GUIwalker

Automatisierte Erfassung von grafischen Benutzungsschnittstellen und automatisierter Ableitung von Testfällen für grafische Benutzungsschnittstellen

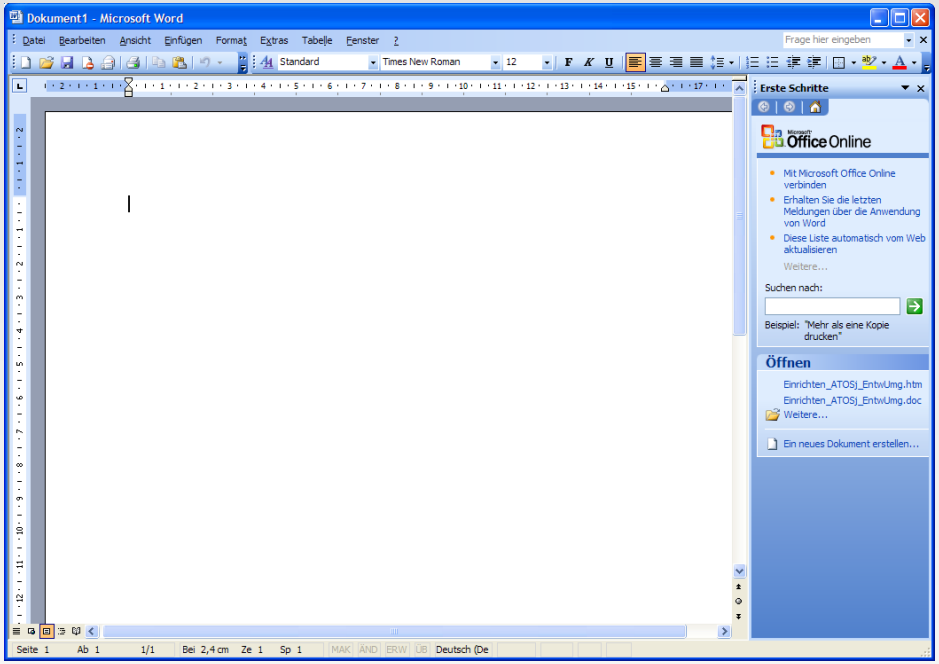
Teil 1 – Version 0.5

Christoph Graupner

Überblick

- Begriffsklärung
- Motivation
- Automatische Testfallgenerierung
- GUIwalker
- Ausblick

Begriffsklärung
Was ist eine GUI?



GUI:

- A computer environment or program that displays, or facilitates the display of, on-screen options, usually in the form of icons (pictorial symbols) or menus (lists of alphanumeric characters) by means of which users may enter commands.

ATIS Telecom Glossary 2000

Begriffsklärung
GUI

- 2 Arten von GUIs
  - Standard Windows/Unix/... Applikationen
  - Webbrowserbasierte Applikationen
- Konzentration auf Applikationen, die
  - hierarchisches, grafisches Front-end zu einem Softwaresystem sind, welches benutzer- und systemgenerierte Ereignisse akzeptiert und determinierte grafischen Ausgaben erzeugt.

Begriffsklärung
Widgets

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

Begriffsklärung
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

Überblick

- Begriffsklärung
- Motivation
- Automatische Testfallgenerierung
- Architektur GUIwalker
- GUI Modell
- Ausblick

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

Motivation
Testdurchführung bisher

- Schritte meistens ausgeführt
  - “von Hand”
    - Testsequenzen werden durch Menschen einzeln an der Software ausgeführt
  - Skripten
    - Skript, das Benutzeraktionen beschreiben und Ausgabewerte definiert mit einem Tool ausführen/vergleichen
  - Capture/Replay
    - Aufzeichnen/Wiedergabe von Benutzeraktion und Softwarereaktionen durch ein Tool

Motivation
Testdurchführung demnächst

- Wunsch:
  - Testsequenzen automatisch aus der Spezifikation ableiten
- Eine Möglichkeit
  - Modell der GUI-Struktur erstellen
  - Aus dem Modell Aktionen und Reaktionen ableiten
  - Automatisch Tests durchführen und Überdeckung analysieren

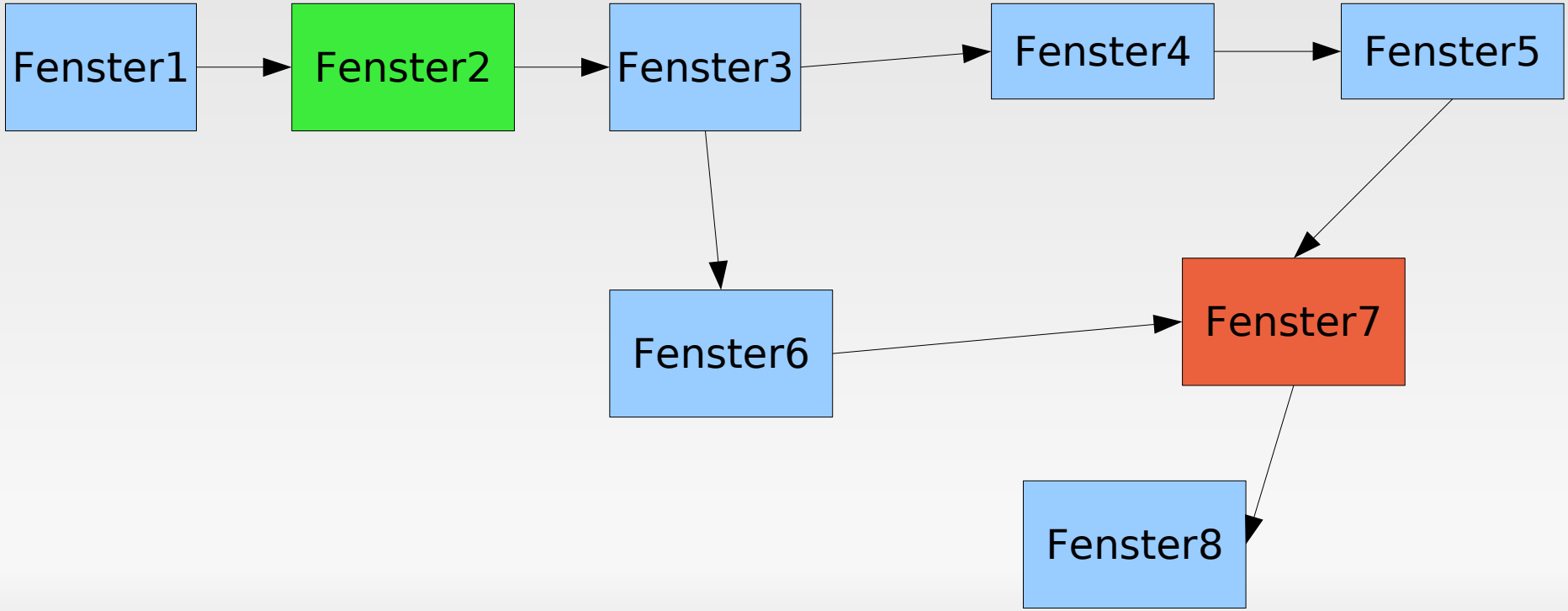
Überblick

- Begriffsklärung
- Motivation
- Automatische Testfallgenerierung
- GUIwalker
- Ausblick

- Atif M. Memon: “A Comprehensive Framework For Testing Graphical User Interfaces”, Dissertation, 2001, Universität von Pittsburgh
  - GUI-Beschreibungsmodell
  - Model eines Testframework, das das GUI-Modell nutzt um
    - Testsequenzen abzuleiten und
    - Tests und Überdeckungsanalyse durchführt
    - Testfälle repariert

