Software Engineering in der Praxis

Praktische Übungen

Objektorientierte Analyse

Florin Pinte Marc Spisländer

Lehrstuhl für Software Engineering
Friedrich-Alexander-Universität Erlangen-Nürnberg

Zweck der Objektorientierten Analyse

Was soll das System leisten (nicht *wie*)

- Erkundung der Natur des gegebenen Problems
- Systemgrenzen, Außenschnittstellen identifizieren
- Hilfe bei der “Zerlegung” des Systems in Teile
- Abläufe, (grobe) Kontrollflüsse, Nebenläufigkeit
- Schaffung eines Ausgangspunktes für das Design

Lernprozeß, kein Design!

Zweck der Objektorientierten Analyse

Was soll das System leisten (nicht *wie*)

- Erkundung der Natur des gegebenen Problems
- Systemgrenzen, Außenschnittstellen identifizieren
- Hilfe bei der “Zerlegung” des Systems in Teile
- Abläufe, (grobe) Kontrollflüsse, Nebenläufigkeit
- Schaffung eines Ausgangspunktes für das Design

Lernprozeß, kein Design!

Zweck der Objektorientierten Analyse

Was soll das System leisten (nicht *wie*)

- Erkundung der Natur des gegebenen Problems
- Systemgrenzen, Außenschnittstellen identifizieren
- Hilfe bei der “Zerlegung” des Systems in Teile
- Abläufe, (grobe) Kontrollflüsse, Nebenläufigkeit
- Schaffung eines Ausgangspunktes für das Design

Lernprozeß, kein Design!

Zweck der Objektorientierten Analyse

Was soll das System leisten (nicht *wie*)

- Erkundung der Natur des gegebenen Problems
- Systemgrenzen, Außenschnittstellen identifizieren
- Hilfe bei der “Zerlegung” des Systems in Teile
- Abläufe, (grobe) Kontrollflüsse, Nebenläufigkeit
- Schaffung eines Ausgangspunktes für das Design

Lernprozeß, kein Design!

Zweck der Objektorientierten Analyse

Was soll das System leisten (nicht *wie*)

- Erkundung der Natur des gegebenen Problems
- Systemgrenzen, Außenschnittstellen identifizieren
- Hilfe bei der “Zerlegung” des Systems in Teile
- Abläufe, (grobe) Kontrollflüsse, Nebenläufigkeit
- Schaffung eines Ausgangspunktes für das Design

Lernprozeß, kein Design!

Zweck der Objektorientierten Analyse

Was soll das System leisten (nicht *wie*)

- Erkundung der Natur des gegebenen Problems
- Systemgrenzen, Außenschnittstellen identifizieren
- Hilfe bei der “Zerlegung” des Systems in Teile
- Abläufe, (grobe) Kontrollflüsse, Nebenläufigkeit
- Schaffung eines Ausgangspunktes für das Design

Lernprozeß, kein Design!

Zweck der Objektorientierten Analyse

Was soll das System leisten (nicht *wie*)

- Erkundung der Natur des gegebenen Problems
- Systemgrenzen, Außenschnittstellen identifizieren
- Hilfe bei der “Zerlegung” des Systems in Teile
- Abläufe, (grobe) Kontrollflüsse, Nebenläufigkeit
- Schaffung eines Ausgangspunktes für das Design

Lernprozeß, kein Design!

Use-Case Analyse

UML Anwendungsfall-Diagramme

- Fokus auf das äußere funktionale Verhalten (“Feature”)
- Nutzer, Akteure
- Anwendungsfälle
- Systemgrenzen
- Beziehungen zwischen Akteuren und Anwendungsfällen

Systematische Erhebung der Ereignisse, die die Systemgrenzen passieren

Use-Case Analyse

UML Anwendungsfall-Diagramme

- Fokus auf das äußere funktionale Verhalten (“Feature”)
- Nutzer, Akteure
- Anwendungsfälle
- Systemgrenzen
- Beziehungen zwischen Akteuren und Anwendungsfällen

