Modellbasiertes Testen mit UTP

Daniel Löffelholz

16. Dezember 2008

Inhaltsverzeichnis

Motivation

- Steigende Komplexität von Softwaresystemen

Motivation

- Steigende Komplexität von Softwaresystemen
- Durch steigende Allgegenwärtigkeit (*Ubiquitous Computing*) auch steigender Anspruch an die Qualität

Motivation

- Steigende Komplexität von Softwaresystemen
- Durch steigende Allgegenwärtigkeit (*Ubiquitous Computing*) auch steigender Anspruch an die Qualität
- Neue Paradigmen gefordert!

Motivation

- Steigende Komplexität von Softwaresystemen
- Durch steigende Allgegenwärtigkeit (*Ubiquitous Computing*) auch steigender Anspruch an die Qualität
- Neue Paradigmen gefordert!
- Ingenieursmässiges Vorgehen: Abstraktion/Verwenden von Modellen

Motivation

- Steigende Komplexität von Softwaresystemen
- Durch steigende Allgegenwärtigkeit (*Ubiquitous Computing*) auch steigender Anspruch an die Qualität
- Neue Paradigmen gefordert!
- Ingenieursmässiges Vorgehen: Abstraktion/Verwenden von Modellen
- Hält mit Modellgetriebener Softwareentwicklung auch in der Softwareindustrie Einzug

Motivation

- Steigende Komplexität von Softwaresystemen
- Durch steigende Allgegenwärtigkeit (*Ubiquitous Computing*) auch steigender Anspruch an die Qualität
- Neue Paradigmen gefordert!
- Ingenieursmässiges Vorgehen: Abstraktion/Verwenden von Modellen
- Hält mit Modellgetriebener Softwareentwicklung auch in der Softwareindustrie Einzug
- Aber: Änderungen nicht nur bei der Entwicklung, sondern auch beim Testen nötig

Inhalt

Verschiedenste Techniken zur Nutzung von Modellartefakten im Testprozess

Modellbasiertes Testen

- Erste Ansätze mit endlichen Automaten in den 60ern

Modellbasiertes Testen

- Erste Ansätze mit endlichen Automaten in den 60ern
- Bisher nicht durchgesetzt

Modellbasiertes Testen

- Erste Ansätze mit endlichen Automaten in den 60ern
- Bisher nicht durchgesetzt
- Grundvoraussetzung erst seit kurzer Zeit erfüllt

Modellbasiertes Testen

- Erste Ansätze mit endlichen Automaten in den 60ern
- Bisher nicht durchgesetzt
- Grundvoraussetzung erst seit kurzer Zeit erfüllt
- Schaffung einer einheitlichen Modellierungssprache (UML)

Modellbasiertes Testen

- Erste Ansätze mit endlichen Automaten in den 60ern
- Bisher nicht durchgesetzt
- Grundvoraussetzung erst seit kurzer Zeit erfüllt
- Schaffung einer einheitlichen Modellierungssprache (UML)
- Propagierung eines modellbasierten Entwicklungsansatzes (MDA)

Modellbasiertes Testen

- Erste Ansätze mit endlichen Automaten in den 60ern
- Bisher nicht durchgesetzt
- Grundvoraussetzung erst seit kurzer Zeit erfüllt
- Schaffung einer einheitlichen Modellierungssprache (UML)
- Propagierung eines modellbasierten Entwicklungsansatzes (MDA)
- Enthält kein explizites Testvorgehen

Modellbasiertes Testen

- Erste Ansätze mit endlichen Automaten in den 60ern
- Bisher nicht durchgesetzt
- Grundvoraussetzung erst seit kurzer Zeit erfüllt
- Schaffung einer einheitlichen Modellierungssprache (UML)
- Propagierung eines modellbasierten Entwicklungsansatzes (MDA)
- Enthält kein explizites Testvorgehen
- MDT benötigt seperate Modellierungstechnologien und Testprozessschritte

Modellbasiertes Testen

- Erste Ansätze mit endlichen Automaten in den 60ern
- Bisher nicht durchgesetzt
- Grundvoraussetzung erst seit kurzer Zeit erfüllt
- Schaffung einer einheitlichen Modellierungssprache (UML)
- Propagierung eines modellbasierten Entwicklungsansatzes (MDA)
- Enthält kein explizites Testvorgehen
- MDT benötigt seperate Modellierungstechnologien und Testprozessschritte