Testsequenzen ableiten

- Idee:
  - Anfangs und Endzustand angeben und Pfad durch KI-Planung ermitteln.
- Pfad ermitteln. Wie?
  - Graph -> alle Pfade zwischen zwei Punkten bestimmen
- Welcher Graph?
  - 2 sich ergänzende Graphtypen: Event Flow Graph & Integration Tree


Fenster1
Fenster2
Fenster3
Fenster4
Fenster5
Fenster6
Fenster7
Fenster8

Graphen zur Planung

- Event Flow Graph (auch GUI control flow graph)
  - Gibt an, welche Ereignisse auf ein anderes folgen können
- Integration Tree (auch GUI call graph)
  - Gibt die Komponentenbeziehungen an

Event Flow Graph
Beispiel Wordpad - Main

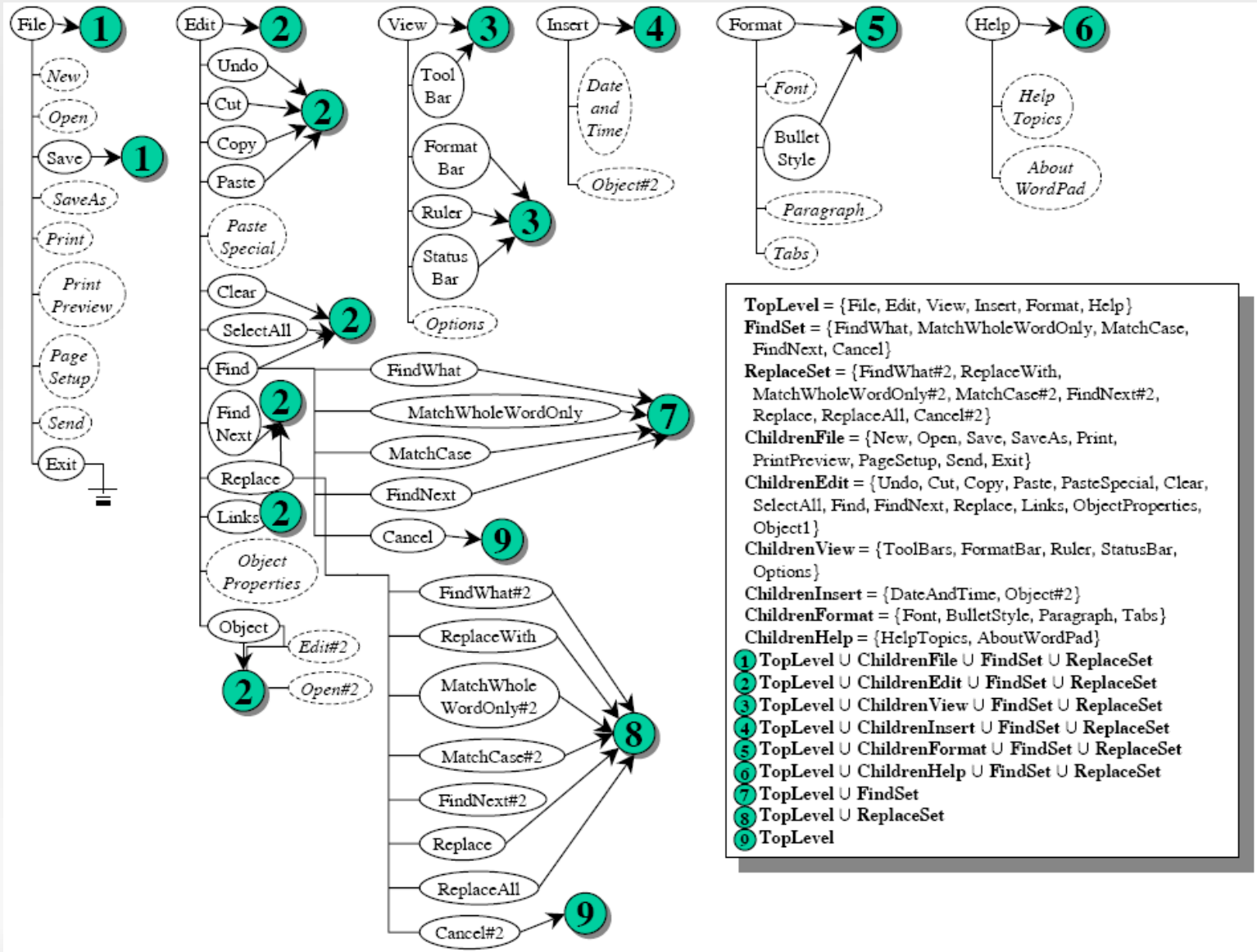
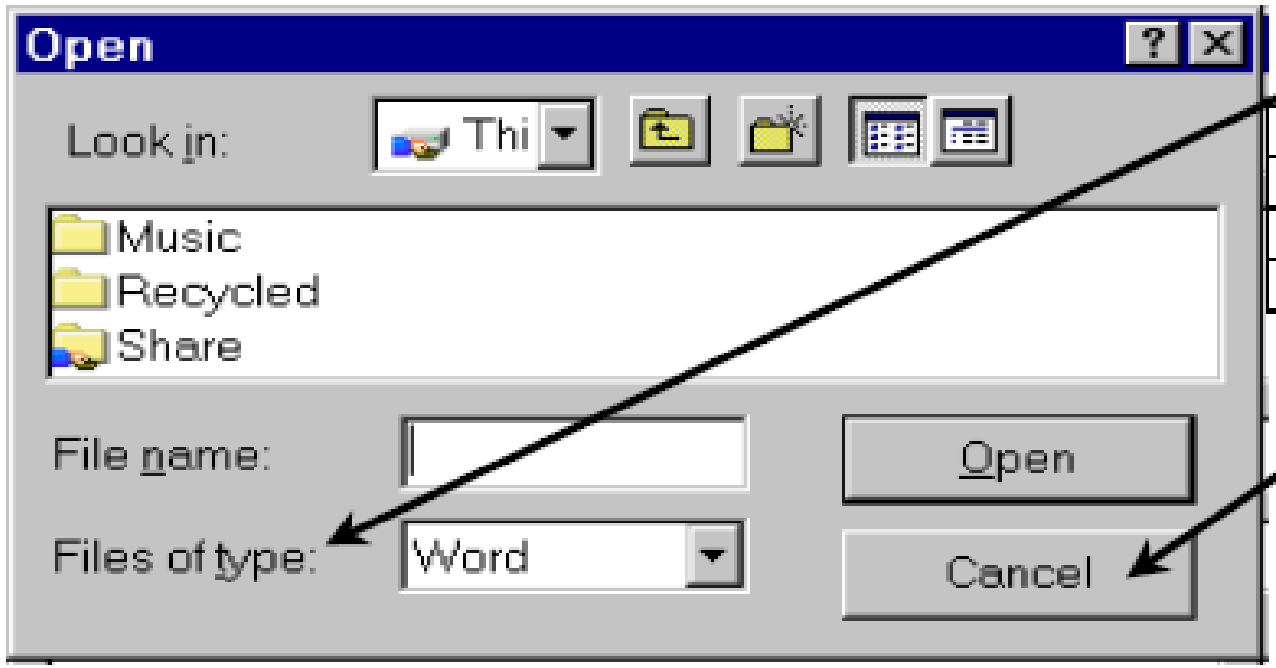


Figure 3.10: Event-flow Graph for the `Main` Component of MS WordPad.

Zustand eines GUI
Was muss repräsentiert werden?