Systematische Erhebung der Ereignisse, die die Systemgrenzen passieren

Use-Case Analyse

UML Anwendungsfall-Diagramme

- Fokus auf das äußere funktionale Verhalten (“Feature”)
- Nutzer, Akteure
- Anwendungsfälle
- Systemgrenzen
- Beziehungen zwischen Akteuren und Anwendungsfällen

Systematische Erhebung der Ereignisse, die die Systemgrenzen passieren

Use-Case Analyse

UML Anwendungsfall-Diagramme

- Fokus auf das äußere funktionale Verhalten (“Feature”)
- Nutzer, Akteure
- Anwendungsfälle
- Systemgrenzen
- Beziehungen zwischen Akteuren und Anwendungsfällen

Systematische Erhebung der Ereignisse, die die Systemgrenzen passieren

Use-Case Analyse

UML Anwendungsfall-Diagramme

- Fokus auf das äußere funktionale Verhalten (“Feature”)
- Nutzer, Akteure
- Anwendungsfälle
- Systemgrenzen
- Beziehungen zwischen Akteuren und Anwendungsfällen

Systematische Erhebung der Ereignisse, die die Systemgrenzen passieren

Use-Case Analyse

UML Anwendungsfall-Diagramme

- Fokus auf das äußere funktionale Verhalten (“Feature”)
- Nutzer, Akteure
- Anwendungsfälle
- Systemgrenzen
- Beziehungen zwischen Akteuren und Anwendungsfällen