Vor- und Nachteile

- Eigenschaften der MDA werden übernommen

Vor- und Nachteile

- Eigenschaften der MDA werden übernommen
- (+) Automatisierungsmöglichkeiten

Vor- und Nachteile

- Eigenschaften der MDA werden übernommen
- (+) Automatisierungsmöglichkeiten
- (+) Wiederverwendung

Vor- und Nachteile

- Eigenschaften der MDA werden übernommen
- (+) Automatisierungsmöglichkeiten
- (+) Wiederverwendung
- (+) Abstraktion

Vor- und Nachteile

- Eigenschaften der MDA werden übernommen
- (+) Automatisierungsmöglichkeiten
- (+) Wiederverwendung
- (+) Abstraktion
- (-) Modellierungsoverhead

Vor- und Nachteile

- Eigenschaften der MDA werden übernommen
- (+) Automatisierungsmöglichkeiten
- (+) Wiederverwendung
- (+) Abstraktion
- (-) Modellierungsoverhead
- (-) Nicht für jede Anwendung geeignet

Modellbasiertes Testen II

Zwei Möglichkeiten:

Modellbasiertes Testen II

Zwei Möglichkeiten:

- Testen von Modellen mit Modellen

Modellbasiertes Testen II

Zwei Möglichkeiten:

- Testen von Modellen mit Modellen
- Generieren vom Testsystem aus einem Testmodell

Modellbasiertes Testen II

Zwei Möglichkeiten:

- Testen von Modellen mit Modellen
- Generieren vom Testsystem aus einem Testmodell

Statisches Testen

Testen ohne Ausführung der Modelle

- Syntax: Korrektheit im Sinne der Modellierungsvorschrift - Meta-Modelle

Statisches Testen

Testen ohne Ausführung der Modelle

- Syntax: Korrektheit im Sinne der Modellierungsvorschrift - Meta-Modelle
- Stil: Formal kaum möglich - Modellierungsstandards

Statisches Testen

Testen ohne Ausführung der Modelle

- Syntax: Korrektheit im Sinne der Modellierungsvorschrift - Meta-Modelle
- Stil: Formal kaum möglich - Modellierungsstandards
- Semantik
  - Konsistenz: Aufdecken von wiedersprüchlichen Aussagen der Teilmodelle

Statisches Testen

Testen ohne Ausführung der Modelle

- Syntax: Korrektheit im Sinne der Modellierungsvorschrift - Meta-Modelle
- Stil: Formal kaum möglich - Modellierungsstandards
- Semantik
  - Konsistenz: Aufdecken von wiedersprüchlichen Aussagen der Teilmodelle
  - Vollständigkeit: Umsetzung der Anforderungen in das Modell

Dynamisches Testen

Testen mit Ausführung der Modelle

Dynamisches Testen

Testen mit Ausführung der Modelle

- Aktiv: Prüfung des Systemverhaltens mit definierten Testfällen

Dynamisches Testen

Testen mit Ausführung der Modelle

- Aktiv: Prüfung des Systemverhaltens mit definierten Testfällen
- Passiv: Prüfung ob Systeminvarianten und Zustandsbedingungen eingehalten werden

Dynamisches Testen

Testen mit Ausführung der Modelle

- Aktiv: Prüfung des Systemverhaltens mit definierten Testfällen
- Passiv: Prüfung ob Systeminvarianten und Zustandsbedingungen eingehalten werden

Von den Anforderungen zum Code

- Abstraktionsstufen passend zum “MDA Guide” der OMG

Von den Anforderungen zum Code

- Abstraktionsstufen passend zum “MDA Guide” der OMG
- Modelltransformationen bis zur Testcodegenerierung

Von den Anforderungen zum Code

- Abstraktionsstufen passend zum “MDA Guide” der OMG
- Modelltransformationen bis zur Testcodegenerierung
- Zwischen den Modellen: Generierung von Testmodellen und Anreicherung mit Testkonzepten

