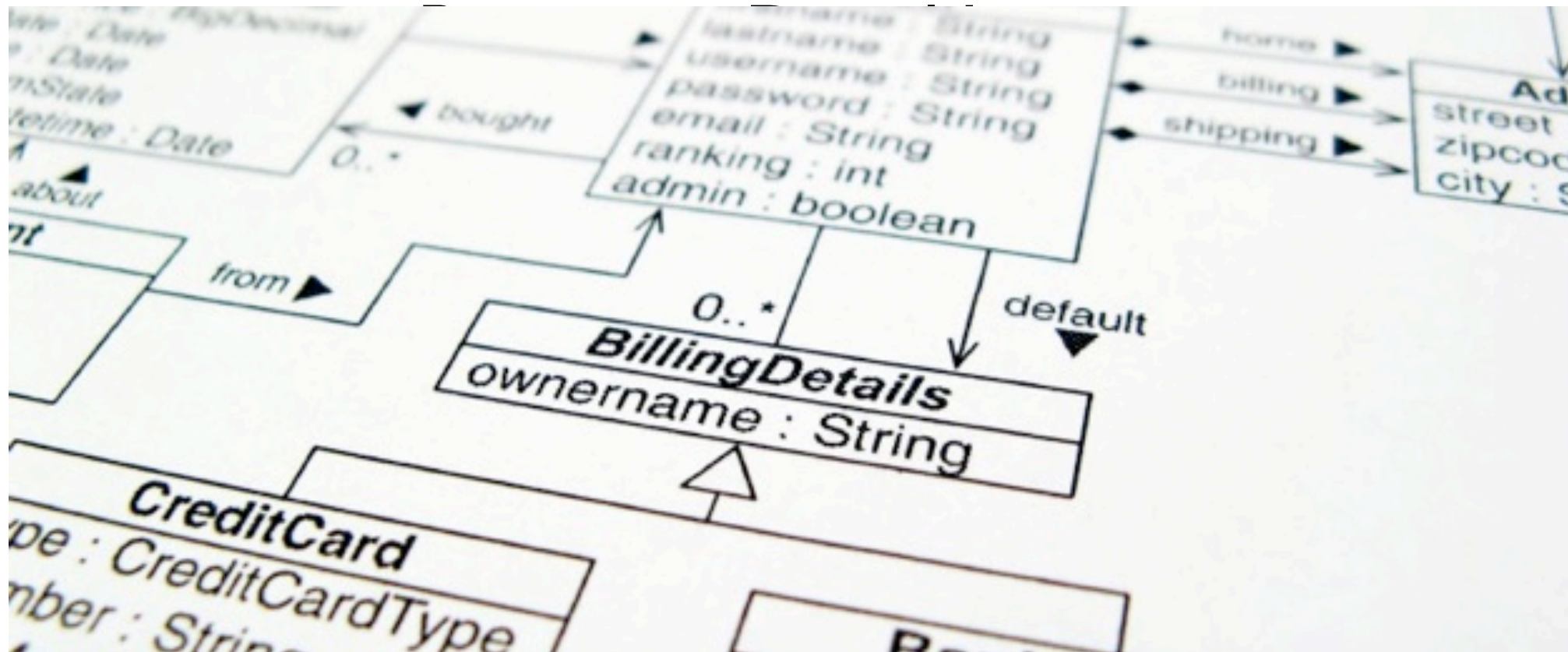


Model based Testing of Embedded Systems



- Einführung & Motivation
- Grundlagen & Testansätze
- Anforderungsanalyse
- themenbezogene Arbeiten
 - Projekte
 - vorhandene Tools
- der eigene MBT Ansatz
- Zusammenfassung und Ausblick



Why Testing?