Systematische Erhebung der Ereignisse, die die Systemgrenzen passieren

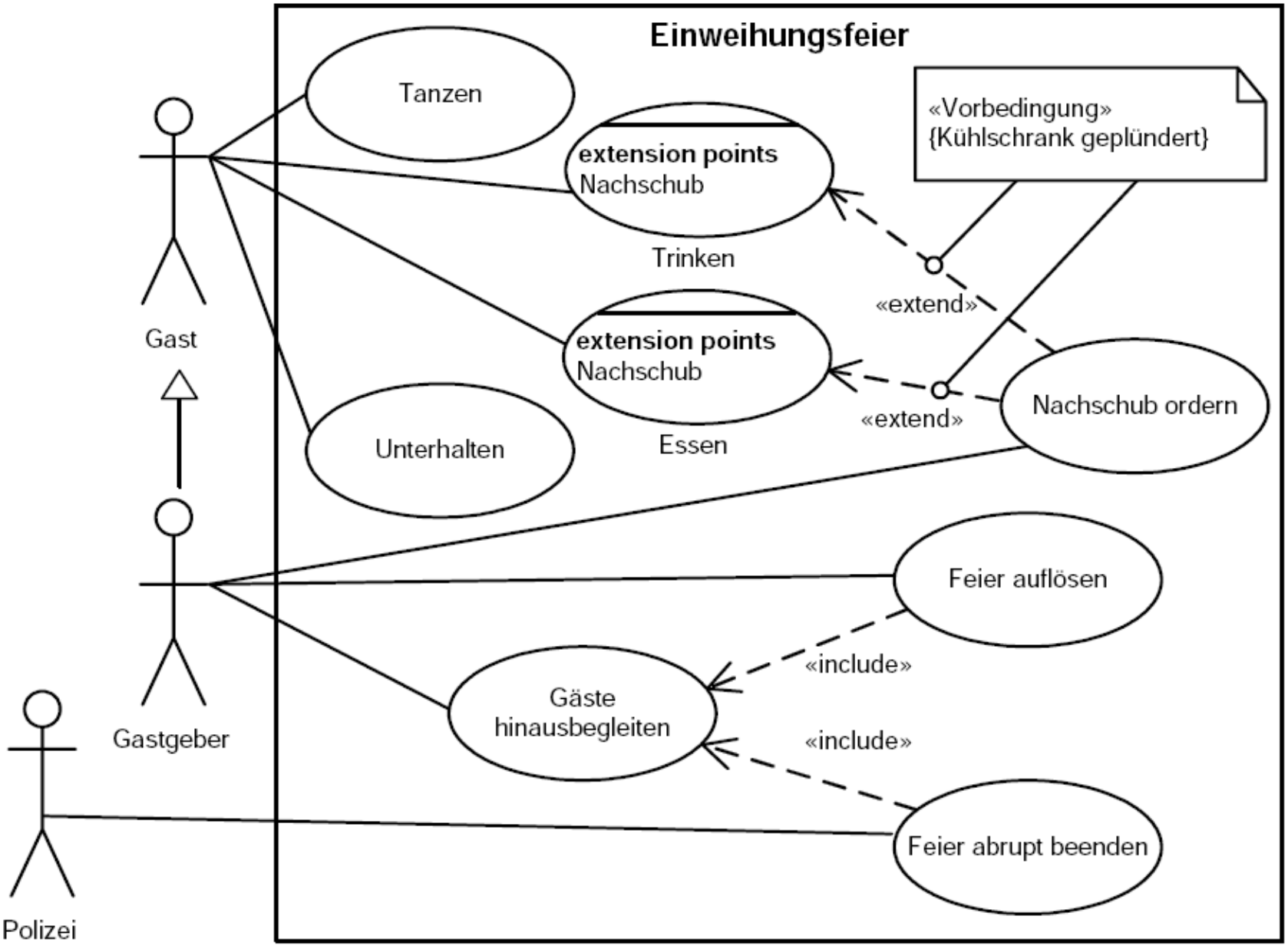
Use-Case Analyse

UML Anwendungsfall-Diagramme

- Fokus auf das äußere funktionale Verhalten (“Feature”)
- Nutzer, Akteure
- Anwendungsfälle
- Systemgrenzen
- Beziehungen zwischen Akteuren und Anwendungsfällen

Systematische Erhebung der Ereignisse, die die Systemgrenzen passieren

Beispiel


Einweihungsfeier
Tanzen
extension points
Nachschub
Trinken
«Vorbedingung»
{Kühlschrank geplündert}
«extend»
extension points
Nachschub
Essen
«extend»
Nachschub ordern
Unterhalten
Gast
Gastgeber
Feier auflösen
«include»
Gäste
hinausbegleiten
«include»
Feier abrupt beenden
Polizei

Analyse der automatischen Produktionszelle

Use-Case Diagramme?

- Betrachte Nachbarsysteme als Nutzer
- Zeichne UseCase-Diagramme für die Systemteile...

Oder wähle einen geeigneteren Diagrammtyp aus!

Analyse der automatischen Produktionszelle

Use-Case Diagramme?

- Betrachte Nachbarsysteme als Nutzer
- Zeichne UseCase-Diagramme für die Systemteile...

Oder wähle einen geeigneteren Diagrammtyp aus!

Analyse der automatischen Produktionszelle

Use-Case Diagramme?

- Betrachte Nachbarsysteme als Nutzer
- Zeichne UseCase-Diagramme für die Systemteile...

Oder wähle einen geeigneteren Diagrammtyp aus!

Analyse der automatischen Produktionszelle

Use-Case Diagramme?

- Betrachte Nachbarsysteme als Nutzer
- Zeichne UseCase-Diagramme für die Systemteile...

Oder wähle einen geeigneteren Diagrammtyp aus!

Aktivitätsdiagramme

Fokus auf Abläufe, die im System stattfinden

- Geschäftsprozesse, Systemprozesse, Methoden einer Klasse
- Eignet sich für Analyse *und* Design
- Aktivität := Menge potentieller Abläufe
- Kontrollelemente: Verzweigung, Vereinigung, Fork, Join
- Elemente: Aktionen, Kanten, Objektknoten, [Pin-Notation ...]
- Kontrolltoken, Datentoken
- uvm.

Inspiriert von Petri-Netzen

Aktivitätsdiagramme

Fokus auf Abläufe, die im System stattfinden

- Geschäftsprozesse, Systemprozesse, Methoden einer Klasse
- Eignet sich für Analyse *und* Design
- Aktivität := Menge potentieller Abläufe
- Kontrollelemente: Verzweigung, Vereinigung, Fork, Join
- Elemente: Aktionen, Kanten, Objektknoten, [Pin-Notation ...]
- Kontrolltoken, Datentoken
- uvm.

