Model-based Testing of Software Product Lines – Part I

Prof. Ina Schaefer, 09.06.2015, Halmstad Summer School on Testing
Dynamic Black-Box Software Testing

1. Specification
2. Test Case Design
3. Tester
4. System under Test (SUT)
5. Test Case Execution
6. O
Roadmap

- Model-based Testing - Overall Concept
- Model-based Testing - Some Theory
MBT – Overall Idea

„MBT encompasses the processes and techniques for the automatic derivation of abstract test cases from abstract models, the generation of concrete tests from abstract tests, and the manual or automated execution of the resulting concrete test cases.“

Process of MBT

- Test Selection Criteria
- Requirements
- Test Cases
- Test Script
- Verdicts

Traceability

Requirements → Reqs–Model → Model → Model–Tests → Tests

Model:
- T1: Init ; Op1 ; Op2(NN,AA) ; Op1
- T2: Init ; Op3(AA) ; Op1 ; Op1
- T3: Init; Op3(BB)
- T4: Init ; Op3(AA) ; Op3(BB) ; Op1
- T5: Init ; Op1 ; Op1 ; Op1 ; Op1

Reqs–Tests

From [Utting/ Legeard, Practical MBT, 2007]
MBT – Example: State Charts

Example:

Coverage Criteria:
Scope of MBT

- Functional
- Usability
- Performance
- Robustness

Scale of SUT

- System
- Component
- Integration

Requirements (Black Box)

- Code (White Box)

Tests derived from...

Characteristics being tested

From [Utting/ Legeard, Practical MBT, 2007]
The choice of these six dimensions directly reflects the process introduced in Section 2. Step 1 (building the model) is reflected by the three dimensions within the model specification category: scope, characteristics, and modelling paradigm. Steps 2 and 3 (choosing test criteria and building test case specifications) are reflected by the test selection criteria dimension within the test generation category. Step 4 (generating tests) is reflected by the technology dimension within the test generation category. Step 5 (running tests) is reflected by the on/offline dimension of the test execution category.

Other perspectives that give rise to a taxonomy of MBT and that do not start from the process can, of course, also be justified. For instance, one could also start from the different artefacts that are developed or used in that process, e.g. models, test specifications, test drivers, properties, tests, etc. The rationale for the decision to use the process as a basis is that it is easier to agree on the activities of the process, and thus to justify the completeness of the taxonomy, than to agree on the different relevant artefacts. This, of course, does not mean that such a different taxonomy would not be valuable as well.

### 3.1. Model scope

The first dimension is the scope of the model, which is classified into a binary decision: does the model specify only the inputs to the SUT, or does it specify the expected input–output behaviour of the SUT? The input-only models are generally easier to specify, but they have the disadvantage that the generated tests will not be able to act as an oracle. The generated tests may implement an implicit 'robustness' oracle, such as checking that the SUT does not crash or throw any exceptions, but they cannot check the correctness of the actual SUT output values, since the model does not specify the expected output values. So input-only models produce weak oracles that are incapable of verifying the correctness of the SUT functional behaviour.

Input–output models of the SUT not only model the allowable inputs that can be sent to the SUT, but must also capture some of the intended behaviour of the SUT. That is, the model must...
Advantages of MBT

- Systematic and automatic test case generation
- Useable in early development phases (Model-in-the-Loop)
- Automatization of test case execution
- Regression test planning by analysis of model changes
Model-Based Testing – A more theoretical perspective

- Test model
- Model-based test generation
- System model
- Test execution
- SUT
- Pass/fail
Conformance Relations

implementation relation: \( i \approx s \) with *implementation* \( i \) and *formal behavioral specification* \( s \)

preorder relation: \( i \preceq s \) implementation shows at most the behaviors of the specification

intentional conformance: \( i \text{ conforms } s :\iff \llbracket i \rrbracket \subseteq \llbracket s \rrbracket \)

where \( \llbracket \cdot \rrbracket \) defines sets of all observable behaviors

extensional conformance: \( i \text{ conforms } s :\iff \forall u \in \mathcal{U} : \text{obs}(u, i) \approx \text{obs}(u, s) \)

where \( \mathcal{U} \) defines sets of all observers
Model-based I/O Conformance Testing