- Zustand einer GUI
  - Enthaltene Objekte
  - Eigenschaften der Objekte (angezeigter Text, Button aktiv, Hintergrundfarbe, etc.)
- Zustandsübergänge (Ereignisse)
  - Funktion, beschrieben durch
    - Vorbedingungen
      - Menge von benötigten Eigenschaften und deren Wert
    - Auswirkungen (Effekte)
      - Eigenschaften, die verändert werden und der neue WertVorbedingungen

Eigenschaften
Beispiel WordPad - OpenDlg



Label1

Align	alNone
Caption	Files of type:
Color	clBtnFace
Font	(tFont)

Button1

Caption	Cancel
Enabled	TRUE
Visible	TRUE
Height	65

(a)

State = {(Label1, Align, alNone), (Label1, Caption, "Files of type:"), (Label1, Color, clBtnFace), (Label1, Font, (tfont)), (Form1, WState, wsNormal), (Form1, Width, 1088), (Form1, Scroll, TRUE), (Button1, Caption, Cancel), (Button1, Enabled, TRUE), (Button1, Visible, TRUE), (Button1, Height, 65), ...}

(b)

Ereignis
Beispiel: Hintergrundfarbe

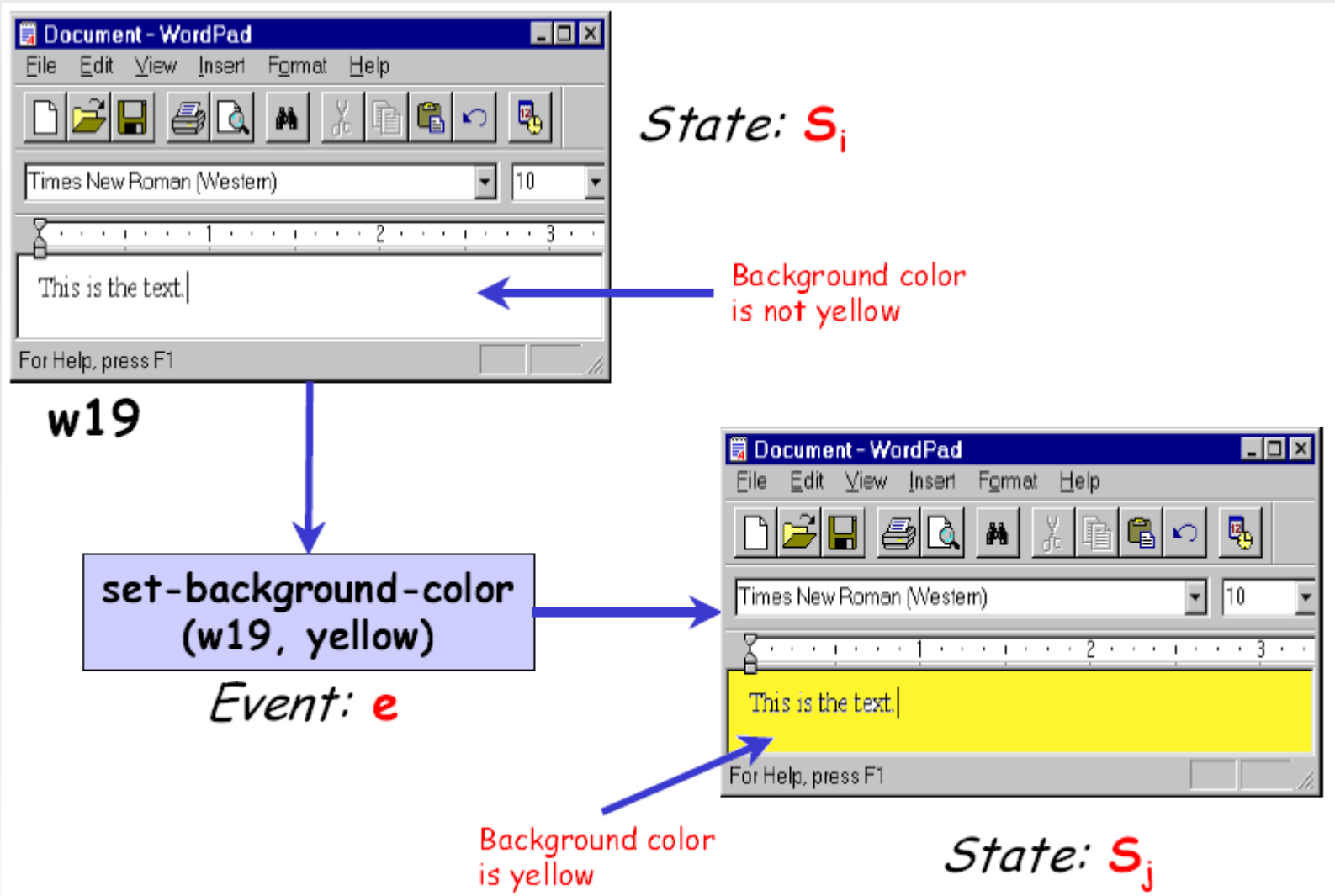
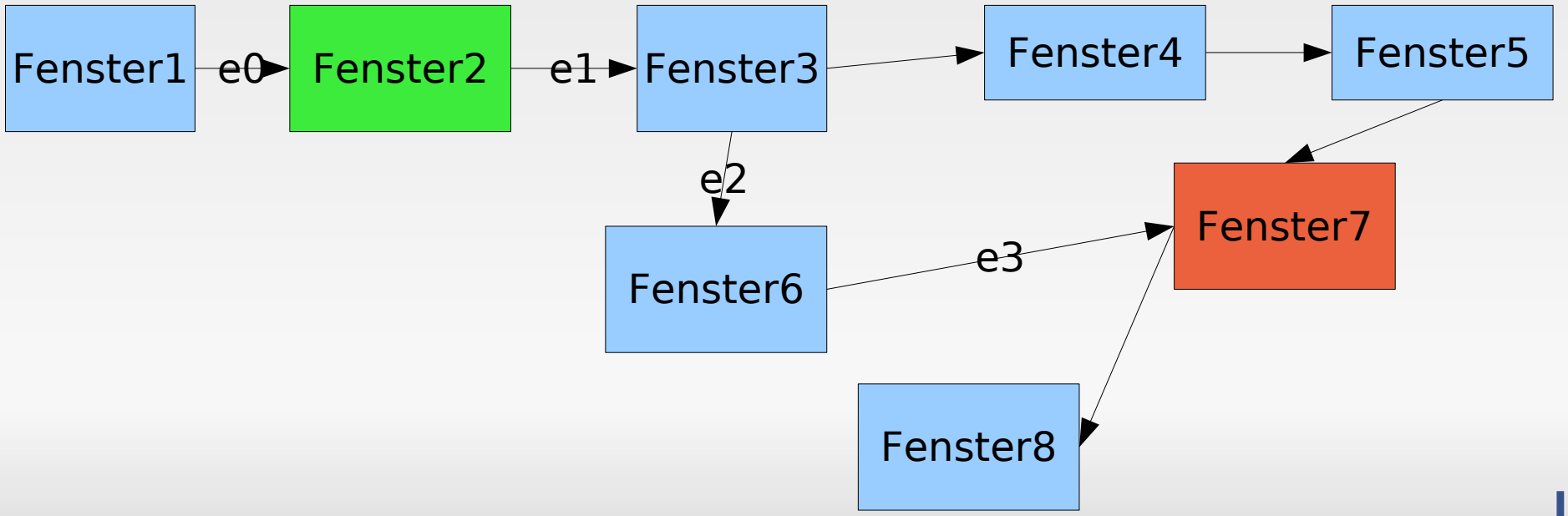


Figure 3.4: An Event Changes the State of the GUI.