Inspiriert von Petri-Netzen

Aktivitätsdiagramme

Fokus auf Abläufe, die im System stattfinden

- Geschäftsprozesse, Systemprozesse, Methoden einer Klasse
- Eignet sich für Analyse *und* Design
- Aktivität := Menge potentieller Abläufe
- Kontrollelemente: Verzweigung, Vereinigung, Fork, Join
- Elemente: Aktionen, Kanten, Objektknoten, [Pin-Notation ...]
- Kontrolltoken, Datentoken
- uvm.

Inspiriert von Petri-Netzen

Aktivitätsdiagramme

Fokus auf Abläufe, die im System stattfinden

- Geschäftsprozesse, Systemprozesse, Methoden einer Klasse
- Eignet sich für Analyse *und* Design
- Aktivität := Menge potentieller Abläufe
- Kontrollelemente: Verzweigung, Vereinigung, Fork, Join
- Elemente: Aktionen, Kanten, Objektknoten, [Pin-Notation ...]
- Kontrolltoken, Datentoken
- uvm.

Inspiriert von Petri-Netzen

Aktivitätsdiagramme

Fokus auf Abläufe, die im System stattfinden

- Geschäftsprozesse, Systemprozesse, Methoden einer Klasse
- Eignet sich für Analyse *und* Design
- Aktivität := Menge potentieller Abläufe
- Kontrollelemente: Verzweigung, Vereinigung, Fork, Join
- Elemente: Aktionen, Kanten, Objektknoten, [Pin-Notation ...]
- Kontrolltoken, Datentoken
- uvm.