System- und Testmodelle

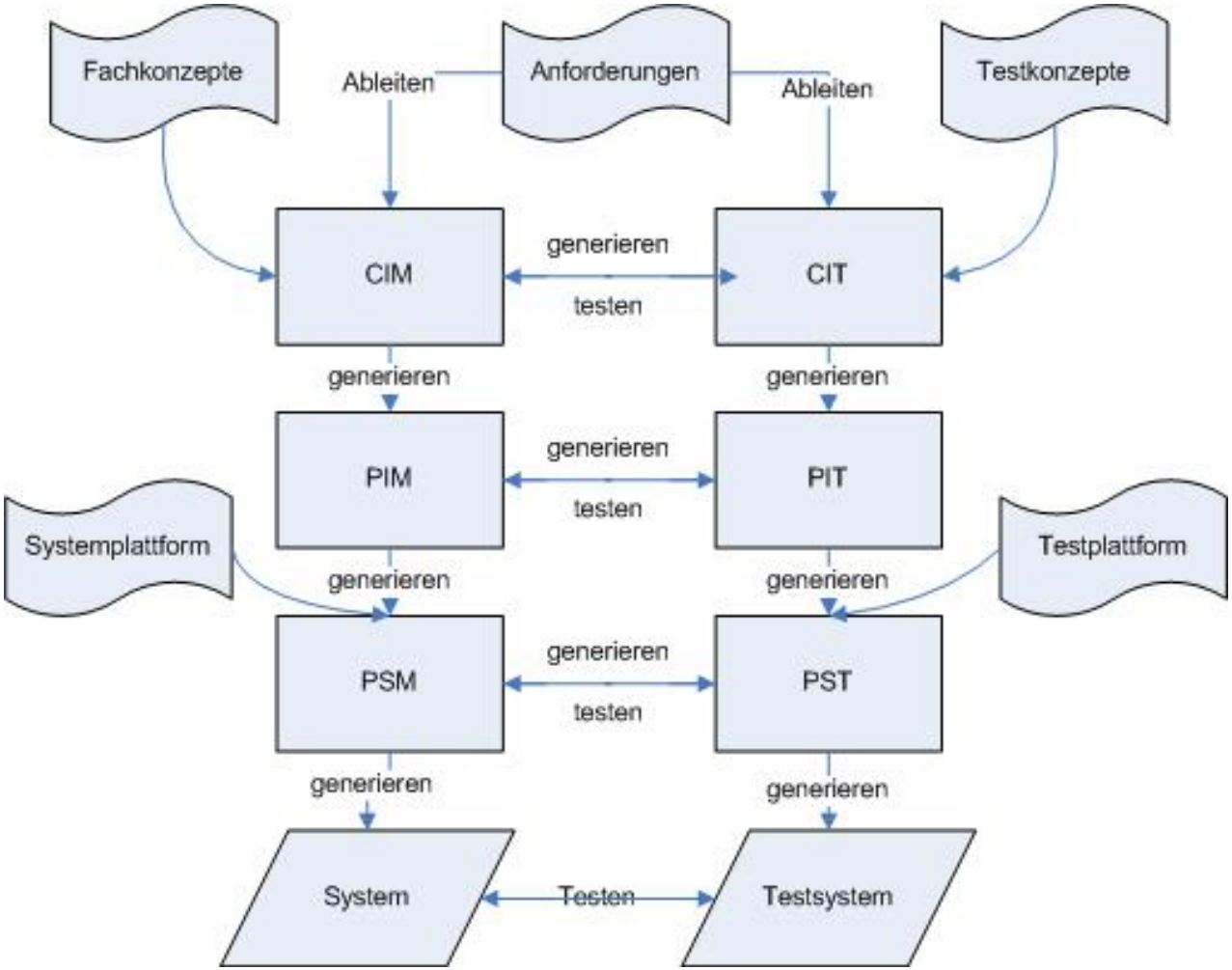

Abbildung: Rollen der Modelle

Inhalt

UML Testing Profile

- MOF-basiertes Metamodell für UML-Testing

UML Testing Profile

- MOF-basiertes Metamodell für UML-Testing
- Ging aus einem Request for Papers von 2001 hervor

UML Testing Profile

- MOF-basiertes Metamodell für UML-Testing
- Ging aus einem Request for Papers von 2001 hervor
- Ermöglicht UML-konforme Spezifikation von statischen und dynamischen Tests

UML Testing Profile

- MOF-basiertes Metamodell für UML-Testing
- Ging aus einem Request for Papers von 2001 hervor
- Ermöglicht UML-konforme Spezifikation von statischen und dynamischen Tests
- Ist mit existierenden Blackboxtest-Technologien verknüpfbar (z.B. JUnit, TTCN-3)