Anfangs- zu Endzustand

- Vom Anfangs- zum Endzustand
  - mehrfache Anwendung von Ereignissen auf einen Folgezustand vom Anfangszustand
  - Bis Endzustand erreicht ist

- e1(F2) -> F3
- e2(F3) -> F6
- e3(F6) -> F7



Event Flow Graph
Komplexitätsreduzierung

- Ein Event Flow-Graph für gesamte Anwendung
  - zu viele Ereignisse in einem Graph
- daher
  - Komplexitätsreduzierung durch Komponentenbildung
  - Für jede Komponente eigener Event Flow Graph
  - Komponentenbeziehung durch Integration Tree dargestellt

Komponente
Definition

- Komponenten sind
  - Ein modales Fenster (Dialog) und
  - Seine gesamten nicht-modalen Unterfenster
- Warum nicht Fenster?
  - Modale Fenster + alle nicht-modalen Unterfenster bilden abgeschlossenes System gegenüber Ereignisabfolge:
  - (fast) jedes Folgeereignisse liegt in der selben Komponente

Integration Tree
Beispiel Wordpad (Ein Teil)

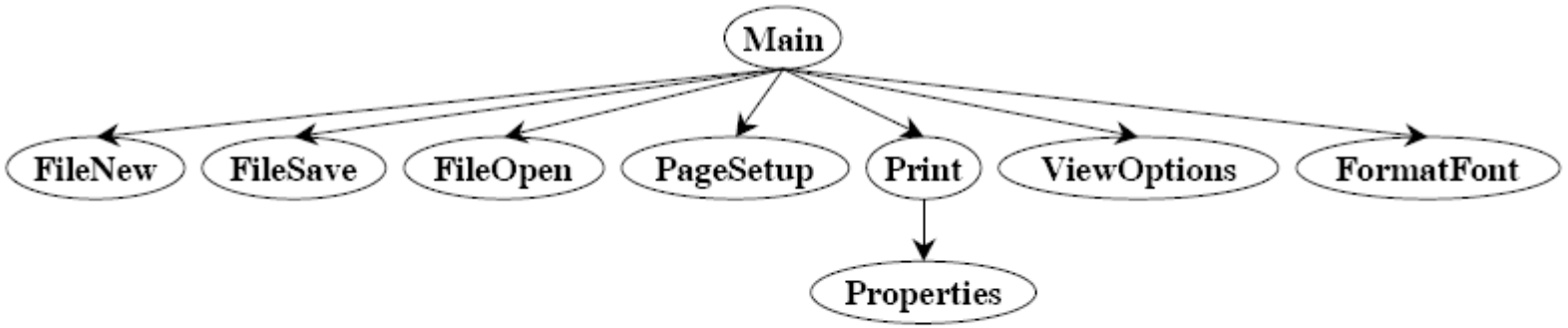


Figure 3.12: An Integration Tree for a Part of MS WordPad.

Event Flow Graph
Beispiel Wordpad - Main

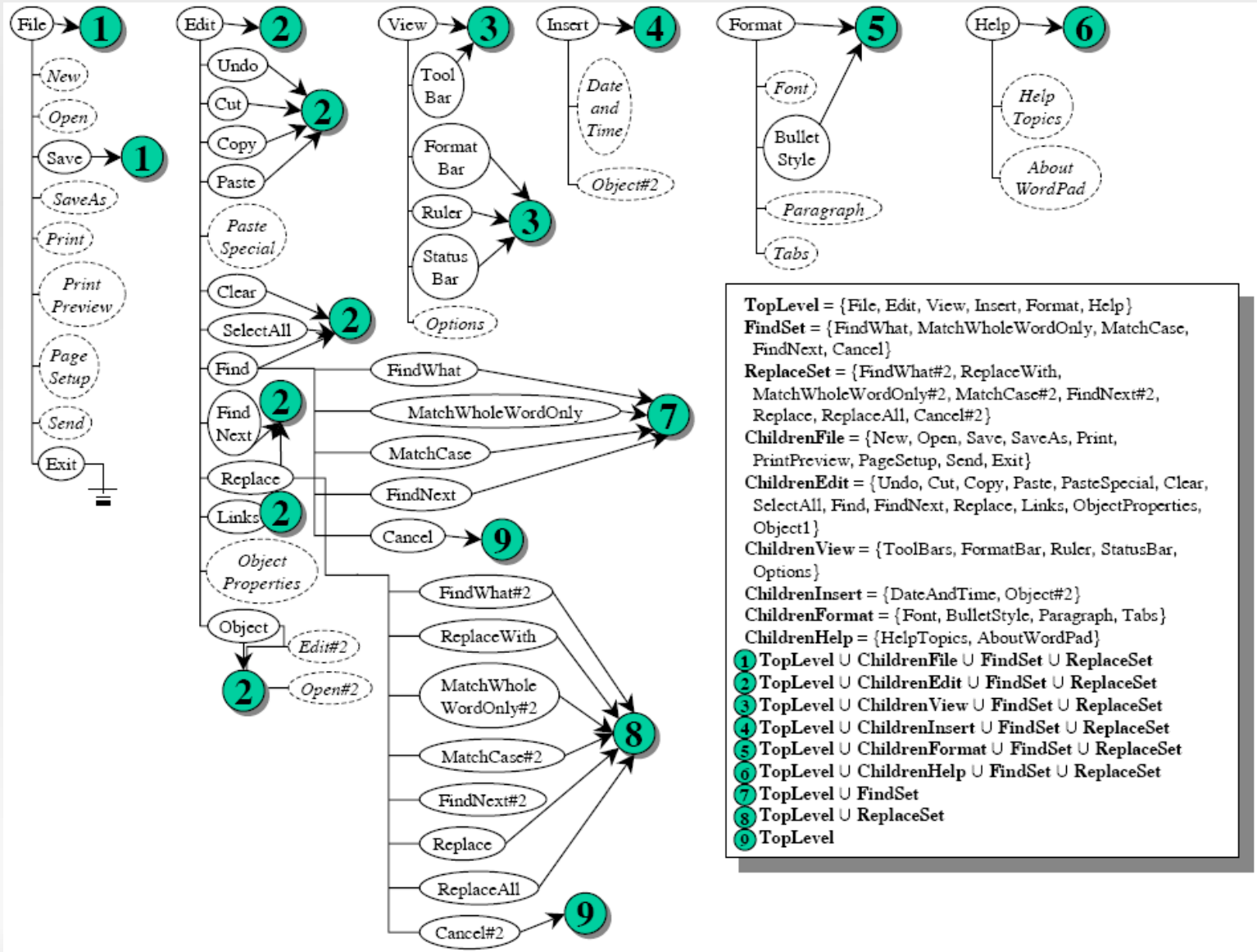


Figure 3.10: Event-flow Graph for the `Main` Component of MS WordPad.

Planungsoperatoren

- Zur effizienteren Planung, wird nicht direkt auf unterster Ebene (EFG) geplant
- Mehrere Ereignisse zusammengefasst zu Planungsoperatoren
- Bei konkreter Testsequenz werden Operatoren wieder zerlegt
- Sinnvolle Zusammenfassung durch Klassifizierung von Ereignisse

Vorteil Highlevel-Planung

- Highlevel-Plan muss nicht immer geändert werden
  - Änderungen einzelner Ereignisse im Operator verborgen
  - Nur betroffene Low-Level-Plänen müssen neuerstellt werden
- Low-Level-Pläne schneller erstellbar, da Eckdaten vorgegeben und kleine Ereignismengen
- Spart Zeit beim erneuten Planen

