

Telling TestStories – Modellbasiertes Akzeptanz-Testen Serviceorientierter Systeme

Michael Felderer

Workshop „Requirements Engineering meets Testing“

Bad Honnef, 5. Juni 2008

Überblick

- **Grundbegriffe**
- **Motivation**
- **Telling TestStories**
 - Überblick
 - Methode
 - System- und Testmodell
 - Systemarchitektur
 - Technologien
 - Einordnung



Grundbegriffe

- **Systemtest**

- Test des gesamten Systems gegen seine Anforderungsspezifikation (Akzeptanztest)

- **Modellbasiertes Testen**

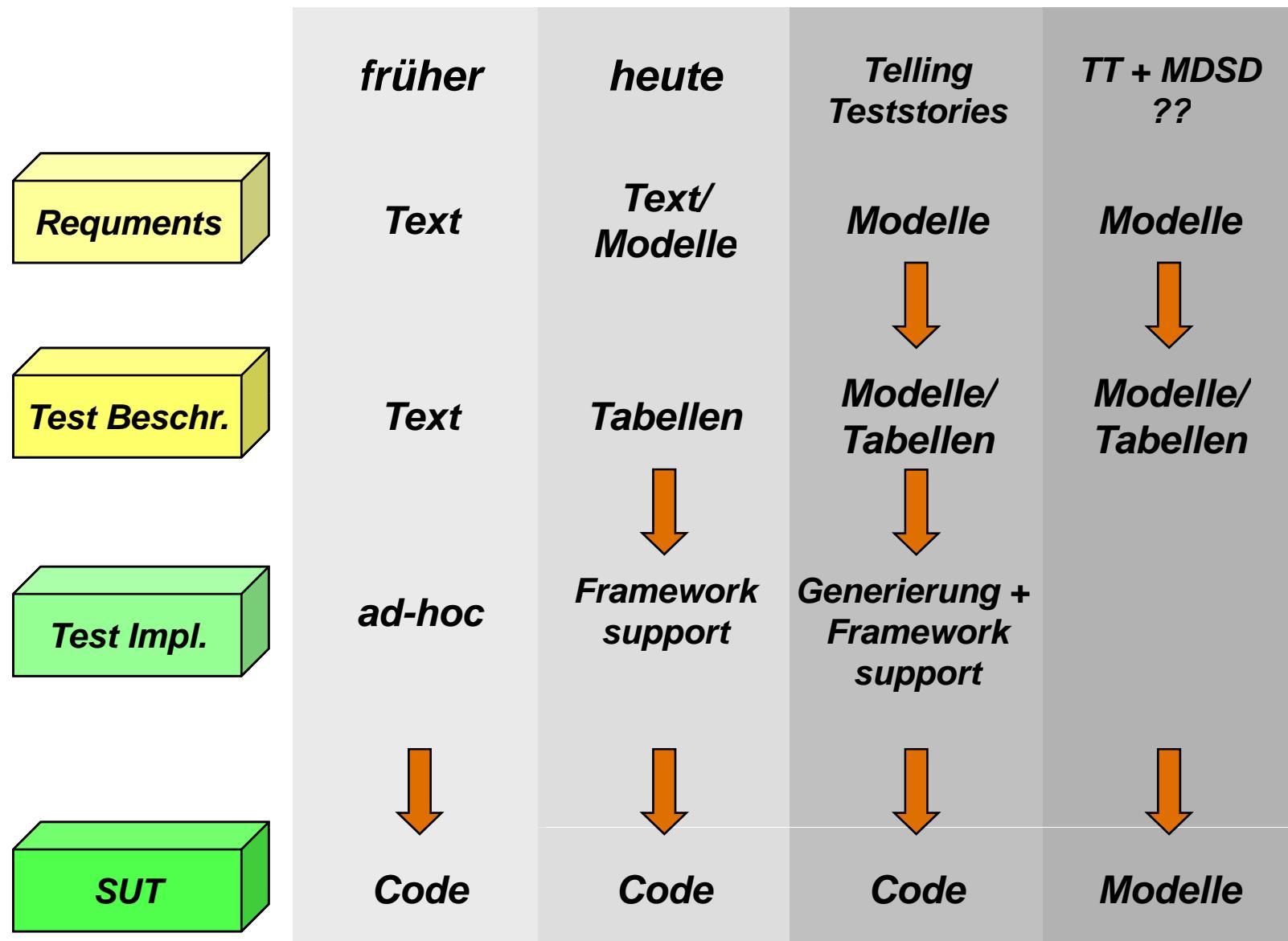
- Generierung bzw. Definition von Testfällen basierend auf einem Modell des zu testenden Systems (SUT)