- Fehler in den Programmen finden
- verifizieren des korrekten Verhaltens der SW
- gegen die Anforderungen Validieren
- mögliche Kapitalschäden verhindern
- Testen als ein begleitender Teil des Entwicklungsprozesses

Model Driven Engineering

- eine abstrakte Sicht haben auf:
 - Struktur
 - Verhalten
 - Anforderungen
 - Daten
- ein besseres Verständnis und Überblick
- als Basis der Kommunikation in Teams

the first tools to support MDE were developed in the 1980's

UML 1997

Model Based Testing (MBT)

- immer komplexere Systeme
- Testen ist teuer
- nutzt die Vorteile von MBE
- automatische TestCase Generierung möglich
- besonders embedded System brauchen ein Werkzeug, um das Verifizieren und Validieren in der frühen Modellierungsphase
- Modellierung als heutiger Standard in der Entwicklung

44% of embedded System reaches only 20 % of functionality and performance expectations

- Einführung & Motivation
- Grundlagen & Testansätze
- Anforderungsanalyse
- themenbezogene Arbeiten
 - Projekte
 - vorhandene Tools
- der eigene MBT Ansatz
- Zusammenfassung und Ausblick



Software Test

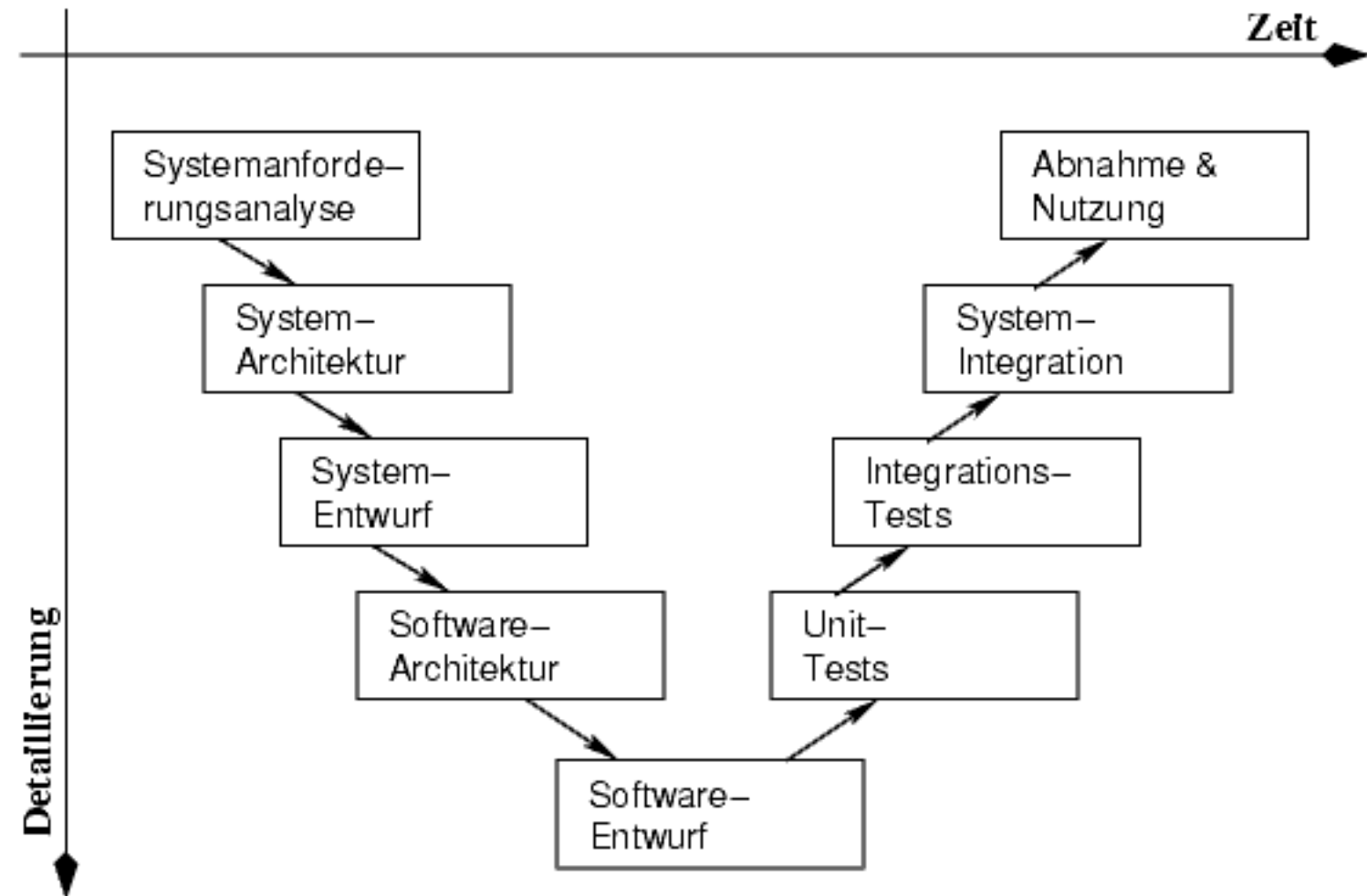
- ISTQB (gegründet 2002) Certified Tester
- Standards: ISO 9126 / DIN 66271
- Softwarequalitäts-Charakteristiken
 - Funktionalität
 - Zuverlässigkeit
 - Handhabbarkeit
 - Effizienz
 - Änderbarkeit
 - Übertragbarkeit

