustände müssten gemerkt werden und bei nur nicht-modalen Fenstern wird‘s schwierig (aber sehr selten))

Probleme

Reduzierung des Umfangs für Diplomarbeit

- Was müsste noch getan werden?
  - Automatisches Erfassen der GUI
    - Ermittlung der Eingabewerte
  - Schnittstelle für Angabe des Anfang-/Endzustand
  - Planung der Zwischenschritte/Testfallgenerierung
    - Erkennung der relevanten Sequenzen
    - Ermittlung von Eingabewerten und erwartetem Ergebnis
  - ~~Überprüfung der Überdeckung des Sourcecodes~~
  - ~~Reparatur von fehlerhaften Testsequenzen nach Programmmodifikation~~
    - ~~Ermittlung der Differenzen~~

Probleme

Reduzierung des Umfangs für Diplomarbeit

- Ermittlung der Eingabewerte durch
  - ~~Automatische Schätzung~~
  - Erweiterbare Auswahlliste von Werten für Nutzer
  - ~~Eingabe durch Nutzer~~
- Erkennung von relevanten Sequenzen
  - ~~Vollkommen automatisch~~
  - Angabe von Checkpoint-Zuständen zw. Anfangs- und Endzustand

Motivation

Testschritte

- Schritte fürs GUI-Testen
  1. Bestimmen was getestet wird
  2. Eingaben generieren
  3. Erwartete Ausgaben ermitteln
  4. Test durchführen und Ausgaben vergleichen
  5. Bestimmen, ob Tests ausreichende Abdeckung der Software erzielten
  6. Entdeckte Probleme in Software beheben
  7. Tests an modifizierter Software durchführen