- Proposed by Jan Tretman in the 90’s
- Model-based functional conformance testing of systems with reactive, non-deterministic behaviors
- Input, output, and quiescence based testing theory
- Based on I/O labeled transition systems as test models AND implementation models
- Proven sound and exhaustive
- Rich tool support
- Formal basis for many advanced testing frameworks
Running Example

Beverage vending machine

- Input actions
  - $I = \{1\text{€}, 2\text{€}\}$
  - Transitions labels prefixed with “?”

- Output actions
  - $U = \{\text{coffee, tea}\}$
  - Transition labels prefixed with “!”
I/O-Labeled Transition Systems

I/O Labeled Transition System: \((Q, q_0, I, U, \rightarrow)\), where

- \(Q\) is a countable set of states,
- \(q_0 \in Q\) is the initial state,
- \(I\) and \(U\) are disjoint sets of input actions and output actions, and
- \(\rightarrow \subseteq Q \times \text{act} \times Q\) is a labeled transition relation.
\[ T_r(q_1) = \{ ?1\€, ?1\€ \cdot !coffee, ?1\€ \cdot !tea \} \]

\[ T_r(q_5) = \{ ?1\€, ?2\€ \} \]
Each computation refers to some path

\[ q_0 \xrightarrow{\mu_1} s_1 \xrightarrow{\mu_2} s_2 \xrightarrow{\mu_3} \ldots \xrightarrow{\mu_{n-1}} s_{n-1} \xrightarrow{\mu_n} s_n \]

The behavior of a computation is defined by a trace

\[ \text{trace } \sigma = \mu_1 \mu_2 \ldots \mu_n \in \text{act}^* \]
LTS - Examples

\[ T_r(q_3) = \{ ?1\€, ?2\€, ?1\€ \cdot !coffee, ?2\€ \cdot !coffee \} \]

\[ T_r(q_2) = \{ ?1\€, ?2\€, ?1\€ \cdot !coffee, ?1\€ \cdot !tea, ?2\€ \cdot !coffee, ?2\€ \cdot !tea \} \]
Let \( s \) be an I/O LTS, \( \mu_i \in I \cup U \cup \{ \tau \} \) and \( a_i \in I \cup U \)

\[
\begin{align*}
s \xrightarrow{\mu_1 \cdots \mu_n} s' & := \exists s_0, \ldots, s_n : s = s_0 \xrightarrow{\mu_1} s_1 \xrightarrow{\mu_2} \cdots \xrightarrow{\mu_n} s_n = s' \\
\neg s \xrightarrow{\mu_1 \cdots \mu_n} & := \forall s' : s \xrightarrow{\mu_1 \cdots \mu_n} s'
\end{align*}
\]
LTS Trace Notations

Let $s$ be an I/O LTS, $\mu_i \in I \cup U \cup \{\tau\}$ and $a_i \in I \cup U$

\[
\epsilon \quad s \Rightarrow s' := s = s' \text{ or } s \xrightarrow{\tau \cdots \tau} s'
\]

\[
a \quad s \Rightarrow s' := \exists s_1, s_2 : s \Rightarrow s_1 \Rightarrow s_2 \Rightarrow s'
\]

\[
\begin{array}{c}
a_1 \cdots a_n \Rightarrow s' := \exists s_0, \ldots, s_n : s = s_0 \Rightarrow s_1 \Rightarrow \cdots \Rightarrow s_n = s'
\end{array}
\]
LTS Trace Notations

Let $s$ be an I/O LTS, $\sigma \in (I \cup U)^*$

$$
\begin{align*}
\sigma s & \Rightarrow := \exists s' : s \Rightarrow s' \\
\neg \sigma s & \Rightarrow := \forall s' : s \Rightarrow s'
\end{align*}
$$
LTS Trace Notations