International Software
Testing Quality Board
founded 2002

a lot Terms included in
standards

Software Test

- funktional vs. nicht-funktional
- statisch vs. dynamisch
- Teststufe:
 - Komponenten Tests
 - Integrationstests
 - Systemtests
 - Abnahmetests



Software Test

- Testlauf
- Testfälle
- Testdaten
- Testscenarien

MBT

- automatisches Testen:
 - Testfallmodellierung
 - Testfallgenerierung
 - Testdatengenerierung
 - Testlauf
 - Testanalyse - offline vs. online
 - Testdokumentation

- Einführung & Motivation
- Grundlagen & Testansätze
- Anforderungsanalyse
- themenbezogene Arbeiten
 - Projekte
 - vorhandene Tools
- der eigene MBT Ansatz
- Zusammenfassung und Ausblick



Modelle für MBT

- brauchen geeignete Modelle fürs Testen:
 - Petrinetz
 - Automaten (timed Automata)
 - SDL
 - DSL (bsp. Simulink Modelle)
 - UML

UML

- OMG - Object Management Group
- 1997 wurde UML vervollständigt und von der OMG standardisiert
- vereinigt viele Diagramme unter einem Dach
- abstrakte Modelle
- ISO/IEC 19501
- bei den Softwareentwicklern etabliert



UML Profile

- Erweiterung der UML für bestimmte domänen
- Stereotypen und Constraints erweitern die UML und können Einschränkungen definieren
- UTP - UML Testing Profile