Ereignisse
Klassifizierung

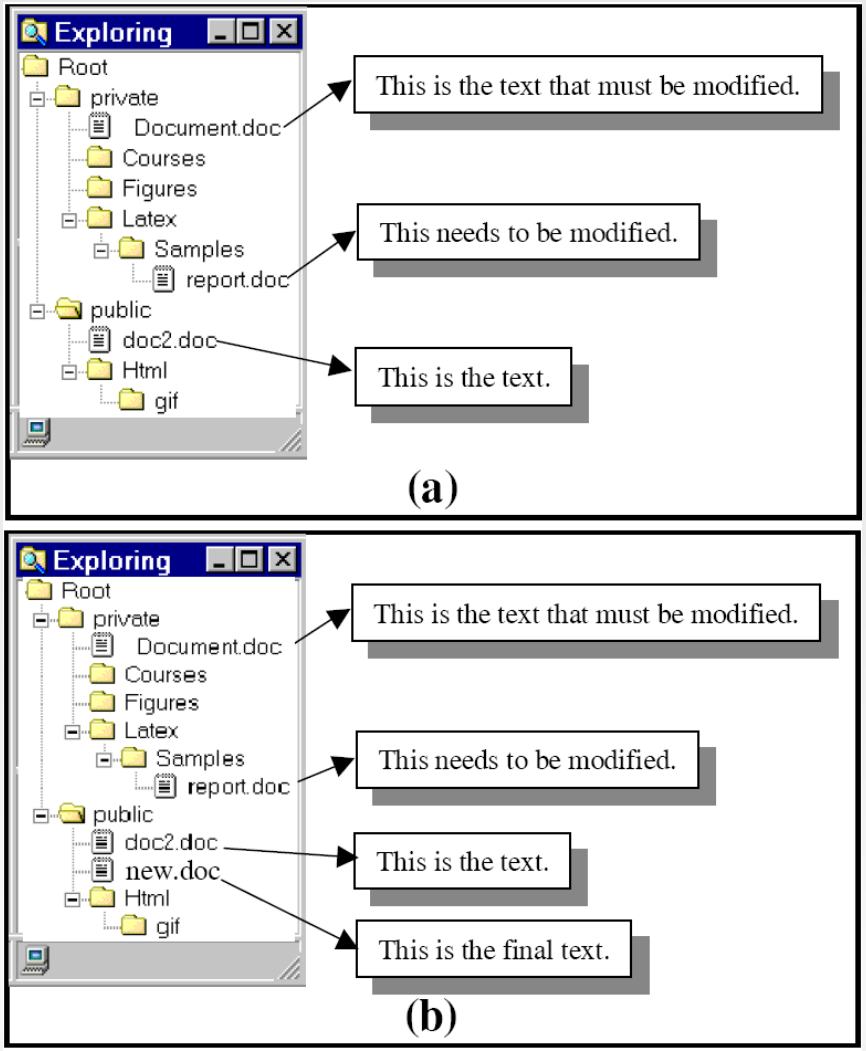
- Restricted-focus
  - Öffnet modales Fenster (Dialog)
- Unrestricted-focus
  - Öffnet nicht-modales Fenster
- Termination
  - Schließt Fenster
- Menu-open
- System-interaction

Planungsoperatoren

- System-interaction Operator
  - Sequenz von
    - 0 oder mehr menu-open und unrestricted-focus
    - 1 system-interaction Ereignis
  - z.B. EDIT_CUT = <Edit, Cut>
- Komponenten Operator
  - Menge aller Ereignisse, die in einer Komponente sind, die durch ein restricted-focus Ereignis gerufen wird
  - z.B. File_Open = <File, Open>

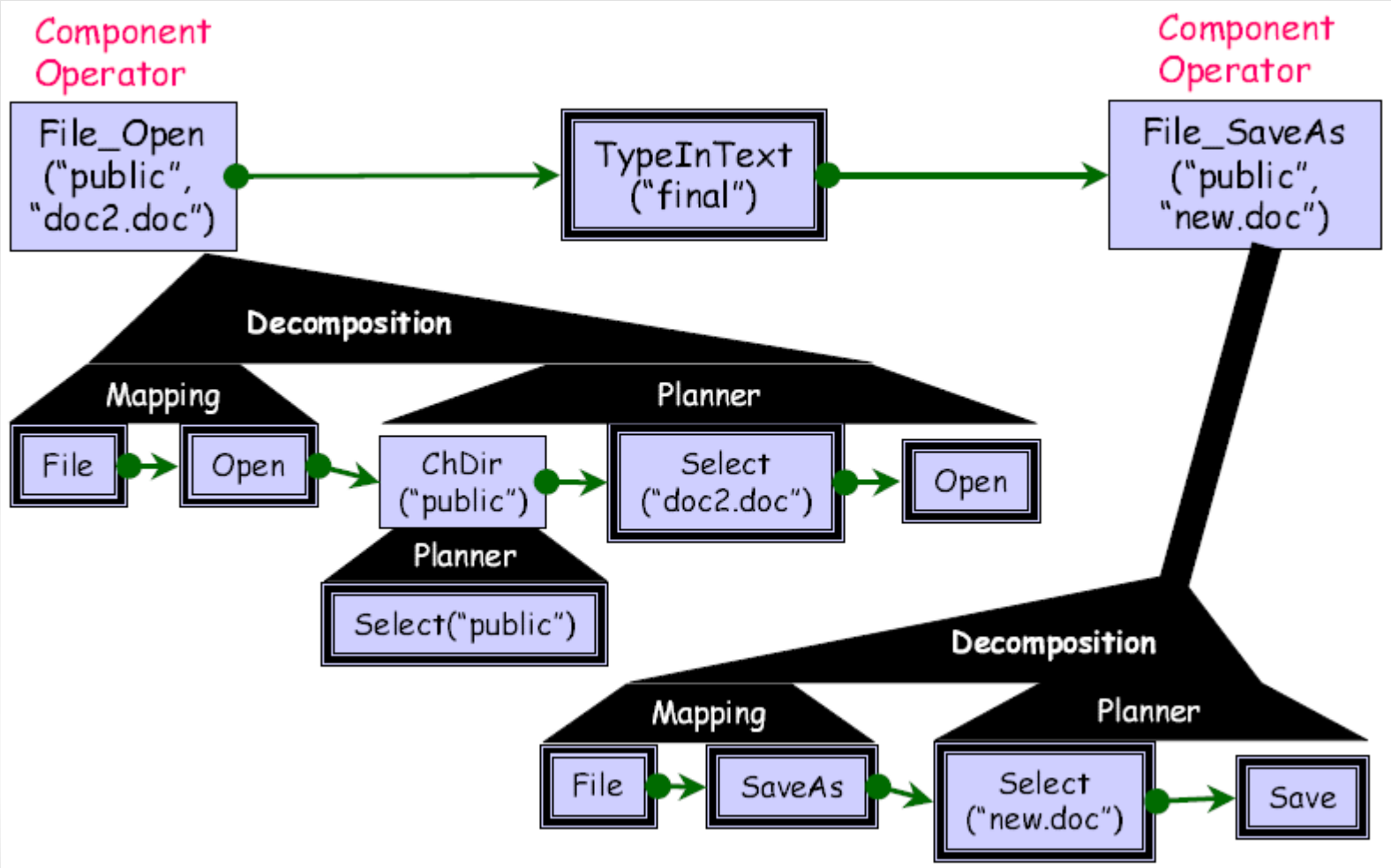
Plangenerierung
Anfangs/Endzustand festlegen