UML Testing Profile

- MOF-basiertes Metamodell für UML-Testing
- Ging aus einem Request for Papers von 2001 hervor
- Ermöglicht UML-konforme Spezifikation von statischen und dynamischen Tests
- Ist mit existierenden Blackboxtest-Technologien verknüpfbar (z.B. JUnit, TTCN-3)

Vorteile von Profilen

- Modelle und Profile können in Standard UML Tools ausgetauscht werden

Vorteile von Profilen

- Modelle und Profile können in Standard UML Tools ausgetauscht werden
- Tools müssen keine Funktionalität zur Veränderung des Metamodells (UML Sprachkern) anbieten

Vorteile von Profilen

- Modelle und Profile können in Standard UML Tools ausgetauscht werden
- Tools müssen keine Funktionalität zur Veränderung des Metamodells (UML Sprachkern) anbieten
- Viele UML-Basiskonzepte bleiben unangetastet und müssen nicht neu erlernt werden

Vorteile von Profilen

- Modelle und Profile können in Standard UML Tools ausgetauscht werden
- Tools müssen keine Funktionalität zur Veränderung des Metamodells (UML Sprachkern) anbieten
- Viele UML-Basiskonzepte bleiben unangetastet und müssen nicht neu erlernt werden

Beispiel: Mobile Gaming System

UTP ist in vier Konzeptgruppen unterteilt:

Beispiel: Mobile Gaming System

UTP ist in vier Konzeptgruppen unterteilt:

- *Testarchitekturkonzepte*: Konzepte bezogen auf Teststruktur- und Testkonfiguration

Beispiel: Mobile Gaming System

UTP ist in vier Konzeptgruppen unterteilt:

- *Testarchitekturkonzepte*: Konzepte bezogen auf Teststruktur- und Testkonfiguration
- *Testverhaltenkonzepte*: Testfälle, Stimuli, Logging,....

Beispiel: Mobile Gaming System

UTP ist in vier Konzeptgruppen unterteilt:

- *Testarchitekturkonzepte*: Konzepte bezogen auf Teststruktur- und Testkonfiguration
- *Testverhaltenkonzepte*: Testfälle, Stimuli, Logging,....
- *Testdatenkonzepte*: Modellierung von Testdaten

Beispiel: Mobile Gaming System

UTP ist in vier Konzeptgruppen unterteilt:

- *Testarchitekturkonzepte*: Konzepte bezogen auf Teststruktur- und Testkonfiguration
- *Testverhaltenkonzepte*: Testfälle, Stimuli, Logging,....
- *Testdatenkonzepte*: Modellierung von Testdaten
- *Testzeitkonzepte*: Timeouts, Bearbeitungsdauer, ...