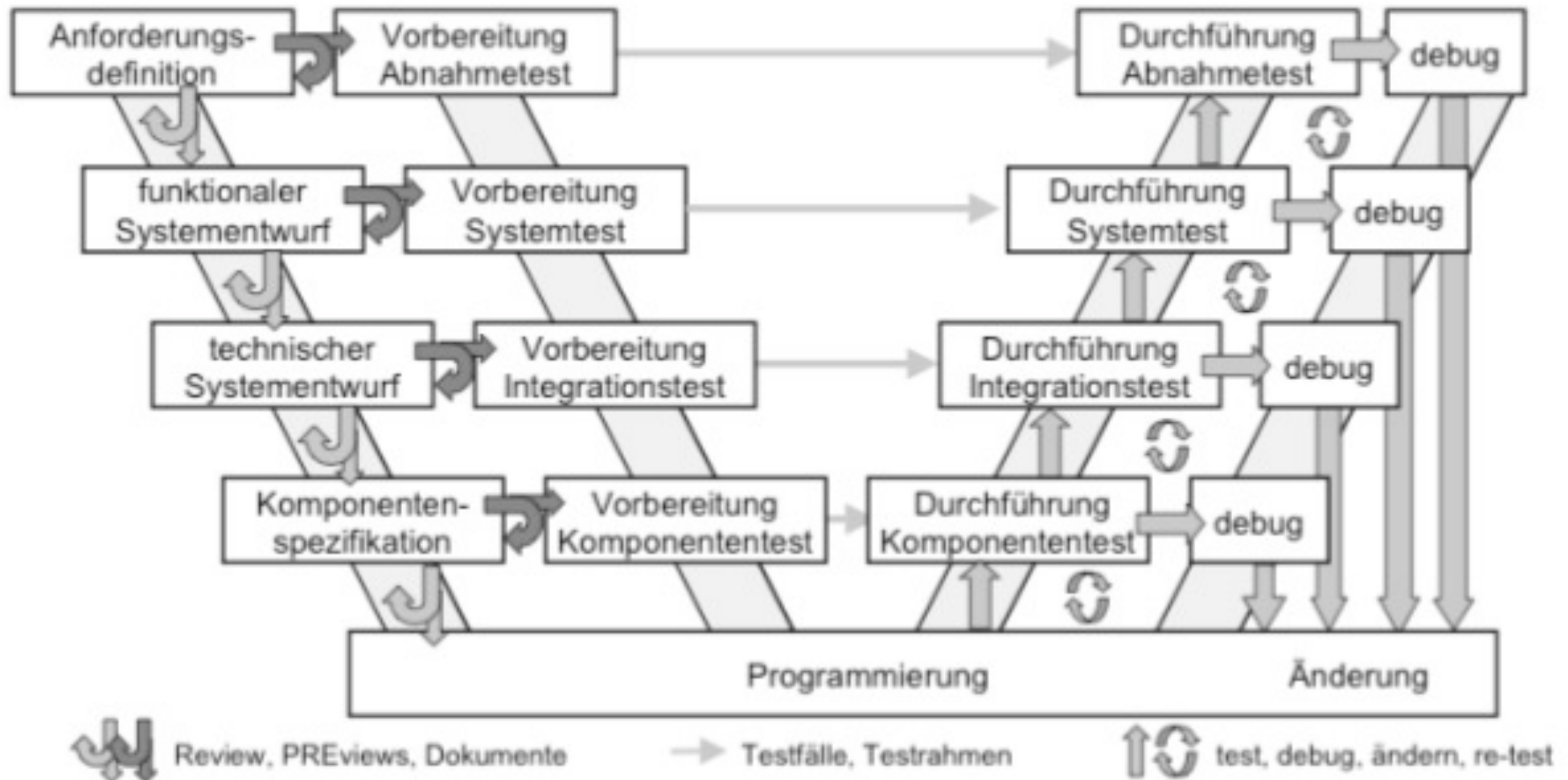
UML Modelle

- Strukturdiagramme vs. Verhaltensdiagramme
- Diagramme können semantisch untereinander verknüpft werden

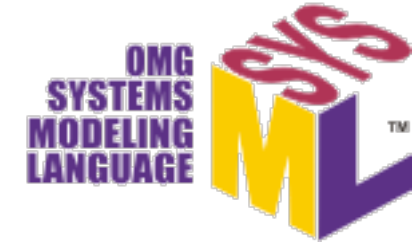
UTP

- Domäne fürs Testen
- Verwendete Diagramme:
 - Sequenzdiagramm
 - Interaktionsdiagramm
 - Zustandsdiagramm
 - UseCase-Diagramm
- wichtigsten Stereotypen:
 - TestComponent
 - SUT
 - DataPool

UTP



SysML




- System Engineering
- 2001 hat INCOSE das Projekt in leben gerufen
- 2007 von der OMG veröffentlicht
- vereinigt eine Teilmenge mit der UML
- Ziele:
 - Systemanforderungen definieren
 - Evaluieren von Anforderungen
 - Systeminformationen kommunizieren

