ENHANCING TEST MODELS BY INCORPORATING MONITORED USAGE INFORMATION

Presented by Steffen Herbold
Agenda

• Data in Test Models

• Generating Data from Usage

• Proof-of-concept: From SOAP to UML

• Conclusion
DATA IN TEST MODELS
Data in Test Models

- Different Models used in Model-Based Testing (MBT)
- Focus on behavior

And the Test Data?
Simple for Literals

• Example
  • `purchaseOrder(int amount)`

• Test logic can determine values
  • Boundary-value analysis, constraint solving, symbolic execution

• For testers also possible to set values manually
Difficult for Complex Types

• Example
  • `purchaseOrder(purchaseOrderType order)`
  • `purchaseOrderType` itself can have complex types as attributes

• Deep nesting possible
• Hard to determine good values automatically
• Hard for testers to set values manually

Automated Support Required!
TEST DATA FROM USAGE
Solution: Usage-based Testing

• Collect usage data:
  • Observe users and collect data about the SUTs usage
    • Timestamps
    • Called operations
    • Transferred data
    • ...

• Create usage profile and generate tests
  • Create stochastic model of usage
  • Walks through the model for test definition

Foundation for test data
Precursors

• For regressions
  • Or at least partial implementations

• Test model must be modifiable
  • Possibly difficult with proprietary models

• Monitored data can be mapped to test model
PROOF-OF-CONCEPT

From SOAP to UML
Our Scenario

• Testing of SOA Applications

• Monitored data = SOAP calls
  • XML!

• Test model: MIDAS DSL
  • UML and UML Testing Profile (UTP) based Language
Data in the MIDAS DSL

• MIDAS DSL is a UML/UTP based modeling language

• UML Data Types and Literals for the structural description of test data

• UML Instance Specifications for definition of values for tests
Data in the MIDAS DSL

- **Structural Data Definitions**
  - `dataType` `purchaseOrder`
    - `purchaseOrderXML : orderMessageResponseType`

- **Concrete Instances in Tests**
  - `purchaseOrder_Instance : purchaseOrder`
    - `purchaseOrderXML = purchaseOrderXML_Instance`

- **UML Data Types and Literals**
  - `dataType` `orderMessageResponseType`
    - `idOrder : String`
    - `idProduct : String`
    - `quantity : Integer`

- **UML Instance Specifications**
  - `purchaseOrderXML_Instance : orderMessageResponseType`
    - `idOrder = "order-18gog"`
    - `idProduct = "POTATOES"`
    - `quantity = 5000`
Problems with the MIDAS Pilots

- SOAP data often deeply nested
- Sometimes over 1600 instance specifications for a single message!
  - Cannot be generated manually
- Lucky coincidence: usage-based testing part of MIDAS
Practical Considerations

• Usage data in XML logs
• Test models in UML
  • No direct connection!

• Solution: string matching
  • Names of XML tags / attributes must match data type attributes in UML

• Error prone if test model is created manually
  • Automated support for test model creation helpful
Matching SOAP to UML

```xml
<purchaseOrderXML>
  <idOrder>order-18gog</idOrder>
  <idProduct>POTATOES</idProduct>
  <quantity>5000</quantity>
</purchaseOrderXML>
```
Conclusion

• Test Data in test models often neglected
  • Definition potentially hard task
  • Complex data problematic

• Solution: data from usage
  • Monitored data must be of high quality
  • Test models must be of high quality
  • (Textual) Mapping must be possible