ENDE

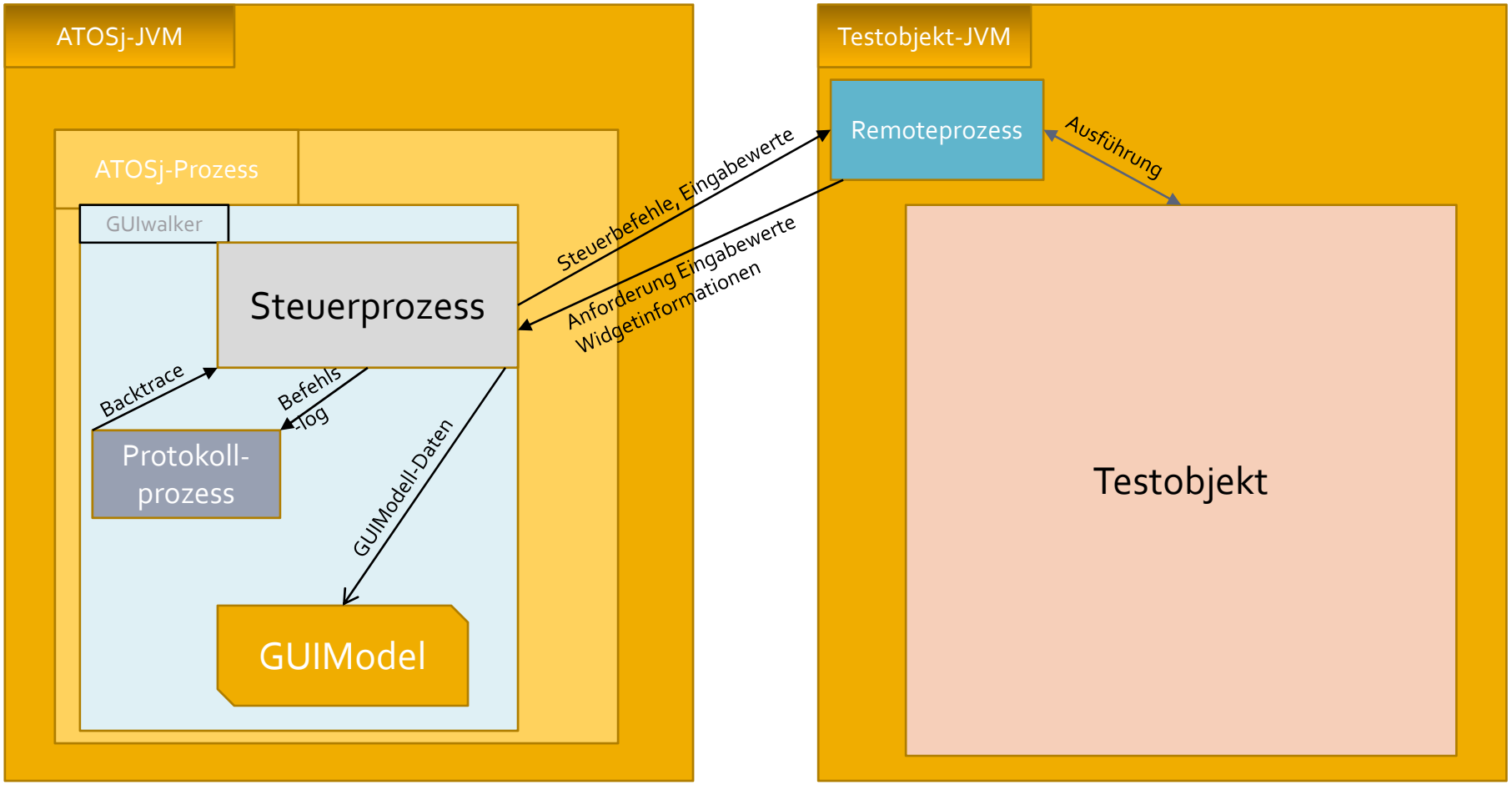
GUIwalker 0.5

Algorithmen zur automatischen GUIModell-Erfassung
Überblick

- Vorgehen
- Algorithmen
  - Erkennung GUI-Kompententyp
  - Erkennung Wertebereich Texteingabefeld
  - Erkennung Ereignisdefinition
  - Wiederherstellung eines Zustands nach Absturz des Testobjekts

Vorgehen Erfassung

Übersicht Komponenten



Vorgehen Erfassung

Allgemeines Vorgehen für den Erfassungsprozess - Starten

- Steuerdialog starten
  - Benutzereinstellungen erfassen
  - Nutzer löst Erfassungsprozess aus
- Lokalen Steuer-Prozess initialisieren
  - Protokollierung (Backtrace) starten
- Remote-Analyzer-Modul (RM) in neuer JVM-Instanz starten+initialisieren
- Steuer-Prozess gibt RM Befehl zum Initialisieren des Testobjekts
  - Reproduzierbarer Ausgangszustand für Testobjekt wird hergestellt
    - Dateien neu kopieren/erstellen
  - In Sandbox ausführen (Aktivitäten kontrollieren/protokollieren)

Vorgehen Erfassung

Steuer-Prozess (lokal)

- Steuerprozess
  - Sendet Kommandos
  - Protokolliert Abfolge Kommandos
  - Ermittelt Ereignisdefinition
  - Liefert Eingabewerte auf Anfrage vom Remoteprozess

Vorgehen Erfassung

Remote-Prozess (Remote, im Testobjekt-Prozess)

- Remoteprozess
  - Führt Kommandos aus
  - Identifiziert Objekte und stellt Informationen dazu bereit
  - Identifiziert den Eingabebereich und dessen Beschränkungen

Begriffe

- Testobjekt
  - Die Software, deren GUI erfasst/getestet werden soll
- Modaler Aufruf eines Fensters
  - Modales Fenster:
    - Das Fenster besitzt den alleinigen Eingabefokus für eine Anwendung
    - Eingaben in allen *vorher* aufgerufenen Fenster **können nicht** mehr gemacht werden
    - In *nachfolgend* gerufenen Fenstern können Eingaben gemacht werden
  - Nicht-Modales Fenster
    - Eingaben in *vorher und nachher* aufgerufenen Fenster **können** gemacht werden

Begriffe

Widgets

- GUI besteht aus grafischen Objekten (Widgets)
  - Jedes hat bestimmte Menge an Eigenschaften
  - Einige (z.B. Fenster, Panels, Registerkarten, Bäume,…) haben Unterobjekte und werden als Container bezeichnet

Begriffe

Eigenschaft/Ereignis

- Eigenschaft
  - Eigenschaften haben
    - Eindeutige Beschreibung (Namen)
    - diskrete Werte (keine Videos)
  - angezeigter Text, Größe, Position, Textfarbe, etc.
- Ereignis
  - Braucht bestimmte Voraussetzungen
  - Ändert den Zustand der Software