- Planer (Mensch) legt Anfangs- und Endzustand fest



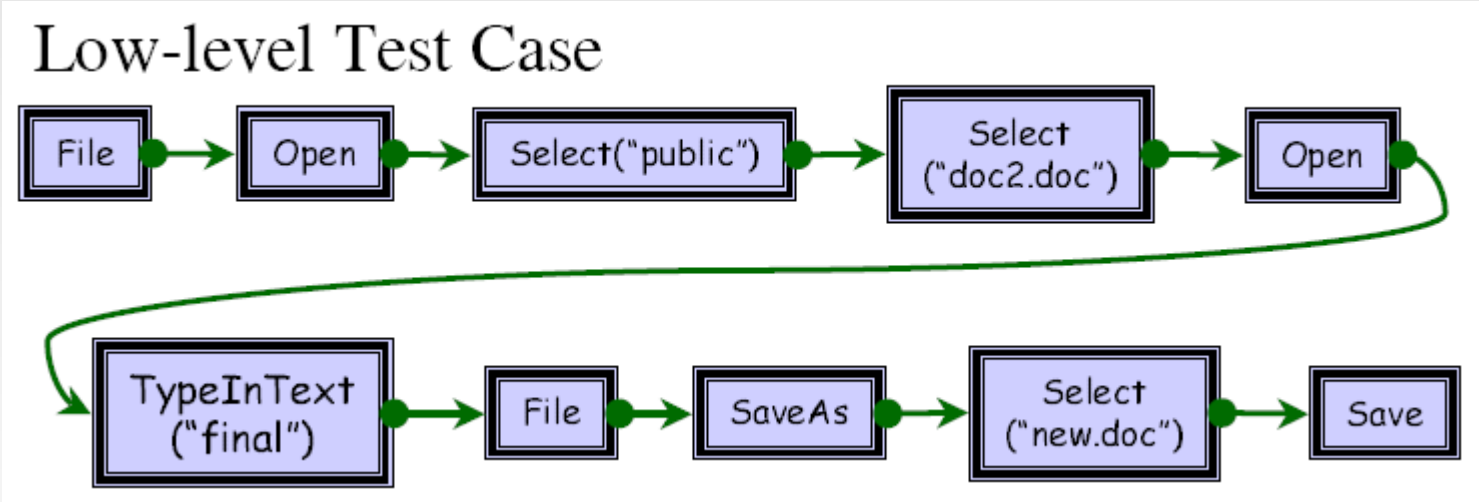
Plangenerierung
Highlevel-Plan

- Testgenerator erstellt Highlevel-Plan



Plangenerierung
Lowlevel-Plan

- Testgenerator zerlegt Highlevel-Plan zu Testsequenzen



Überblick

- Begriffsklärung
- Motivation
- Automatische Testfallgenerierung
- GUIwalker
- Ausblick

GUIwalker
Worum geht es?

- Programm, welches ATOSj um Funktionen der automatischen Erstellung eines GUI-Modells und automatischen Ableitung von Testfällen erweitert.
- Version 0.5 (Studienarbeit)
  - Erfassung des GUI-Modells
- Version 1.0 (Diplomarbeit)
  - Automatische Ableitung und Reparatur von Testfällen

Zielbestimmung
Musskriterien

- Musskriterien
  - Erfassung der GUI-Struktur der Software unter Test
  - Bearbeiten der erfassten GUI-Struktur
  - Generierung von Testeingabewerten
  - Ermittlung der erwarteten Testeingabereaktionen
  - Ableiten von relevanten Testfällen
  - Identifizierung und Reparatur oder Löschung von inkorrekten Testskripten

Zielbestimmung
Wunschkriterien

- Wunschkriterien
  - Anzeige der GUI-Struktur in einem Graphen
  - Messung der Abdeckung der Softwarefunktionen der SUT durch die generierten Testfälle

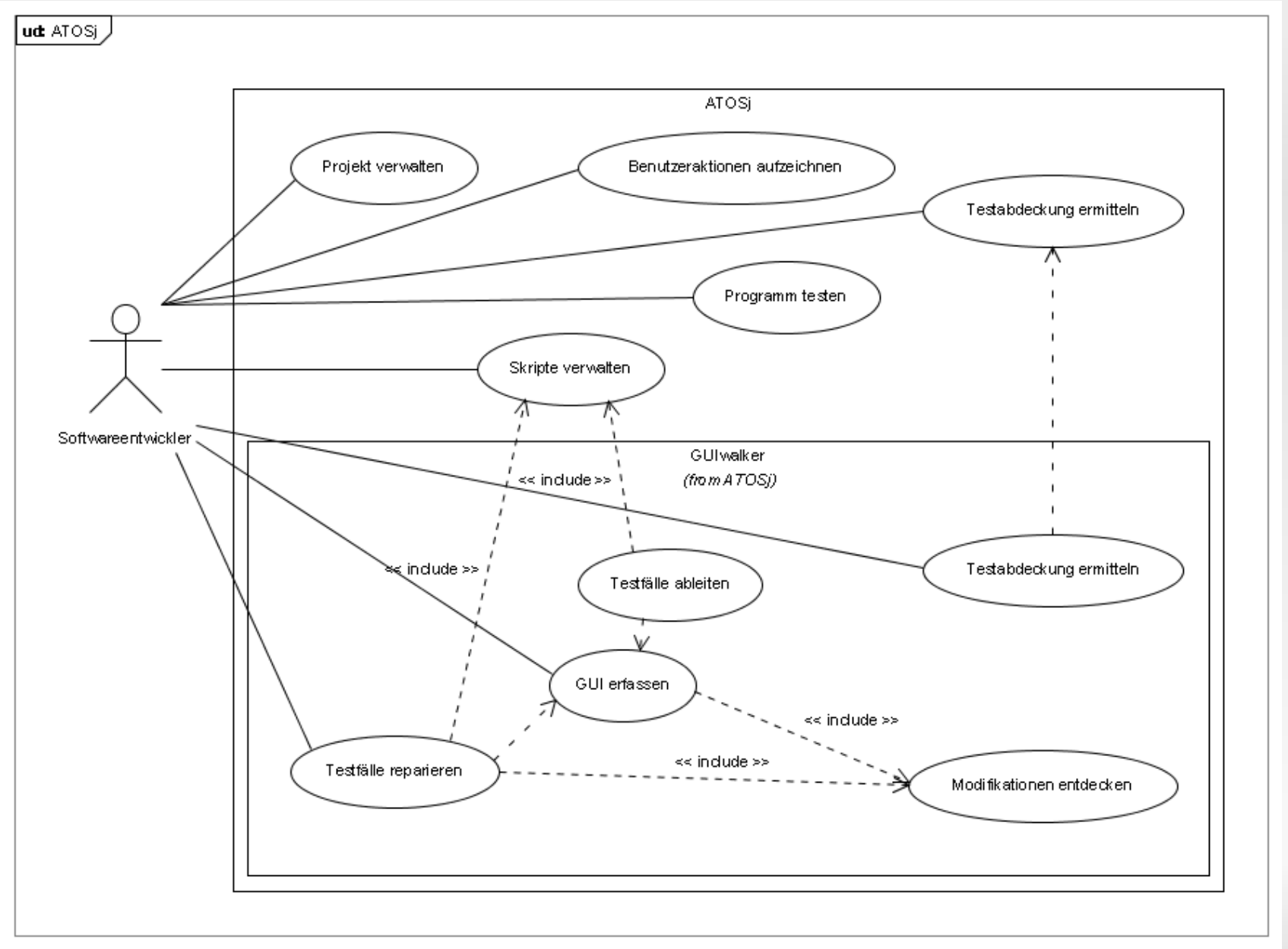
Zielbestimmung
Abgrenzungskriterien

- Abgrenzungskriterien
  - Die Testsequenzausführung, Testskriptbearbeitung und Projektverwaltung obliegt ATOSj.
  - Es werden bei der GUI-Erfassung keine Benutzeraktion aufgezeichnet.