The set of traces in an LTS is defined as

\[ Tr(s) := \{ \sigma \in (I \cup U)^* | \exists s' \in Q : q_0 \xRightarrow{\sigma} s' \}. \]
LTS - Examples

\[ Tr(q_4) = \{? 1\text{\euro}, ? 2\text{\euro}, ? 1\text{\euro} \cdot !\text{coffee}, ? 2\text{\euro} \cdot !\text{tea} \} \]

\[ Tr(q_6) = \{? 1\text{\euro}, ? 1\text{\euro} \cdot !\text{coffee} \} \]
$T(r(q_7)) = \{?1€, ?1€ \cdot !coffee\}$

$T(r(q_8)) = \{}$
An \( LTS \) is (weak) input-enabled iff for every state \( s \in Q \) with \( q_0 \Rightarrow^* s \) and for all \( a \in I \) it holds that \( s \xrightarrow{a} \).
Input Completion - Example

Not input-enabled LTS

Input-enabled LTS
A First Attempt: Conformance as Trace Inclusion

\[ i \text{ conforms } s : \iff Tr(i) \subseteq Tr(s) \]

- Fails to refuse trivial implementations
- Fails to take the asymmetric nature of \( \mathcal{LTS} \) traces with I/O actions into account

Solution: explicit notion of quiescent behavior

Solution: distinguish input and output behaviors in traces
What is the difference?

\[ Tr(q_6) = \{ \text{? \(1\,\text{€}\), ? \(1\,\text{€}\) \(\cdot\) !coffee} \} \]

\[ Tr(q_7) = \{ \text{? \(1\,\text{€}\), ? \(1\,\text{€}\) \(\cdot\) !coffee} \} \]
Some Auxiliary Definitions: Init Sets

Let $s$ be an $\mathcal{LTS}$, $p \in Q, P \subseteq Q$ and $\sigma \in (I \cup U)^*$. 

$$\text{init}(p) := \{ \mu \in (I \cup U)^* | p \xrightarrow{\mu} \}$$

\begin{align*}
\text{init}(p_1) &:= \{ ? 1€ \} \\
\text{init}(p_2) &:= \{ !\text{coffee}, !\text{tea} \} \\
\text{init}(p_3) &:= \{ \} \\
\text{init}(p_4) &:= \{ \}
\end{align*}
Some Auxiliary Definitions: Quiescent States

Let $s$ be an LTS, $p \in Q, P \subseteq Q$ and $\sigma \in (I \cup U)^*$.

$p$ is **quiescent**, denoted $\delta(p)$, if $\text{init}(p) \subseteq I$

$$
\begin{align*}
\text{init}(p_1) & := \{ ?1\€ \} \\
\text{init}(p_2) & := \{ !\text{coffee}, !\text{tea} \} \\
\text{init}(p_3) & := \{ \} \\
\text{init}(p_4) & := \{ \} \\
I & = \{ ?1\€, ?2\€ \}
\end{align*}
$$
Some Auxiliary Definitions: Suspension Traces

Let $s$ be an LTS, $p \in Q$, $P \subseteq Q$ and $\sigma \in (I \cup U)^*$. 

$$Straces(p) := \{\sigma' \in (I_S \cup U_S \cup \{\delta\})^* | p \xrightarrow{\sigma'} q \} \text{ where } q \xrightarrow{\delta} q \text{ iff } \delta(p)$$

$Straces(p_1) = \{\delta, ?1€, ?1€ \cdot !coffee, ?1€ \cdot !tea, ?1€ \cdot !coffee \cdot \delta, ?1€ \cdot !tea \cdot \delta\}$

$Straces(p_2) = \{!coffee, !tea, !coffee \cdot \delta, tea \cdot \delta\}$

$Straces(p_3) = \{\delta\}$

$Straces(p_4) = \{\delta\}$
Quiescent Behaviors

Trace $Tr(q_1)$

$Straces(q_1) = \{\delta, ?1\€, \delta \cdot ?1\€, ?1\€ \cdot !coffee, ?1\€ \cdot !coffee \cdot \delta, \ldots\}$

Allows to discriminate (non-)behaviors

- $?1\€ \cdot \delta \notin Straces(q_6)$, whereas $?1\€ \cdot \delta \in Straces(q_7)$
Some Auxiliary Definitions: After Sets

Let $s$ be an LTS, $p \in Q, P \subseteq Q$ and $\sigma \in (I \cup U)^*$. 