Inspiriert von Petri-Netzen

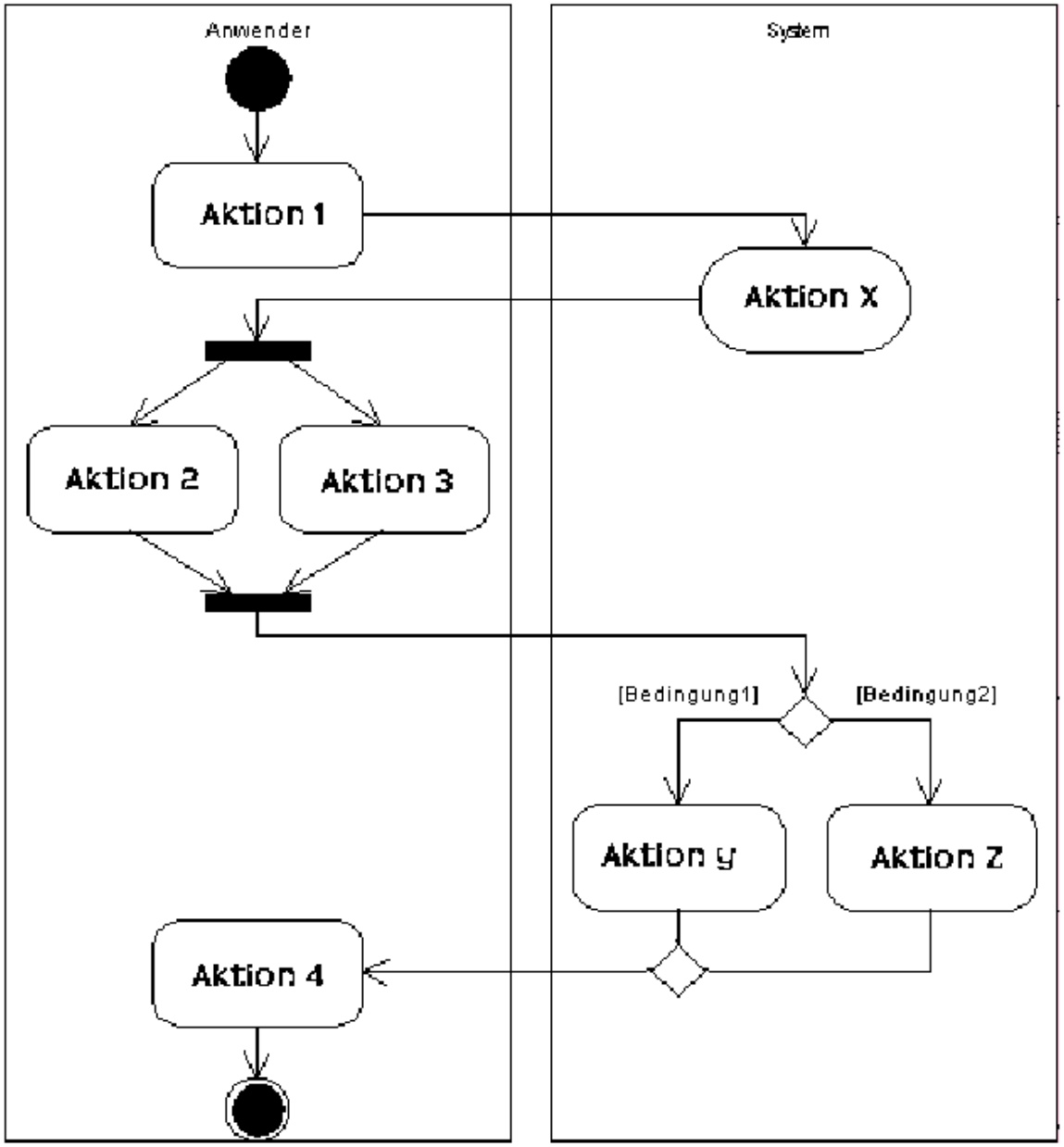
Aktivitätsdiagramme

Fokus auf Abläufe, die im System stattfinden

- Geschäftsprozesse, Systemprozesse, Methoden einer Klasse
- Eignet sich für Analyse *und* Design
- Aktivität := Menge potentieller Abläufe
- Kontrollelemente: Verzweigung, Vereinigung, Fork, Join
- Elemente: Aktionen, Kanten, Objektknoten, [Pin-Notation ...]
- Kontrolltoken, Datentoken
- uvm.

Inspiriert von Petri-Netzen

Aktivitätsdiagramm: Kontrollfluß mit “Swim-Lanes”



Aktivitätsdiagramme

Fokus auf Abläufe, die im System stattfinden

- Geschäftsprozesse, Systemprozesse, Methoden einer Klasse
- Eignet sich für Analyse *und* Design
- Aktivität := Menge potentieller Abläufe
- Kontrollelemente: Verzweigung, Vereinigung, Fork, Join
- Elemente: Aktionen, Kanten, Objektknoten, [Pin-Notation ...]
- Kontrolltoken, Datentoken
- uvm.

Inspiriert von Petri-Netzen

UML-Zustandsautomaten

- Zustände: Name, Verhalten $(3\times)$
- Übergänge: Trigger [Guards] / Effect
- Entscheidungen
- Pseudozustände, Historien, Hierarchien, Regionen ...

UML-Zustandsautomaten

- Zustände: Name, Verhalten $(3\times)$
- Übergänge: Trigger [Guards] / Effect
- Entscheidungen
- Pseudozustände, Historien, Hierarchien, Regionen ...

