Fraunhofer Einrichtung Systeme der Kommunikationstechnik

Berücksichtigung zeitlicher Anforderungen bei der Testgenerierung für den Integrationstest

Das 26. Treffen der GI-TAV Fachgruppe
„Test, Analyse und Verifikation von Software“

Witali Aswolinskiy, Yingfan Lei, Mike Heidrich

6.12.2007

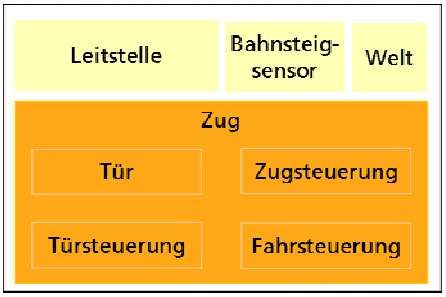
Inhalt

- Einleitung und Motivation der Arbeit
- Problemstellung
  - Testgenerierung für den Integrationstest mit All-Round-Trips
  - Annahme zur Sicherstellung von All-Round-Trips
- Integrationstest mit Berücksichtigung zeitlicher Anforderungen
  - Modellierung zeitlicher Anforderungen
  - Zufallsbasierte bzw. evolutionäre Testgenerierung
- Schlussfolgerung

Einleitung

Testen reaktiver Software-Systeme

- Module/Komponenten können verteilt sein
- Interaktion zwischen den Testfällen und dem zu testenden System
- **Implizite zeitliche Anforderungen:** Reihenfolge von Funktionsaufrufen bzw. Ereignissen
- **Explizite zeitliche Anforderungen:** Vorgegebene Zeitrahmen für Abstände zwischen Funktionsaufrufen bzw. Ereignissen
- Fallstudie: U-Bahn-System



Fraunhofer ESK Einrichtung Systeme der Kommunikationstechnik

Motivation

Modellbasierte Testgenerierung für den Integrationstest mit Berücksichtigung zeitlicher Anforderungen

- UML: eine semi-formale Modellierungssprache für Software-Design
- Zustandsdiagramme: das Verhalten eines zu testenden Systems
- Testgenerierung mitttels All-Round-Trips: Durchlauf im Zustandsdiagramm

  Allgemeine Überdeckung: Zustand, Transition

  Variable im `Guard`
- Timing: die Reihenfolge von Funktionsaufrufen bzw. Ereignissen und der zeitliche Abstand zwischen ihnen.

  Beispiele:
  Nachdem sich eine Tür geöffnet hat, soll sie sich nach 6 Sekunden wieder schließen. [Zeitereignis]

  Das Öffnen der Tür darf nicht weniger als 900 ms und nicht mehr als 1200 ms verbrauchen. [Zeitbedingung]

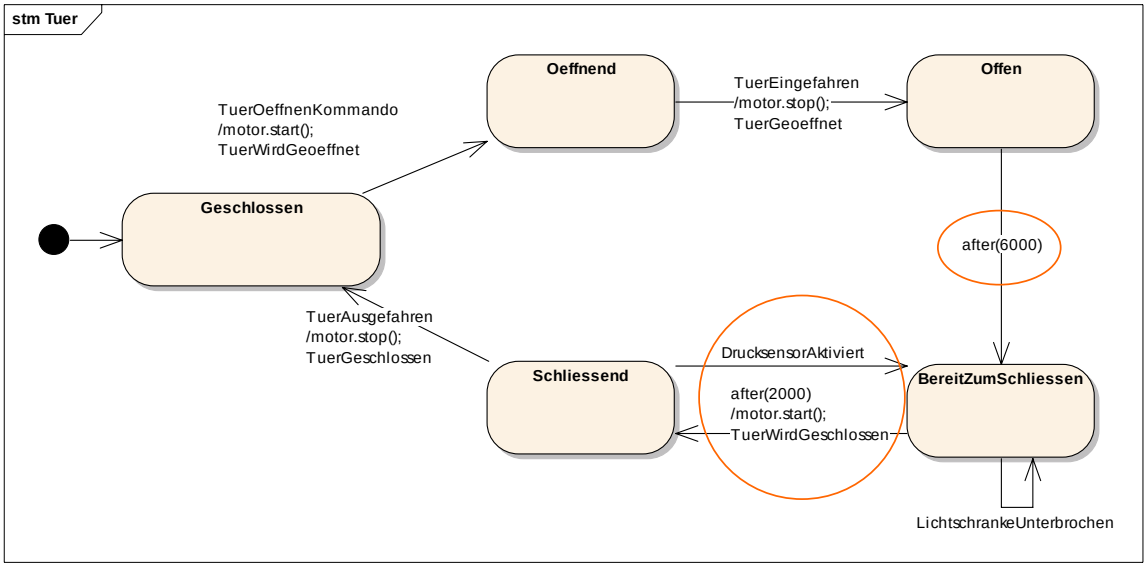
  Nachdem ein Zug Angehalten hat, darf er erst losfahren, wenn seine Türen verriegelt sind. [Reihenfolgebedingung]