$p \text{ after } \sigma := \{ q \in U \mid p \xrightarrow{\sigma} q \}$

$U = \{!\text{coffee},!\text{tea}\}$

$p_2 \text{ after } !\text{coffee} = \{p_3\}$

$p_2 \text{ after } !\text{tea} = \{p_4\}$

$p_1 \text{ after } ?1\epsilon = \{}$

$p_4 \text{ after } !\text{tea} = \{}$
Some Auxiliary Definitions: Out Sets

Let $s$ be an LTS, $p \in Q, P \subseteq Q$ and $\sigma \in (I \cup U)^*$.

\[ \text{out}(P) := \{ \mu \in U \mid \exists p \in P : p \xrightarrow{\mu} \} \cup \{ \delta \mid \exists p \in P : \delta(p) \} \]

- $P = p \text{ after } \sigma$
  - $P_1 = \{ p_1 \text{ after } \delta \} = \{ p_1 \}$
  - $P_2 = \{ p_2 \text{ after } !tea, p_2 \text{ after } !coffee \} = \{ p_3, p_4 \}$
  - $P_3 = \{ p_3 \text{ after } \delta \} = \{ p_3 \}$
  - $P_4 = \{ p_4 \text{ after } \delta \} = \{ p_4 \}$
Some Auxiliary Definitions: After-Out Sets

Let $s$ be an LTS, $p \in Q$, $P \subseteq Q$ and $\sigma \in (I \cup U)^*$. 

$$\text{out}(P) := \{\mu \in U \mid \exists p \in P : p \xrightarrow{\mu}\} \cup \{\delta \mid \exists p \in P : \delta(p)\}$$

$P_1 = \{p_1 \text{ after } \delta\} = \{p_1\}$
$P_2 = \{p_2 \text{ after } !\text{tea}, p_2 \text{ after } !\text{coffee}\} = \{p_3, p_4\}$
$P_3 = \{p_3 \text{ after } \delta\} = \{p_3\}$
$P_4 = \{p_4 \text{ after } \delta\} = \{p_4\}$

$\text{Out}(P_1) = \{\delta\}$
$\text{Out}(P_2) = \{!\text{tea}, !\text{coffee}\}$
$\text{Out}(P_3) = \{\delta\}$
$\text{Out}(P_4) = \{\delta\}$
Second Attempt: I/O Conformance (IOR)

Let $s \in \mathcal{LTS}(I \cup U)$ and $i \in \mathcal{JOTS}(I, U)$.

\[ i \text{ ior } s \iff \forall \sigma \in \text{act}_{\delta}^*: \text{out}(i \text{ after } \sigma) \subseteq \text{out}(s \text{ after } \sigma) \]

\[ i \text{ ior } s \iff \text{Straces}(i) \subseteq \text{Straces}(s) \]
Example

- Assume $\delta$-transitions in the $LTS$ example

- Possible environmental stimulations are $\sigma = ?1\€$ and $\sigma' = ?2\€$

- Investigate the observable behavior
Example

\[ \text{out}(q_1 \text{ after } \sigma) = \{\text{coffee, tea}\}, \text{out}(q_1 \text{ after } \sigma') = \{\} \]

\[ \text{out}(q_2 \text{ after } \sigma) = \{\text{coffee, tea}\}, \text{out}(q_2 \text{ after } \sigma') = \{\text{coffee, tea}\} \]
Example

\[ \text{out}(q_3 \text{ after } \sigma) = \{\text{coffee}\}, \text{out}(q_3 \text{ after } \sigma') = \{\text{coffee}\} \]

\[ \text{out}(q_4 \text{ after } \sigma) = \{\text{coffee}\}, \text{out}(q_4 \text{ after } \sigma') = \{\text{tea}\} \]
Example

\[
\text{out}(q_5 \text{ after } \sigma) = \{\delta\}, \text{out}(q_5 \text{ after } \sigma') = \{\delta\}
\]

\[
\text{out}(q_6 \text{ after } \sigma) = \{\text{coffee}\}, \text{out}(q_6 \text{ after } \sigma') = \{\}
\]
Let $s \in LTS(I \cup U)$ and $i \in IOTS(I, U)$.