- **Fit/Fitnesse**

- Automatisierung von Systemtests in Tabellenform (FIT) und Integration mit einem Wiki (Fitnesse)
- Verbindung zum SUT über sog. **Fixtures**
- keine Verbindung zur Spezifikation

CalculateDiscount	
amount	discount()
0.00	0.00
100.00	0.00
999.00	0.00
	50.00 <i>expected</i>
1000.00	
	0.0 <i>actual</i>
2000.00	100.00

Entwicklung von Systemtests

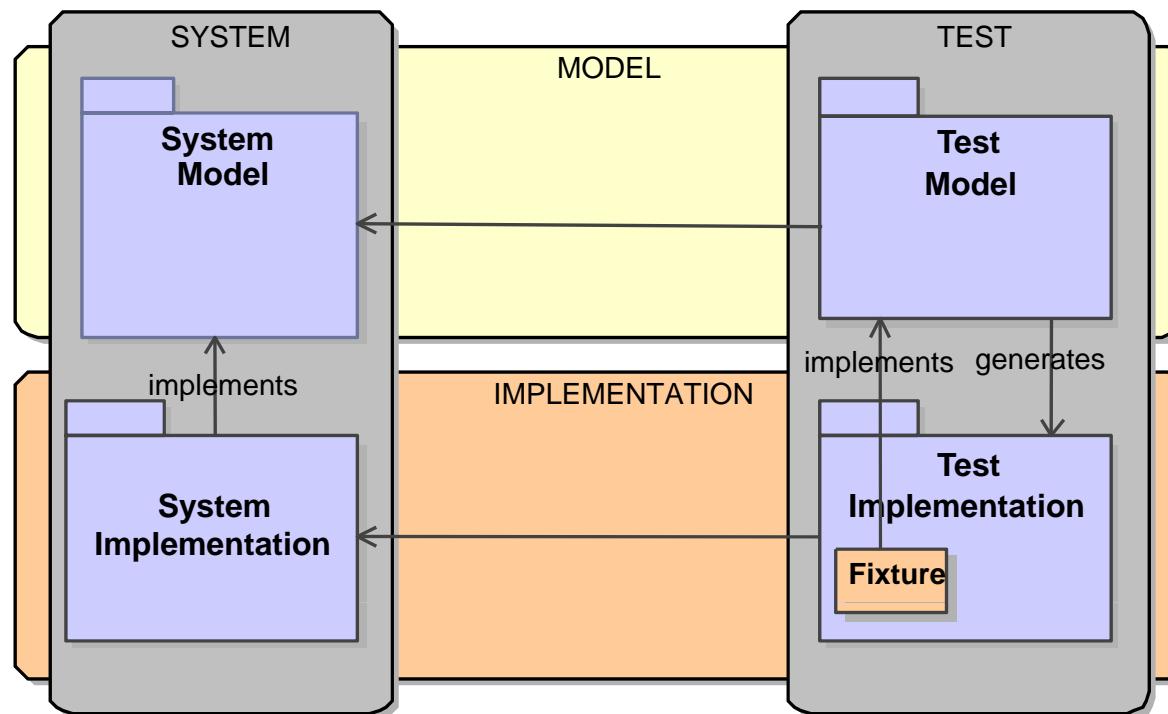


TTS Anforderungen

- **Erstellung eines Testmodells auf Basis der Anforderungen**
- **Ausführbarkeit der Teststories**
 - Teststories sind kontrollierte Abfolgen von Serviceaufrufen auf dem zu testenden System mit Testzusicherungen und Daten aus einer Tabelle
- **Trennung von Teststories und Testdaten**
- **Formales System- und Testmodell**
 - Ermöglicht die automatische Prüfung von Konsistenz- und Überdeckungseigenschaften
- **Testen servicebasierter Systeme**
 - verschiedene Akteure rufen unterschiedliche Services auf
- **Versionierung von Teststories**