Fraunhofer ESK Einrichtung Systeme der Kommunikationstechnik

Testgenerierung mit All-Round-Trips

Testgenerierung mit All-Round-Trips für das Beispiel: UBahn-Tür

- Pfadsuche im Zustandsraum
- Zustands- und Transitionsüberdeckung



Generierte Testfälle für den Komponententest

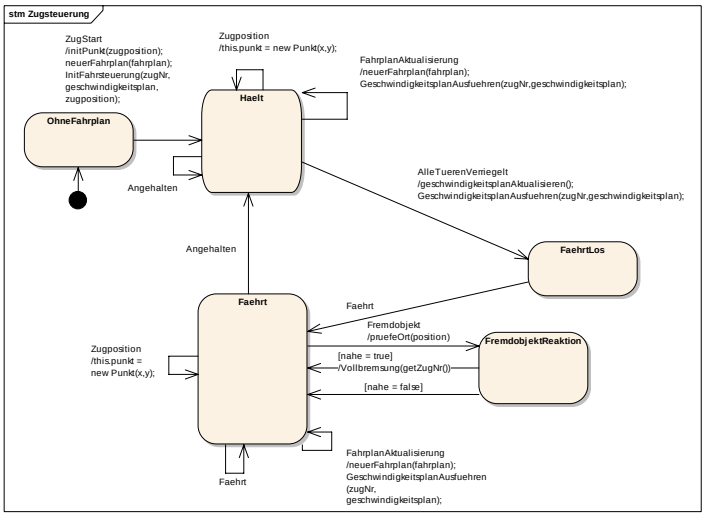
Ein generierter Testfall:

```
public void testTC1() {
    TuerOeffnenKommando tuerOeffnenKommando;
    TimeEvent timeEvent;
    LichtschrankeUnterbrochen lichtschrankeUnterbrochen;
    tuerOeffnenKommando = new TuerOeffnenKommando(1952892356);
    tuerOeffnenKommando.setTuerNr(getTC().getTuerNr());
    send(tuerOeffnenKommando);
    waitFor("Offen");
    assertInState(TuerStates.Offen);
    timeEvent = new TimeEvent(6000L);
    send(timeEvent);
    waitFor("BereitZumSchliessen");
    assertInState(TuerStates.BereitZumSchliessen);
    lichtschrankeUnterbrochen = new LichtschrankeUnterbrochen(465784233,
            -1064639055);
    lichtschrankeUnterbrochen.setTuerNr(getTC().getTuerNr());
    send(lichtschrankeUnterbrochen);
    waitFor("BereitZumSchliessen");
    assertInState(TuerStates.BereitZumSchliessen);
    checkObserverFaults();
}
```

Problemstellung

Testgenerierung für den Integrationstest mit Berücksichtigung zeitlicher Anforderungen

- Modellierung zeitlicher Anforderungen: einfach, standardisiert, aussagekräftig
- Generierung für komplexe Systeme: Ereignisse aus unterschiedlichen Komponenten beeinflussen den Ablauf. z.B. Variable `nahe im Guard`