International Council
SE

Diagramme der SysML

- aus der UML geblieben:
 - UseCase-Diagramme
 - Zustandsdiagramme
 - Sequenzdiagramme
 - Packetdiagramme
- erweitert oder neu hinzugekommen:
 - Klassendiagramm --> Blockdefinitionsdiagramm
 - Kompositionsstrukturdiagramm --> Internes Blockdiagramm
 - Zusicherungs- und Anforderungsdiagramm sind neu in SysML

TTCN-3

- ITU und ETSI Standardisiert 
- für Kommunikationssysteme entwickelt
- drei verschiedene Modelle:
 - Core Language
 - Graphical Format
 - Tabular Format
- alle Modelle lassen sich bijektiv aufeinander abbilden

Testing and Test
Control
Notation

- Einführung & Motivation
- Grundlagen & Testansätze
- Anforderungsanalyse
- themenbezogene Arbeiten
 - Projekte
 - vorhandene Tools
- der eigene MBT Ansatz
- Zusammenfassung und Ausblick



Übersicht

- Projekte:

- MARTE (OMG)
- Uni Bremen & Verified System
- AGEDIS
- Erlang-Consulting - Early Fault Detection
- AUTOSAR
- CTE/ES
- Motion

- Tools:

- Papyrus
- TDE/UML
- MATT
- TTworkbench
- IBM Rational Rhapsody

MARTE

- Neufassung von der OMG für das UML SPT-Profil
- entwickelt um folgende Anforderungen modellieren zu können:
 - Schedulability
 - Performance
 - Zeit
- zusätzliche Analysemöglichkeiten



**Modeling and Analysis
Real-time and Embedded
systems**

Uni Bremen & Verified System

- MBT mit UML 2.0 mit den folgenden Diagrammen:
 - Komponentenstrukturdiagramm
 - Klassendiagramme
 - Zustandsdiagramme
- Testlevel: Integrationstest (SW & SW/HW)
- untersuchte Testaspekte: Zeitabhängige Zusammenhänge, Robustness Tests, Transitionsabdeckung
- Ergebnisse: Modelle fürs Testen müssen erzeugt werden

AGEDIS

- Falluntersuchung über 3 Jahre
- von der Europäische Union 2002 in leben gerufen
- France Telecom, Intrasoftware, IBM
- Ergebnisse:
 - besseres Verständnis des SUT, dadurch einen effektiven Weg zur Analyse komplexer Anforderungen
 - weniger Aufwand bei Änderungen von Anforderungen
 - die Qualität der Tests war sehr gering bei Verwendung der Modelle aus denen der Programmcode generiert wurde

Early Fault Detection

- Erlang Consulting limited
- Faults-Slip-Through
Phase Input Quality (PIQ) = $SF/PF * 100$
- Ergebnisse:
 - MBT findet nicht zwangsweise alle Fehler, somit haben handgeschriebenen Tests ihre Daseinsberechtigung
 - Fehler werden in verschiedenen Testleveln gefunden und nicht immer richtig eingestuft.

- **AUT**omotive **O**pen **S**ystem **AR**chitecture
- offener Standard für Elektronik-Architekturen
- EAST-ADL (Electronics Architecture and Software Technologies - Architecture Description Language) von der ITEA - ein Profil der UML 2.0
- Ist auch in den Modellen von Simulink und Stateflow integriert
- AUTOSAR UML Profil (in Zusammenarbeit mit IBM)

CTE/ES

- **C**lassification **T**ree Method / for **E**MBEDDED **S**ystems
- systematisches Testdesign für eingebettete Systeme
- Äquivalenzklassenbildung
- Testlevel: Komponententest, Integrationstests
- nur funktionale Tests
- Ergebnis: lässt sich in verschiedenen Testfallgeneratoren einbauen