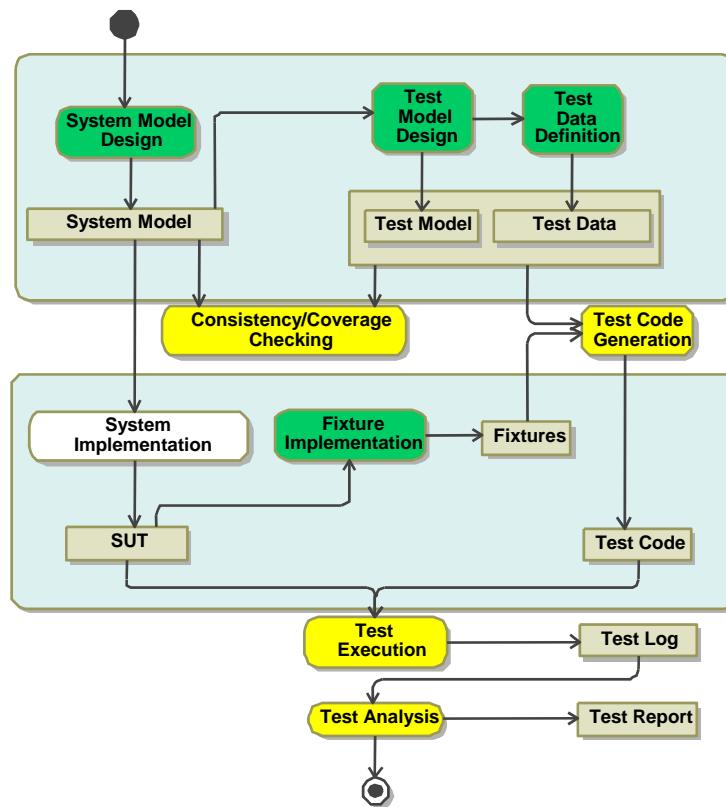
Basiskonzepte

- das Testmodell verwendet Services, Akteure und Klassen des Systemmodells



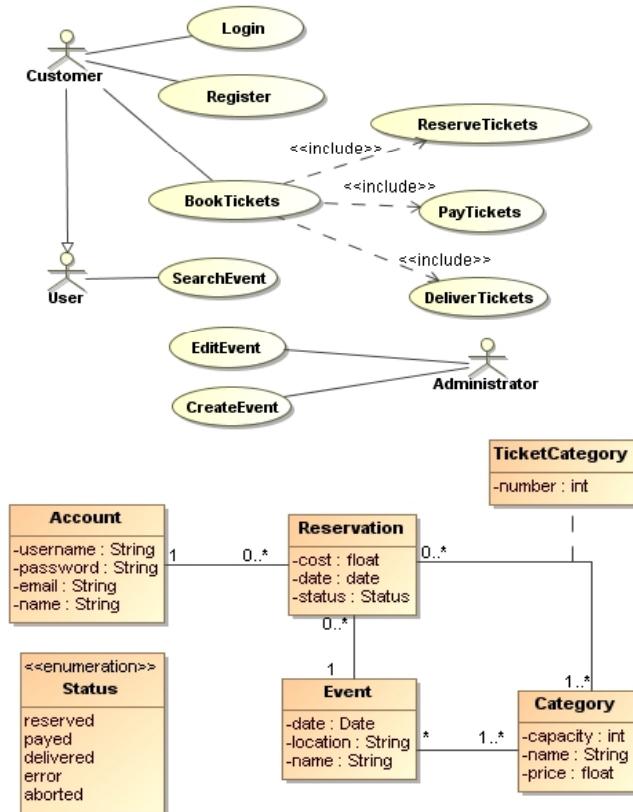
TTS Methode

- Die TTS Methode beinhaltet die Erstellung von System- und Testmodell, Testdaten und Fixtures