Fraunhofer ESK
Einrichtung
Systeme der
Kommunikationstechnik

www.esk.fraunhofer.de

stm Zugsteuerung
ZugStart
/initPunkt(zugposition);
neuerFahrplan(fahrplan);
InitFahrsteuerung(zugNr,
geschwindigkeitsplan,
zugposition);
Zugposition
/this.punkt = new Punkt(x,y);
FahrplanAktualisierung
/neuerFahrplan(fahrplan);
GeschwindigkeitsplanAusfuehren(zugNr,geschwindigkeitsplan);
OhneFahrplan
Haelt
Angehalten
AlleTuerenVerriegelt
/geschwindigkeitsplanAktualisieren();
GeschwindigkeitsplanAusfuehren(zugNr,geschwindigkeitsplan);
Angehalten
FaehrtLos
Faehrt
Faehrt
Fremdobjekt
/pruefeOrt(position)
FremdobjektReaktion
Zugposition
/this.punkt =
new Punkt(x,y);
[nahe = true]
/Vollbremsung(getZugNr())
[nahe = false]
FahrplanAktualisierung
/neuerFahrplan(fahrplan);
GeschwindigkeitsplanAusfuehren
(zugNr,
geschwindigkeitsplan);
Faehrt

Testgenerierung für den Integrationstest

Unsere Lösung

- Modellierung zeitlicher Anforderungen durch Statechart Observer
- Zufallsbasierte bzw. evolutionäre Generierung von Ereignissequenzen

Statechart Observer

- Selbst Zustandsautomat
- Beobachtet andere Zustandsautomaten – die ausgehenden und eingehenden Ereignisse, und die Zeitbedingung

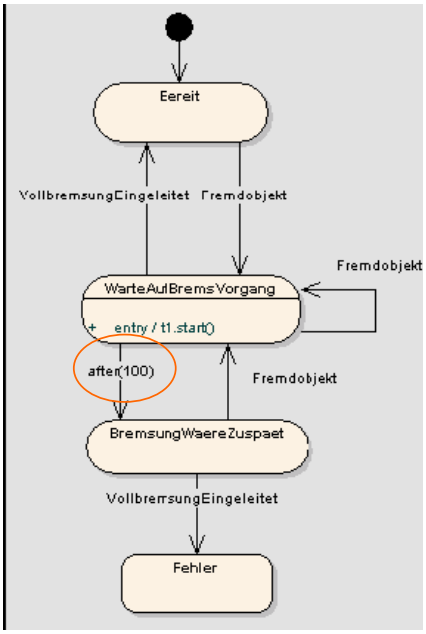
Testgenerierung für den Integrationstest mit Berücksichtigung zeitlicher Anforderungen

- Komposition mehrer Zustandsräume
- Generierung mit Pfadsuche
- Sicherstellung von All-Round-Trips: hoher Aufwand

Modellierung von Statechart Observer

Statechart Observer modelliert die zeitlichen Anforderungen

- Beispiel: Statechart Observer für Vollbremsung
- Modellierung: Zeitbedingung, Reihenfolgebedingung



Generierung von Ereignissequenzen

Ziel: Reihenfolge und Zeitabstand von Ereignissen sollen festgelegt werden. Fehler sollen möglichst schnell gefunden werden.

- Analytisch:
  Beobachtung der Zustandsänderungen von Komponenten
  Ereignisse werden immer so gesendet, dass die Komponenten sie bearbeiten
  Problem: Wie kann die Verzögerung während der Ausführung behandelt werden?
- Zufällig:
  Ereignisse werden immer wieder neu generiert
  Problem: Werden die Ereignisse von den Komponenten abprallen?
- Evolutionär:
  Iterative und inkrementelle Verbesserung der Ereignissequenzen mit den Informationen aus den vorherigen Testausführungen

Testverfahren mit der Fallstudie

- Generierung der Testfälle zuerst per Zufall
- Ausführen der Testfälle
- Berechnung der Fitness-Funktion:
  Bewertung der Ereignissequenzen aufgrund der erhaltenen Ereignisse von den Statechart Observern und Komponenten
  - Anzahl der Zustandsübergänge des zu testenden Systems
  - Anzahl der Zustandsübergänge des Observers
- Selektion: Auswahl der erfolgreichsten Testfälle
- Reproduktion von jeweils zwei Testfällen: Crossover
- Mutation der erzeugten Testfälle: Reihenfolge der Ereignisse, Inhalt der Ereignisse, usw.
- Ausführen der nächsten Generation der Testfälle

Schlussfolgerung

Basiert auf Statechart Observern werden die eingebauten Fehler im Beispiel U-Bahn mit Zufallsgenerierung schneller gefunden.

Gründe:

- Kleine Zustandsdiagramme:
  Das evolutionäre Verfahren optimierte das Timing: Ereignisse wurden so gesendet, dass Transitionen stattfinden konnten.
- Großer Datenraum
  Die jetzige Implementierung der evolutionärer Generierung veränderte die erzeugten Ereignisse nach der ersten, zufallsbasierten Erzeugung nur unwesentlich. D.h. die Abdeckung des Datenraums ist eingeschränkt.