Testpackage

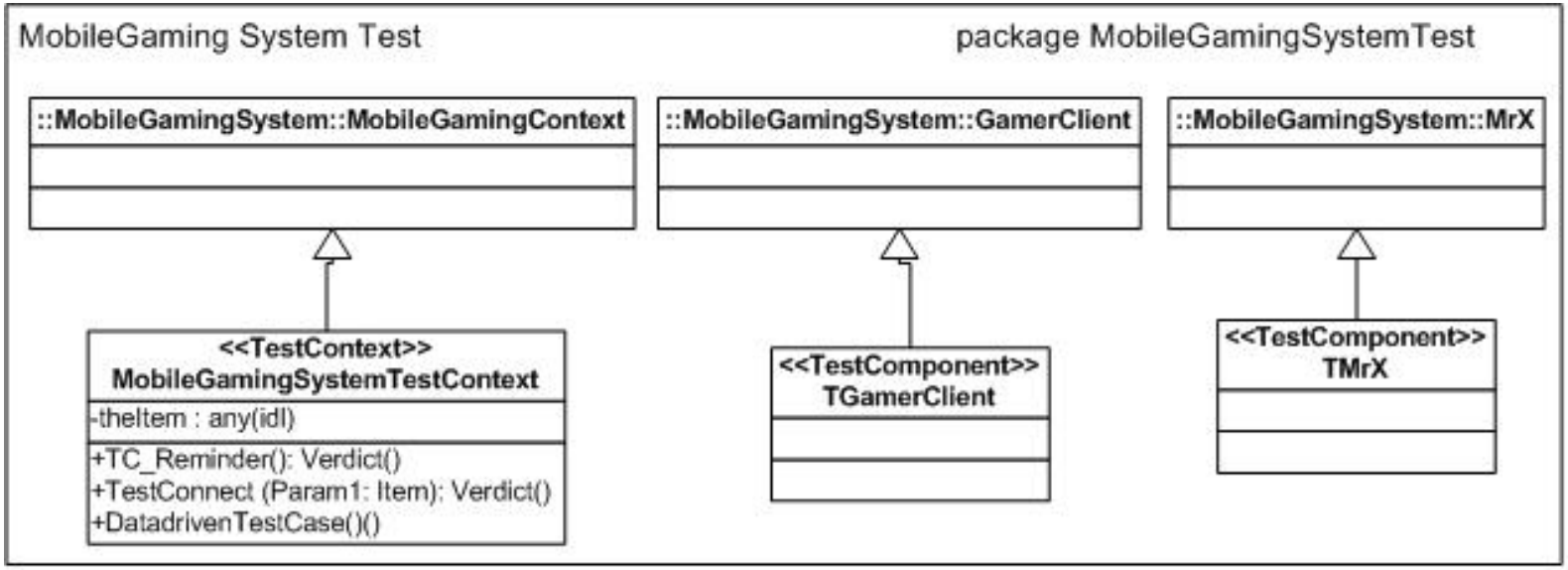

Abbildung: Test Package

Testfall

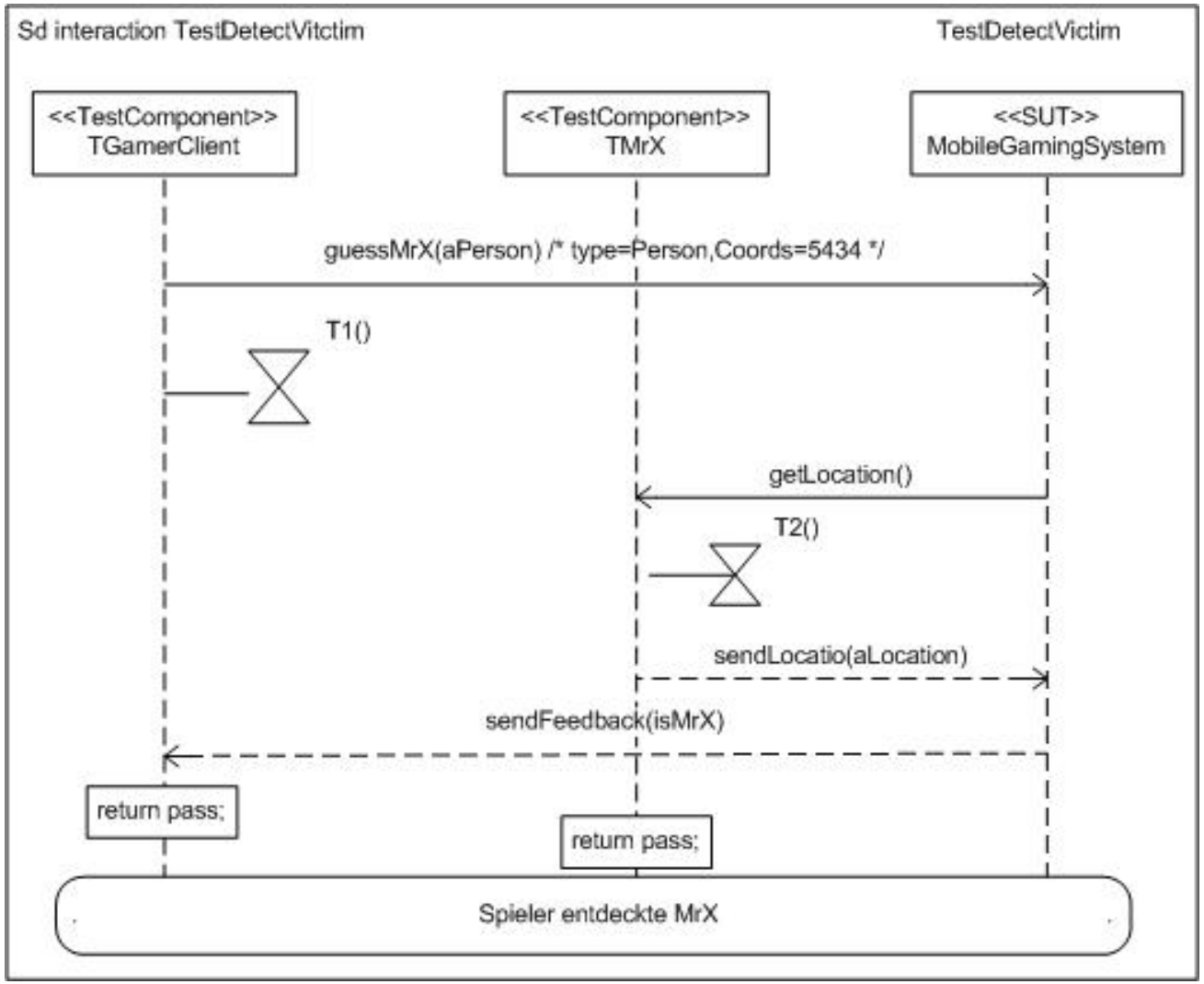

Abbildung: Beispiel Aktivitätsdiagrammtest

Inhalt

Anwendungsbeispiel

- Erstellen eines Testmodells für das Framework-PIM für das Pervasive-Gaming-Projekt

Anwendungsbeispiel

- Erstellen eines Testmodells für das Framework-PIM für das Pervasive-Gaming-Projekt
- Umsetzung des Gelernten

Anwendungsbeispiel

- Erstellen eines Testmodells für das Framework-PIM für das Pervasive-Gaming-Projekt
- Umsetzung des Gelernten
- Parallele Entwicklung zum MDA-Ansatz der OMG

Anwendungsbeispiel

- Erstellen eines Testmodells für das Framework-PIM für das Pervasive-Gaming-Projekt
- Umsetzung des Gelernten
- Parallele Entwicklung zum MDA-Ansatz der OMG
- Für die konkrete Anwendung: Generierung des Testcodes

Anwendungsbeispiel

- Erstellen eines Testmodells für das Framework-PIM für das Pervasive-Gaming-Projekt
- Umsetzung des Gelernten
- Parallele Entwicklung zum MDA-Ansatz der OMG
- Für die konkrete Anwendung: Generierung des Testcodes
- Fragestellungen:
  - Was leisten Werkzeuge mit MDT-Unterstützung aktuell?

Anwendungsbeispiel

- Erstellen eines Testmodells für das Framework-PIM für das Pervasive-Gaming-Projekt
- Umsetzung des Gelernten
- Parallele Entwicklung zum MDA-Ansatz der OMG
- Für die konkrete Anwendung: Generierung des Testcodes