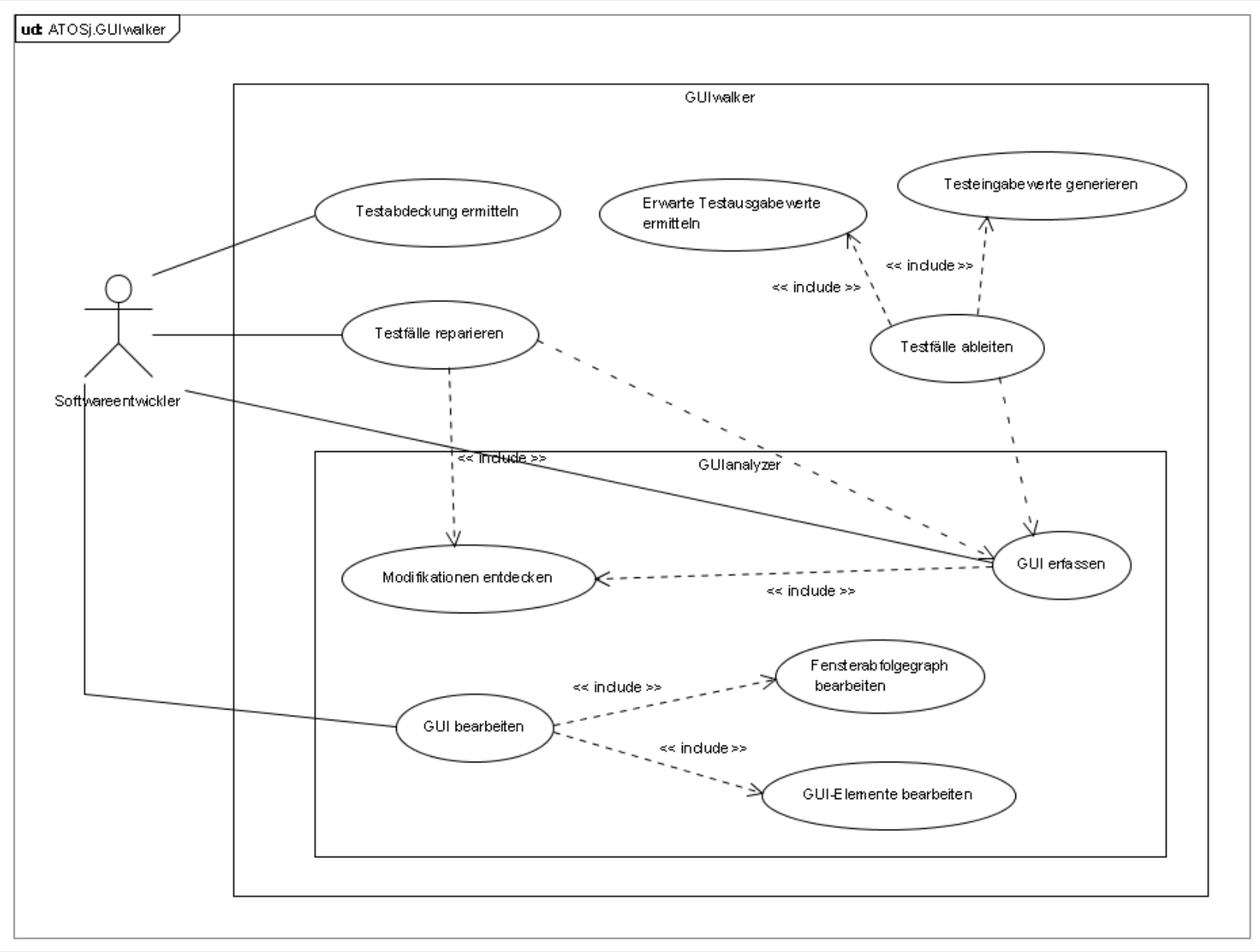
Qualitätsanforderungen
GUIwalker

Produktaualität	Sehr gut	Gut	Normal	Nicht relevant
Funktionalität				
Angemessenheit		x		
Richtigkeit		x		
Interoperabilität			x	
Ordnungsmäßigkeit				
Sicherheit				x
Zuverlässigkeit				
Reife			x	
Fehlertoleranz		x		
Wiederherstellbarkeit			x	
Benutzbarkeit				
Verständlichkeit	x			
Erlernbarkeit		x		
Bedienbarkeit	x			
Effizienz				
Zeitverhalten			x	
Verbrauchsverhalten			x	
Änderbarkeit				
Analysierbarkeit	x			
Modifizierbarkeit	x			
Stabilität		x		
Prüfbarkeit	x			
Übertragbarkeit				
Anpassbarkeit			x	
Installierbarkeit			x	
Austauschbarkeit			x	

GUIwalker in ATOSj



UseCase GUIwalker
Version 0.5



Geschäftsprozess /F10/
GUI-Struktur erfassen 1

- GUI-Struktur erfassen: Vom Starten der SUT bis Anzeige des GUI-Modells
  - Normal:
    1. Vorgänger-GUI-Modell sichern.
    2. Starten der SUT.
    3. Alle in GUI erreichbaren Fenster öffnen und Reihenfolge merken,
    4. bei jedem Fenster dessen Widgets erfassen,
    5. bei jedem Fenster benötigte Eingaben zum Fortfahren zufällig generieren und eintragen.
    6. Gewonnenes GUI-Modell mit Vorgänger-GUI-Modell vergleichen und Änderungen anzeigen.
    7. Anzeige des gewonnen GUI-Modells als Graphenansicht.

Geschäftsprozess /F10/
GUI-Struktur erfassen 2

- GUI-Struktur erfassen
  - Erweiterung:
    - 6a. Manuelle Änderungen aus Vorgänger-Modell ggf. übernehmen.
  - Alternativen:
    - 2a. Wenn SUT bereits gestartet: SUT in Ausgangszustand bringen.
    - 5a. Benutzer bitten Eingaben zu tätigen.
    - 5b. Eingabewerte aus eine vorher angelegten Liste auswählen und eintragen.
    - 7a. Anzeige als Baumansicht.

GUI-Model

- Graph mit folgenden Informationen
  - Aufrufreihenfolge der Fenster
  - Aufruf modal/nicht-modal
  - Startfenster
- Fensteraufbauinformationen
  - Fenstereigenschaften
  - Enthaltene Widgets
- Widgetinformationen
  - Eigenschaften/Ereignisse

Produktdaten
Fensterabfolgegraphdaten

- /D10/ Liste aller Fenster
  - eindeutige Fenster-ID
  - enthaltene GUI-Elemente
  - ist es Startfenster?
- /D20/ Fensterabfolgebeziehung
  - Fenster-ID des vorhergehenden Fenster
  - Fenster-ID des darauf folgenden Fensters
  - Modus des Aufrufs (modal oder nicht)

Produktdaten
GUI-Elementdaten 1

- /D30/ Eigenschaften
  - eindeutiger Eigenschaftsname (ID)
  - zulässiger Wertebereich
  - Standardwert
- /D40/ Emittierte Ereignisse
  - Ereignis-ID
  - Ereigniskategorie
  - alle potentiellen Vorbedingungen,
  - alle möglichen Effekte