Diskussion:

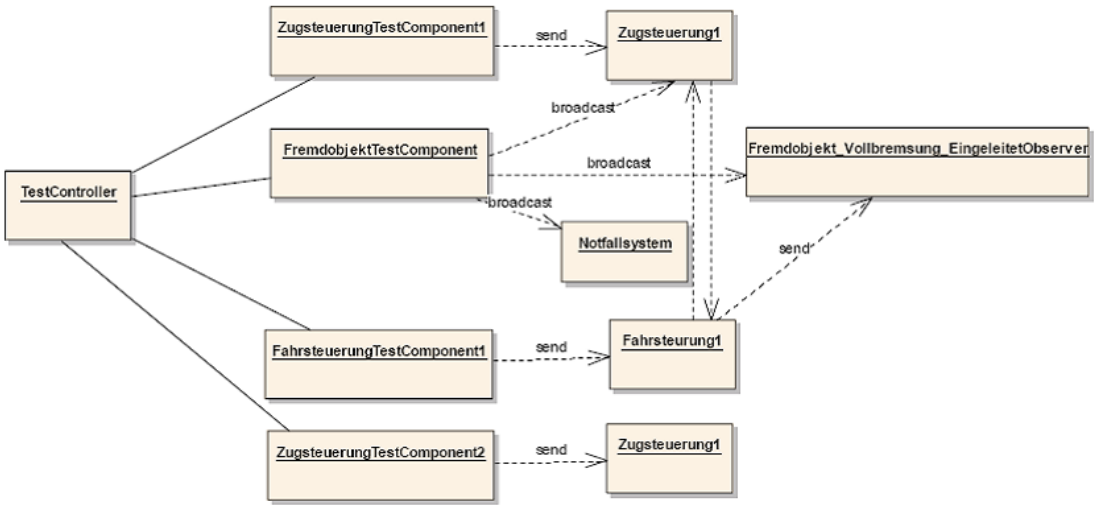
- Zeitverbrauch minimieren
  Die Testfälle für die U-Bahn verbrauchten mehrere Sekunden
  -> Test zeitlicher Anforderungen in größeren Systemen langwierig
- Allgemeingültigkeit: wann und wie kann evolutionäre Testgenerierung günstig sein?
- Allgemeingültigkeit: Integrationstest nicht-funktionaler Anforderungen in **komplexen**, reaktiven Systemen.

Fraunhofer ESK
Einrichtung
Systeme der
Kommunikationstechnik

Vielen Dank für Ihre Aufmerksamkeit!