Problem: this is quite a lot!

\[
i \text{ior } s \iff \forall \sigma \in act_\delta^* : out(i \text{ after } \sigma) \subseteq out(s \text{ after } \sigma)
\]

\[
i \text{ior } s \iff Straces(i) \subseteq Straces(s)
\]
Third Attempt: IOCO

Let $s \in \mathcal{LTS}(I \cup U)$ and $i \in \mathcal{IOTS}(I, U)$.

$$i \ ioco \ s \iff \forall \sigma \in \text{Straces}(s) : \text{out}(i \ after \ \sigma) \subseteq \text{out}(s \ after \ \sigma)$$

$$ior \subseteq ioco$$

Focus on specified behaviors only
Example [Tretmans, 1999]
Third Attempt: IOCO

Let \( s \in LTS(I \cup U) \) and \( i \in IOTS(I, U) \).

\[
i \ ioco \ s \iff \forall \sigma \in Straces(s) : out(i \ after \ \sigma) \subseteq out(s \ after \ \sigma)
\]

\( ior \subseteq ioco \)

Still infinite in case of loops
• The set of suspension traces under consideration is restricted to sub sets $F \subseteq act^*$
• The restricted ioco relation is denoted as

\[ i \text{ ioco}_F s \iff \forall \sigma \in F : out(i \text{ after } \sigma) \subseteq out(s \text{ after } \sigma), \]

where $ior = ioco_{act^*}$ and $ior = ioco_{Straces(s)}$ holds.

This is still an intentional characterization of conformance. How to prove this by testing?
Observers (testers) are characterized by a finite sets of test cases they perform on an SUT.
Test Cases

A test case $t$ is an I/O labeled $\mathcal{LTS}$ such that

- $t$ is deterministic and has a finite set of traces,
- $Q$ contains terminal states $\text{pass}$ and $\text{fail}$ with $\text{init}(\text{pass}) = \text{init}(\text{fail}) = \emptyset$,
- for each non-terminal state $q \in Q$ either
  1. $\text{init}(q) = \{a\}$ for $a \in I$ or
  2. $\text{init}(q) = U \cup \{\theta\}$

holds.

By $\mathcal{TES}T$ we denote the subclass of I/O labeled $\mathcal{LTS}$ representing valid test cases $t$. 

denotes observation of quiescence
Test Case Generation – Example

Test case for specification $q_1$

Stimulated input
- $!1€$

Expected output
- *either coffee*
- *or tea*

Observable errors
- No output occurs: $\Theta$

Diagram:
- $q_1$ with input $!1€$
- Two paths:
  - $!coffee$ to pass
  - $!tea$ to pass
- $t_1$ with output $!1€$
  - $?coffee$ to pass
  - $?tea$ to pass
  - $\Theta$ to fail
Test Case Generation – Example 2

Test case for specification \( q_7 \)

Stimulated input
  - \(!1€\)

Expected output
  - \( \textit{coffee} \)
  - no output: \( \Theta \)

Observable errors
  - \( \textit{tea} \)
IOCO is correct = sound + exhaustive

Let \( s \in \mathcal{LTS}(I \cup U), i \in \mathcal{IOTS}(I \cup U) \) and \( F \subseteq \mathcal{Straces}(s) \)

Then it holds that

1. the set \( \mathcal{TEST} \) of all derivable test cases is **sound** and
2. the set \( \mathcal{TEST} \) of all derivable test cases is **exhaustive**.
Conclusion

Model-based testing

• automates black-box test case generation and execution.

• requires a formal model of the system specificatic

• can be based on a formal notion of conformance.
Some Further Readings

- Malte Lochau, Sven Peldszus, Matthias Kowal, Ina Schaefer: Model-Based Testing. SFM 2014: 310-342