Motion

- Projekt für die Entwicklung und Forschung im Bereich MDE
- Fraunhofer Institut Berlin
- MBT auf verschiedenen Plattformen
- Testlevel: Komponenten-, Integrations-, System- und Abnahmetests
- untersuchen funktionale und nicht-funktionale Test
- Ergebnis: Werkzeuge für die Testspezifizierungssprache TTCN-3



Übersicht

- Projekte:

- MARTE (OMG)
- Uni Bremen & Verified System
- AGEDIS
- Erlang-Consulting - Early Fault Detection
- AUTOSAR
- CTE/ES
- Motion

- Tools:

- Papyrus
- TDE/UML
- MATT
- TTworkbench
- IBM Rational Rhapsody

Papyrus

- open source UML 2.0 Design Tool
- Eclipse Plugin
- beherrscht:
 - UML 2.0 OCL
 - MARTE
 - SysML
 - EAST-ADL
- besitzt einen Code Generator
- einfaches Hinzufügen von neuen Diagrammen oder Code Generatoren

TDE/UML

- reduziertes UML 2.0 Design Tool mit integrierten Testgenerator
- entwickelt von Siemens Corporate Research (SCR)
- Testlevel: Komponenten-, Integrations-, System- und Abnahmetests
- generiert Tests für die Programmiersprache Java (JUnit Tests)

MATT

- **MATLAB Automated Testing Tool**



- automatisches Test Tool für Black Box (funktionale) Tests aus der DSL Sprache SIMULINK Modellen
- gibt die Möglichkeit benutzerdefinierte Tests zu entwickeln
- erlaubt eine offline Evaluierung
- Testaspekt: Datenüberdeckung

TTworkbench

- Tool zum Definieren und Generieren von Tests mit TTCN-3
- das Tool ist größtenteils vom Fraunhofer Institut entwickelt worden und wird immer weiter entwickelt
- Testlevel: Komponenten-, Integrations-, System- und Abnahmetests
- deckt sowohl funktionale als auch nicht-funktionale Tests ab

IBM Rational Rhapsody

- UML 2.0 Design, CodeGenerierungs- und Testfallgenerator Tool
- Eclipse Plugin
- Testlevel: Komponenten-, Integrations-, System- und Abnahmetests
- beherrscht:
 - UML 2.0 OCL
 - SysML
 - AUTOSAR

- Einführung & Motivation
- Grundlagen & Testansätze
- Anforderungsanalyse
- themenbezogene Arbeiten
 - Projekte
 - vorhandene Tools
- der eigene MBT Ansatz
- Zusammenfassung und Ausblick



was soll MBT lösen

- Versuch MBT für nicht-funktionale Tests nutzbar zu machen
- speziell: Performance- und Robusttests

welches Modell?

- SysML vereinigt mit MARTE (und AUTOSAR)
- Untersuchungen über das Vertragen der beiden Sprachen stehen aus!
- Nutzen der UML 2.0
- open source Lösungen nutzen - Papyrus

- Einführung & Motivation
- Grundlagen & Testansätze
- Anforderungsanalyse
- themenbezogene Arbeiten
 - Projekte
 - vorhandene Tools
- der eigene MBT Ansatz
- Zusammenfassung und Ausblick



- MBT wuchs von verschiedenen Disziplinen raus
- MBT hilft beim Testen, ist aber nicht so intuitiv wie der Mensch und erzeugt auch Arbeitsaufwand
- wird schon erfolgreich eingesetzt, aber etliche Untersuchungen stehen aus
- Trend geht weg von DSL zu abstrakten Modellierungsmethoden
- gerade im Embedded Bereich noch sehr viele verschiedene Ansätze -> schwer durchschaubar

- mit SysML und MARTE lassen sich Tests für embedded Systeme modellieren, doch Tools, die das tun, stehen noch aus
- MBT auf abstrakten Modellen wurde bisher in verschiedenen Ansätzen geprüft, aber bisher nicht standardisiert