Beispiel System Model

- Grundelemente des Systemmodells sind Services, Akteure und Klassen
 - zusätzliche Modelle wie ein Anwendungsworkflow erlauben die Generierung von Teststories



```

system test {

  actor User

  service ReserveTickets {

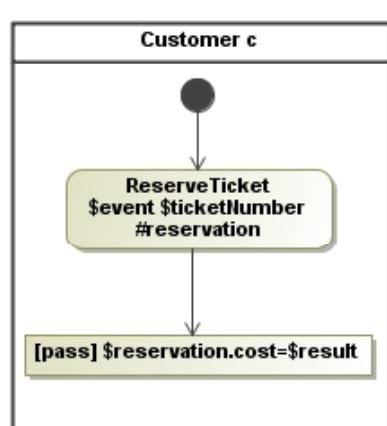
    actors (Customer)
    in (event:Event numPerCategory:Sequence(Integer))
    out reservation:Reservation
    pre "numPerCategory->forall(i|i>=0)"
    post "res.status = reserved"
  }

  class Account {

    attributes ( password:String
                email:String
                username:String
                name:String )
    associations (res:Reservation)
  }
}
  
```

Beispiel Test Model

- Testszenario zur Überprüfung ob die Reservierungskosten stimmen



```

teststory ReserveTickets_Cost {

    actor Customer c
    testtable ReserveTicket_Cost
    initial init_1
    final final_1

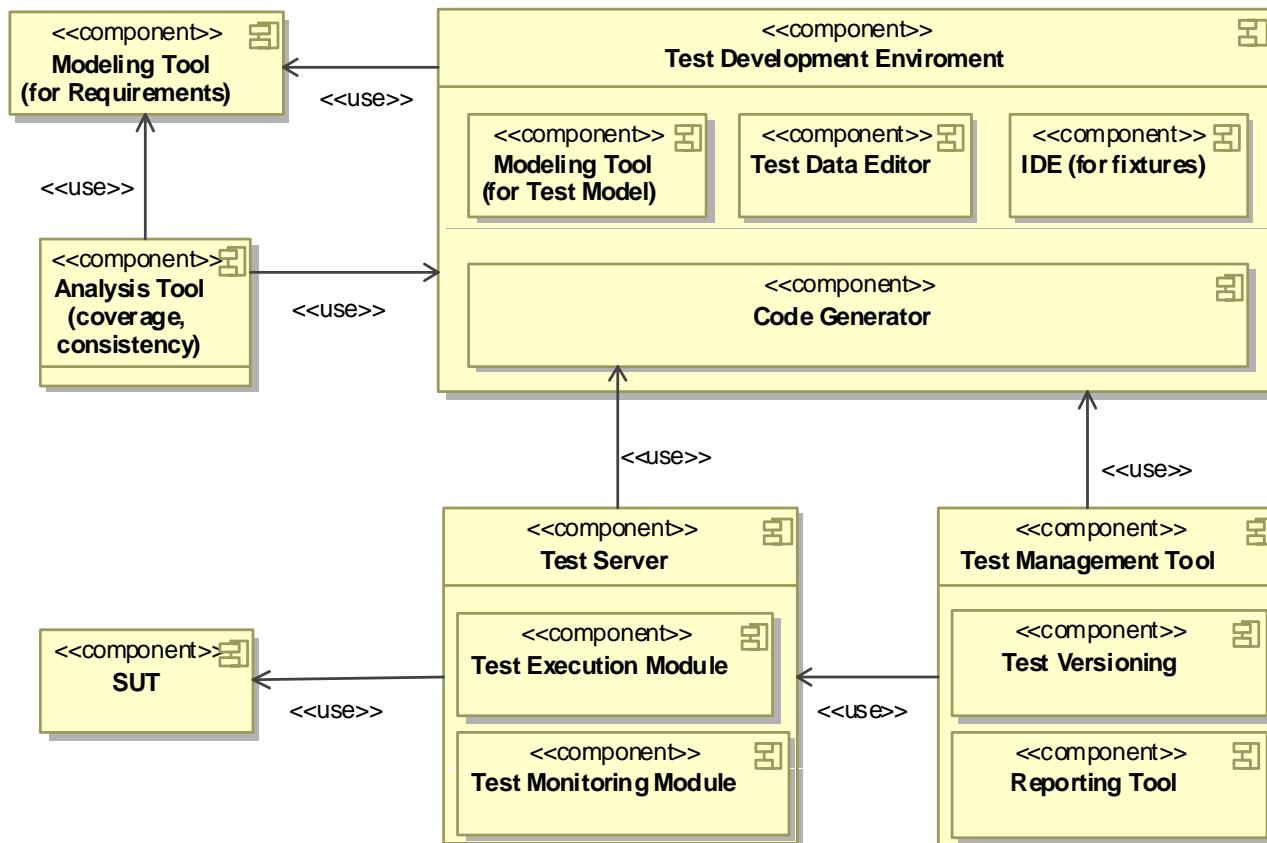
    sequence {
        service c:ReserveTicket ($event $ticketNumber) #reservation
        assertion {
            [pass] #reservation.cost=$result
        }
    }
}
  