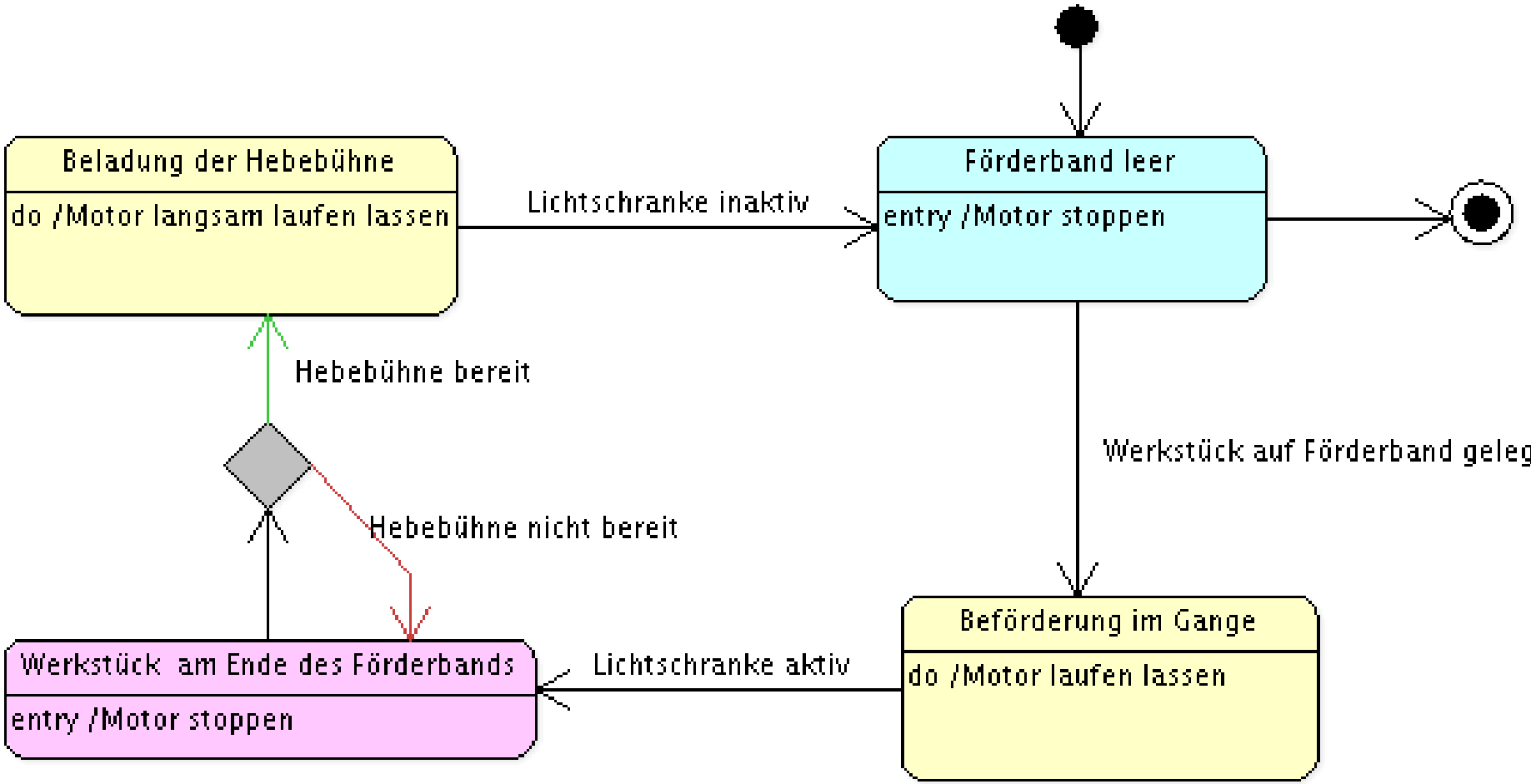
UML-Zustandsautomaten

- Zustände: Name, Verhalten $(3\times)$
- Übergänge: Trigger [Guards] / Effect
- Entscheidungen
- Pseudozustände, Historien, Hierarchien, Regionen ...

UML-Zustandsautomaten

- Zustände: Name, Verhalten $(3\times)$
- Übergänge: Trigger [Guards] / Effect
- Entscheidungen
- Pseudozustände, Historien, Hierarchien, Regionen ...

Zustandsautomat



Werkzeuge (Auswahl)

- Topcased
- Fujaba
- StarUML
- Rational Software Architect
- Borland Together für Eclipse
- Magic Draw
- MID Innovator

Aufgaben

Analyse der Domäne mit Hilfe verschiedener Diagrammtypen:

- Anwendungsfalldiagramm
- Aktivitätsdiagramm
- Zustandsdiagramm