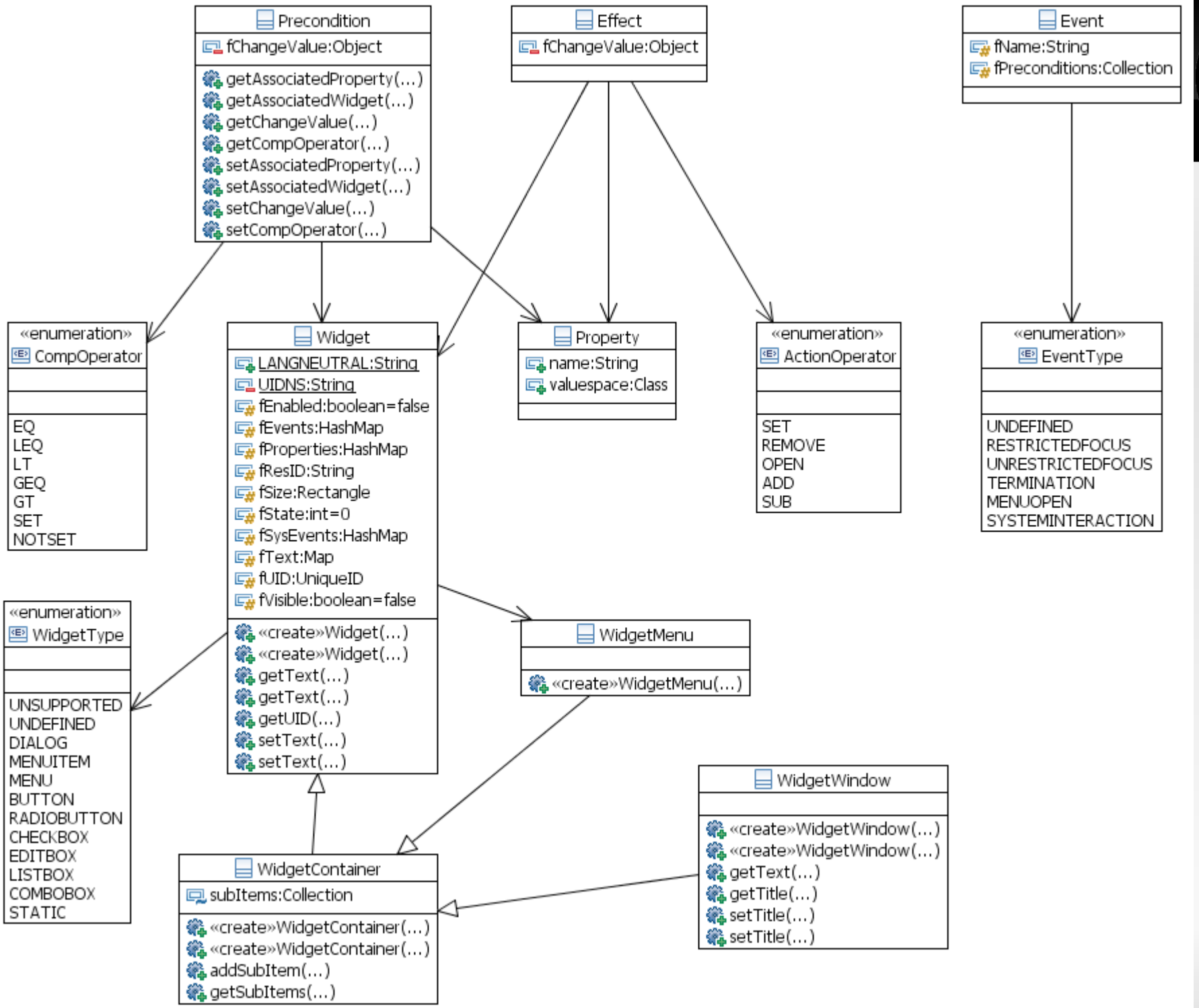
Produktdaten
GUI-Elementdaten 2

- /D41/ Vorbedingungen
  - von Vorbedingung verlangte Eigenschaft
  - Vergleichswert
  - Vergleichsoperator, mit der die Eigenschaft mit dem Vergleichswert verglichen werden soll
- /D42/ Effekte
  - vom Effekt betroffene Eigenschaft
  - Wert
  - Aktion, mit der der Wert auf die Eigenschaft angewendet werden soll.

Graphenarchitektur

Graphentoolkits

- JGraph -> *wird derzeit genutzt*
- JGraphT -> baut auf JGraph auf > langsam
- JUNG -> zu langsam, Darstellungsfehler
- GEF -> nur mit Eclipse
- GVF -> nicht gepflegt
- InfoVis -> Graphen nur Nebensache
- Prefuse -> Graphen nur Nebensache


Precondition
fChangeValue:Object
getAssociatedProperty(...)
getAssociatedWidget(...)
getChangeValue(...)
getCompOperator(...)
setAssociatedProperty(...)
setAssociatedWidget(...)
setChangeValue(...)
setCompOperator(...)
Effect
fChangeValue:Object
Event
fName:String
fPreconditions:Collection
«enumeration»
CompOperator
EQ
LEQ
LT
GEQ
GT
SET
NOTSET
Widget
LANGNEUTRAL:String
UIDNS:String
fEnabled:boolean=false
fEvents:HashMap
fProperties:HashMap
fResID:String
fSize:Rectangle
fState:int=0
fSysEvents:HashMap
fText:Map
fUID:UniqueID
fVisible:boolean=false
«create»Widget(...)
«create»Widget(...)
getText(...)
getText(...)
getUID(...)
setText(...)
setText(...)
Property
name:String
valuespace:Class
«enumeration»
ActionOperator
SET
REMOVE
OPEN
ADD
SUB
«enumeration»
EventType
UNDEFINED
RESTRICTEDFOCUS
UNRESTRICTEDFOCUS
TERMINATION
MENUOPEN
SYSTEMINTERACTION
«enumeration»
WidgetType
UNSUPPORTED
UNDEFINED
DIALOG
MENUITEM
MENU
BUTTON
RADIOBUTTON
CHECKBOX
EDITBOX
LISTBOX
COMBOBOX
STATIC
WidgetMenu
«create»WidgetMenu(...)
WidgetWindow
«create»WidgetWindow(...)
«create»WidgetWindow(...)
getText(...)
getTitle(...)
setTitle(...)
setTitle(...)
WidgetContainer
subItems:Collection
«create»WidgetContainer(...)
«create»WidgetContainer(...)
addSubItem(...)
getSubItems(...)

Erfassen der GUI

- SUT in anderem Prozeß starten
  - Tiefe-zuerst-Suche
  - Widgets jedes Fensters erfassen und dann
  - Spekulation, dass „...“ am Ende der Beschriftung für zu öffnendes Fenster stehen
  - Alle Widgets „drücken“ und Reaktionen erkennen/merken
    - Bei Programmabbruch: Programm neu starten und bis zu dieser Stelle-1 vor spulen

Knackpunkte

- Erkennung aller Ereignisse mit
  - Vorbedingungen
  - Effekten
- Ereignisklassenzuordnung
- Fortsetzung der GUI-Erfassung nach Absturz der SUT

Grenzen

- Keine Automatisierung für
  - Video, Soundausgabe
  - Zeichenprogramme (auch Diagrammeditoren)
  - Webapplikationen
  - Widgets, die nicht in die Standardkategorien passen

Testszenarien

- Testdurchführung in ATOSj ohne Benutzung von GUIwalker
- Laden und Speichern von manuell eingegebenen GUI-Modellen
- Erfassen, Speichern des GUI-Modell eines simplen Beispielprogramms
- Erfassen, Speichern des GUI-Modell eines komplexen Beispielprogramms
- Erfassen und Vergleichen zweier Versionen eines Beispielprogramms

Überblick

- Begriffsklärung
- Motivation
- Automatische Testfallgenerierung
- GUIwalker
- Ausblick

Ausblick
SampleApps-Modul

- Beispielprogramme mit
  - spezifiziertem Verhalten/Fehlern
  - Einfache und komplexe Programme
  - Identische Implementation auf verschiedenen Systemplattformen (Windows, Linux, Java(Swing, SWT), .Net)

Ausblick
GUIwalker 1.0

- Generierung/Ermittlung von Eingabewerten und erwarteten Ausgabewerten
- Automatische Testfallreparatur
- Überdeckungsanalyse, der generierten Testfälle

Ausblick
Spin-Offs

- ATOSj: Weitere GUI-Plattformen unterstützen
  - Windows/.NET/Linux/Mac-OS
- SampleApps-Modul ausbauen
  - Windows: Win32, MFC, QT, GTK
  - Linux: GTK, GNOME, KDE, QT
  - .NET: WinForms, GTK#
- ATOSj nach Eclipse-RCP portieren

ENDE