```

id	event	ticketNumber	result
t_1	ref:event_1	[1,0,0]	135,00
t_2	ref:event_1	[0,1,1]	225,00
t_3	ref:event_1	[3,0,1]	650,00

id	date	location	name
event_1	08.06.2008 20:45	Vienna Happel Stadium	EM08 AUT - CRO
event_2	08.06.2008 20:45	Klagenfurt Stadium	EM08 GER - POL
event_3	09.06.2008 20:45	Zuerich Stadium	EM08 NED - ITA

Architektur

- Module des TTS Frameworks

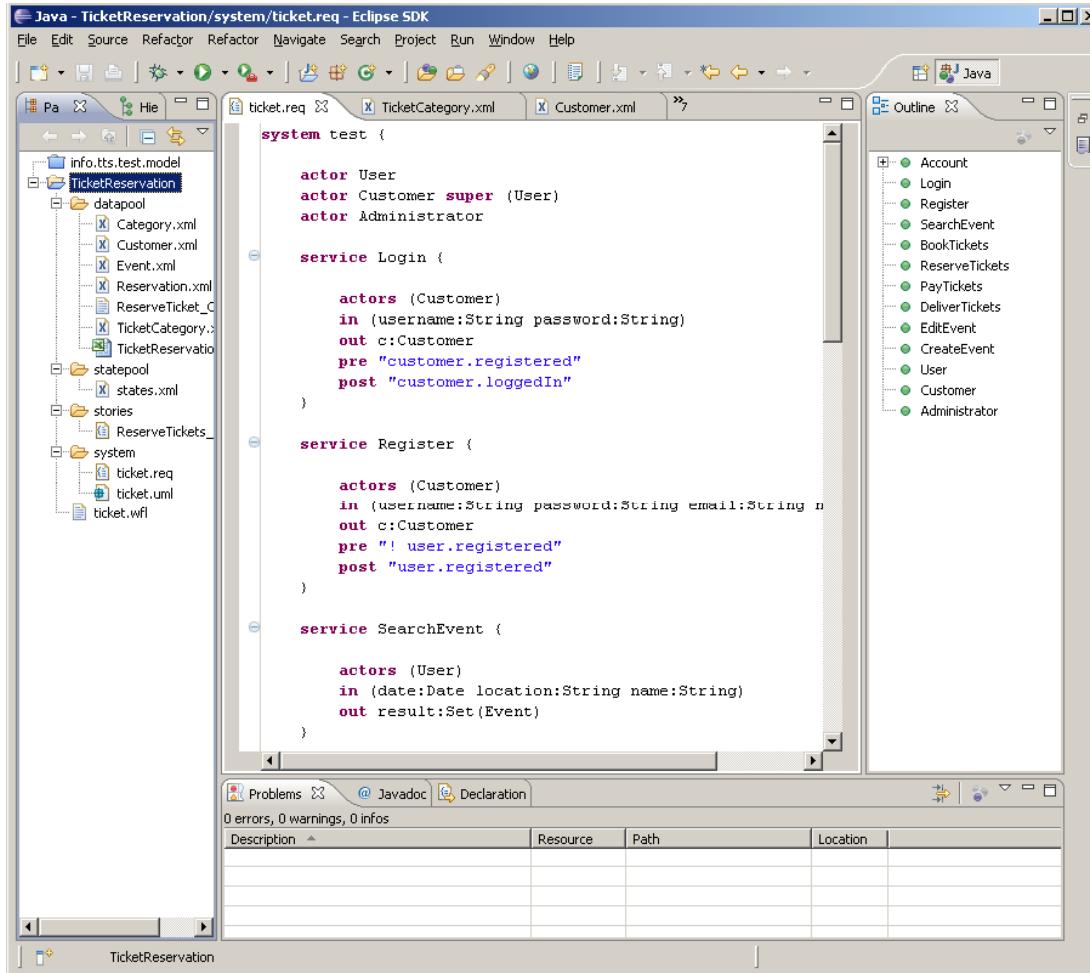


Technologien

- **Eclipse Plugins für die Verwaltung von Testsuiten**
 - Wizard zur Generierung der Teststories
- **EMF Ecore basiertes Metamodell für System- und Testmodell**
 - Formal für Überprüfungen und Testfallgenerierung
- **openArchitectureWare zur die Generierung von Testcode und Editoren sowie der Metamodelle**
- **Workflow Engine SecServ für die Orchestrierung der Services**

Editor Beispiel

- Editor zur Erstellung von Test Suiten



The screenshot shows the Eclipse IDE interface with the following details:

- Title Bar:** Java - TicketReservation/system/ticket.req - Eclipse SDK
- Toolbar:** Standard Eclipse toolbar with icons for file operations, search, and project navigation.
- Left Side:** Project Explorer view showing the project structure under "TicketReservation". It includes "Category.xml", "Customer.xml", "Event.xml", "Reservation.xml", "ReserveTicket_C...", "TicketCategory.xml", "states.xml", "stories", "ReserveTickets", and "system" which contains "ticket.req", "ticket.uml", and "ticket.wfl".
- Middle Area:** The main editor area displays a UML-like test specification for a "TicketReservation" system. The code defines actors (User, Customer, Administrator) and services (Login, Register, SearchEvent). The "Login" service has pre- and post-conditions involving a "Customer" actor. The "Register" service also involves a "Customer" actor and its registration status. The "SearchEvent" service takes parameters like date, location, and name, and returns a set of events.