- Fragestellungen:
  - Was leisten Werkzeuge mit MDT-Unterstützung aktuell?
  - Anwendungsfallspezifisch: Entwicklung eines Frameworktestmodells

Anwendungsbeispiel

- Erstellen eines Testmodells für das Framework-PIM für das Pervasive-Gaming-Projekt
- Umsetzung des Gelernten
- Parallele Entwicklung zum MDA-Ansatz der OMG
- Für die konkrete Anwendung: Generierung des Testcodes
- Fragestellungen:
  - Was leisten Werkzeuge mit MDT-Unterstützung aktuell?
  - Anwendungsfallspezifisch: Entwicklung eines Frameworktestmodells
  - Design-for-Testability des Systemmodells

Anwendungsbeispiel

- Erstellen eines Testmodells für das Framework-PIM für das Pervasive-Gaming-Projekt
- Umsetzung des Gelernten
- Parallele Entwicklung zum MDA-Ansatz der OMG
- Für die konkrete Anwendung: Generierung des Testcodes
- Fragestellungen:
  - Was leisten Werkzeuge mit MDT-Unterstützung aktuell?
  - Anwendungsfallspezifisch: Entwicklung eines Frameworktestmodells
  - Design-for-Testability des Systemmodells

Risiken

- Mangelnde Softwareunterstüzung

Risiken

- Mangelnde Softwareunterstüzung
- Abhängigkeit vom zu testenden Modell

Risiken

- Mangelnde Softwareunterstüzung
- Abhängigkeit vom zu testenden Modell

Weiterführende Fragestellungen

- Ausführbarkeit von Testmodellen: xUML und UTP
- Testen von Verhaltensaspekten an Systemmodellen

Fragen

Danke für die Aufmerksamkeit!