R. Gerlich, R. Gerlich (BSSE)

Integrated Design and Testing of Safety-Critical Real-time Systems in Space

MBTUC11
Model-Based Testing

MBT User Conference 2011
19.10.2011 Berlin, Germany
Model = Abstract representation of a system

Test in context of V&V:
• Tests support verification
• Tests support validation

Verification
• provide input-output vectors
• support fault injection

Validation
• provide feedback on specification
• provide performance figures

MBT in context of MDE and Automation

Code generator generates the code
+ instrumentation for recording of properties

Test generator generates the test environment
for stimulation

Verification

Trash on modelling level
= More trash on target level

Root of both is the
MODEL

The whole process is driven
from the model only!

No manual intervention!

Refine the Model
(and the Modelling Environment) using
Feedback from the System

V&V on target is in focus, too

Automatic Verification
Automatic Generation
Automatic Testing

The Power of Fully Automated Process Chains and Integration

Modelling Level

- RT & Commun. Infrastructure Domain
  - Integration on higher Level after domain-specific testing
  - DSL

Target Level

- Executable Specification + Test Stimulation
  - 40 processes
  - 2 processors
  - 300 state transitions
  - 100 states
  - 300 commands
  - 500 data items
  - O/B database
  - Calibration
  - Monitoring pre-/post-proc.

- Visualised Properties
  - Feedback for Validation of Specification

- Test Generation
  - Code Generation

- Data Processing & Archiving
  - Executable Specification + Test Stimulation
  - 10 min

- Visualised Properties
  - Feedback for Validation of Specification

- Test Generation
  - Code Generation

- Execution on higher level after domain-specific testing

- Continuous correlation model – target
- Continuous feedback from executable system in representative environment

Feedback by Visualisation (examples only)

Due to automated model-based stimulation
immediate feedback from a system executable in a representative environment
What is a Model?

A model represents a specification

What is UML?

Ada code when re-engineering?

Contents of a spreadsheet?

DSL code?

Taking a specification as base for tests

All model types we have used for

- code generation (fully automated)
- test generation (fully automated)
What is a Test Input on Modelling Level?

A test plan and test procedures derived from a model?

Test stimuli automatically derived from a model and automatically documented together with results?

Stimulus for an FSM valid or invalid

Stimulus for commands, msgs and data valid or invalid, lost

A set of non-functional parameters deadline, timeout, period

A variation of ideal parameters time jitter, execution time

A test input may also be omitting of an expected input e.g. in case of fault injection: loss of data or events important for critical, fault-tolerant systems

Automatically derived test stimuli may also support early operation of a system
Material Science Laboratory ISS

2000: tool delivered to customer
2003: system accepted by ESA
2009: launched and put into operation

real-time infrastructure + TC + TM + database:
~80 KLOC when tool delivered
expanded by customer over 3 years

Generation:
~15 min from modelling language ISGL or Excel
automatically generated reports on properties
Input: ~500 Excel-lines

~ 40 processes
~200 states
2 processors

3 faults detected in first version of code and test generators from 2000 - 2003:

- limitation to 250 ground commands
- task priority list for distributed system not correct
- overflow in union (16 bit cmd counter)
- no more faults flagged from the project since 2003/2009
Early Design Validation

State and State Transition Coverage vs. FI probability

- FSM stimulation by messages
- Loss of messages / events
- Varying fault probability

High probability of a system collapse already at very small probability of 0.5% for data loss

Evaluation Means
Fault Injection = Inversion of positive functionality

System Initialisation Procedure
<Probability to reach the end> = ?
How critical is my system?
Pure functional verification based on UML failed:
- non-functional properties were out-of-scope
- performance and fault tolerance issues could not be detected due to stepwise stimulation
- while massive stimulation is required

verified previously by a UML tool but no support of non-functional properties especially of fault injection
Evaluation of non-functional properties
In presence of Fault Injection

Dead code due to distributed logic spreading over two processes

Deadcode in model

Deadlock in case of loss of signal though fault tolerance should cover it

Erroneous fault-tolerant approach: lost signal is really lost, but next signal is duplicated
result of verification ACK or NAK

Wrong cmd stored

Missing requirement on WCET or Uplink Rate

Potential Loss of Commands

Missing performance requirement on uplink rate or WCET

Extracting a Model from Ada code: Coverage of Finite State Machines

- Random stimulation
- Dynamic stimulation of FSMs, inputs derived from the model
- Surprising, because reachability of states was not a test goal:
  - Sub-sets (nets) of states, no transition possible
- Fault? in this case: hidden information ⇒ testability?
- Conclusion: difficult (impossible) verification of application regarding behaviour
Performance Characteristics

- **Ada**
  - ~1 Mio. lines of source code, ~430 KLOC

- **FSM**
  - 38 FSMs / processes
  - 616 different commands (inputs of FSMs, stimuli)
  - 637 commands in total, tuples of (FSM,cmd)
  - 360 different states
  - 381 states in total, tuples of (FSM,state)
  - 1475 transitions (names)
  - 4695 different tuples (FSM, msg, initial state, final state)
  - 9778 atomic actions in FSMs

- **Time statistics**
  - ISGL model generation from Ada: < 5s
  - system code generation: ~10min
  - stimulation: 2000% coverage of input domain (20x at least) ~70 min. (~ 3cmds/s)
Evaluation of non-functional properties: Distributed Synchronous System

Synthesis of two models
ISGL for behaviour and real-time
Scade/Lustre for control algorithms

Integration on C code level
Stimulation from behavioural model
Stimulation data provided from Scade analysis

Sensor inputs

Preprocessing

Monitoring

Redundancy

Controller Outputs

Actuators

Operator Interface

Controller Outputs

FB

2/2 2/2

Mon1 Mon2 Mon3

PP1 PP2 PP3

man/machine interface

Conronic E

Ops 1 Ops 2

Voter output: yes / no

Impact of Time Jitter and Data Loss

high fault rate at rather low time jitter and/or low rate of loss of data

↑ % faults / voter discrepancies

Theoretical prediction and "confirmation" after raising doubts, but before ISG V&V: "should be robust in case of time jitter"
Verification of the Code Generator

Tests automatically derived from a model require the generated code to unveil its properties.

Reference Model → Automated Code + Test Generation → Execution of Generated Code → Observation of Properties

Oracle → Transformation → Automated Code + Test Generation

Comparison

Reference Model