www.esk.fraunhofer.de

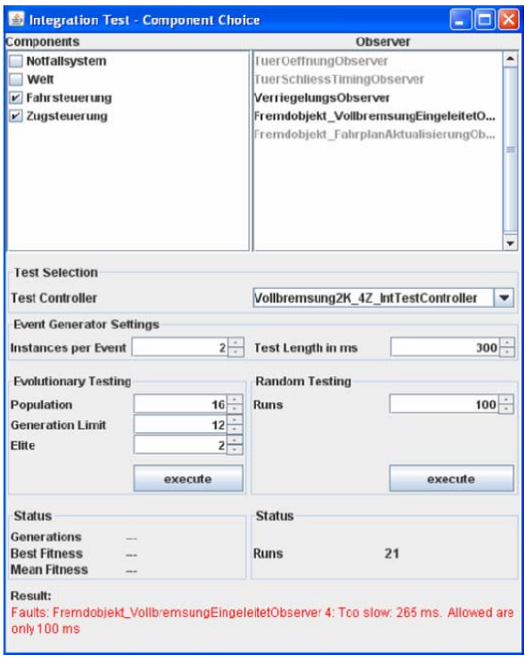
Anhang



Testfall und TestController

Fraunhofer ESK Einrichtung Systeme der Kommunikationstechnik

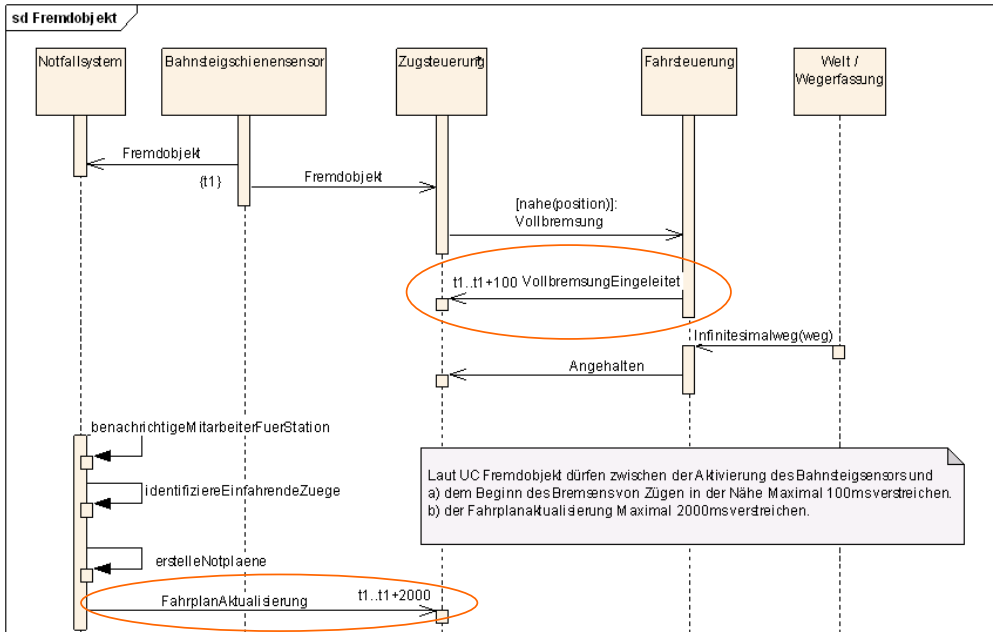
Anhang



GUI für das Testverfahren

Fraunhofer ESK Einrichtung Systeme der Kommunikationstechnik

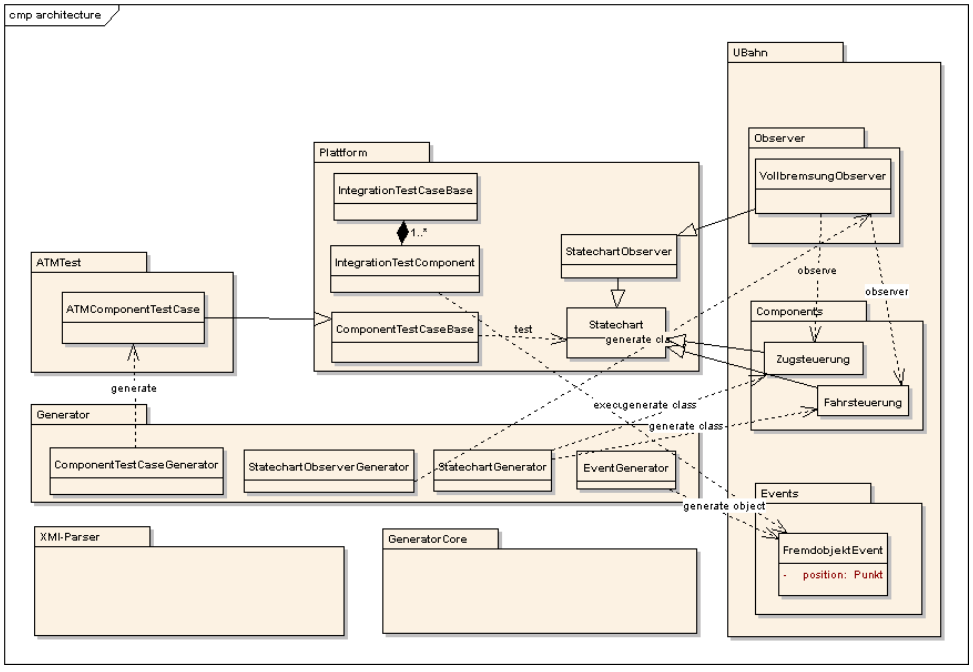
Anhang



Sequenzdiagramm für den Fall des Auftretens eines Fremdobjektes auf dem Gleis

Fraunhofer ESK Einrichtung Systeme der Kommunikationstechnik

Anhang



Design des ESK-Testgenerators und der UBahn-Fallstudie

Fraunhofer ESK Einrichtung Systeme der Kommunikationstechnik