- Right Side:** Outline view showing a list of available actors: Account, Login, Register, SearchEvent, BookTickets, ReserveTickets, PayTickets, DeliverTickets, EditEvent, CreateEvent, User, Customer, and Administrator.
- Bottom:** Problems view showing 0 errors, 0 warnings, and 0 infos. A table for viewing detailed information about errors, resources, paths, and locations.

Vorteile

- **Modellbasiertes Testen**
 - Erstellung von Tests in früher Projektphase, Codeunabhängigkeit
- **flexibel einsetzbares Systemmodell**
 - Systemmodell sehr allgemeine, damit in vielen Situationen einsetzbar
- **Trennung von System- und Testmodell**
 - Qualität des Testmodells nur bedingt von Qualität des Systemmodells abhängig
- **Trennung von Testlogik und Testdaten**
 - Unterschiedliche Personen können Logik und Daten erstellen
- **Verbindung zwischen Anforderungsspezifikation und Testmodell**
 - Automatische Prüfung der Testqualität

Wissenschaftliche Fragestellungen

- **Automatische Generierung von Test Stories und Testfällen**
- **Konsistenzprüfung zwischen Testmodell und Anforderungsspezifikation**
- **Definition und Implementierung von Überdeckungskriterien der Anforderungsspezifikation durch Test Stories**
- **Testen Serviceorientierter Anwendungen**
- **Testausführung und -auswertung**
 - Definition komplexer Verdiktfunktionen
 - Formalisierung des Zustandsmodells
 - Workflow für Testkampagnen

Fallstudie

- **Testen Verteilter Systemkomponenten in einem Softwaresystem zur bidirektionalen Datenübertragung in Fahrzeugen über eine Telefonanlage**
- **Anwendungs- und Testspezifikation**
- **Testausführung**



Zusammenfassung

- **Modellbasiertes Testen servicebasierter Systeme**
- **Trennung von System- und Testmodell**
- **Metamodellbasierter Ansatz**
- **Trennung von Teststory und Testdaten**