Testing of Event-Triggered Real-time Systems (TETReS)

Birgitta Lindström
Mats Grindal
Robert Nilsson
Overview

- Us
- Vision
- Problem
- Approach
  - Design for testable real-time
  - Automatic testing of timeliness
  - Evaluation of testing methods
- Current status
TETReS people

Prof. Sten F. Andler

Jonas Mellin

Mats Grindal (50%)

Birgitta Lindström (80%)

Robert Nilsson (80%)

November 19, 2002

Robert Nilsson
University of Skövde
Vision

Tools and methods for supporting system level testing

- aim external quality attributes, e.g.
  - timeliness
  - robustness
  - reliability
- Determine relevant test methods and quality metrics
- Estimate testability of real-time system designs
- Select and execute test cases automatically
- Evaluate results (using event monitor logs)
**Project focus**

- Testing for timeliness in event-triggered systems
  - Birgitta, increasing testability while maintaining flexibility
  - Robert, automatic selection and generation of test cases

- Testing for different quality attributes
  - Mats, effective combinations of methods
Testing: The onion model

- Multi-processor
- ET Real-time
- TT Real-time
- Concurrent
- Seq. software
- Distributed
Testability issues: Observability

- **Observability (Necessary)**
  - Possibility to observe system during testing
  - Minimum instrumentation
    - Task synchronization, external interrupts, and access to time
  - Instrumentation left in operational system
    - Avoid probe effect

- **Controllability (Desired)**
  - Possibility to enforce a desired system state
Testability issues: Timeliness

- Often dense time base in dynamic real-time systems
- Classic test cases
  - A seq. of typed events with values and time stamps
  - (System state = Task state + global variable state + time)
    - Seldom considered
- Consequences in dynamic real-time systems
  - Small variations may lead to different behaviors
  - Same test case may give different timely behavior
  - Hard to identify worst cases
Example: Observation of events

- **Event occurrence**
- **Task**
- **State** $s_0$
- **Events** $e_1$, $e_2$

**EDF order:** $x_0$, $x_2$, $x_1$

**States:**
- $s_0$

**Tasks:**
- $x_0$, $x_1$, $x_2$

**Deadlines:**
- $x_0$
- $x_1$
- $x_2$

**Timeline:**
- $t_0$
- $t_2$

**Event occurrence diagram:**
- $e_1$
- $e_2$
Example: Varying execution time

EDF order: $x_0, x_2, x_1$

$C(x_1) > t(e_2) - t(e_1)$

Actual order: $x_0, x_{1-1}, x_2, x_{1-2}$

EDF order: $x_0, x_2, x_1$

$C(x_1) < t(e_2) - t(e_1)$

Actual order: $x_0, x_1, x_2$
Assumptions on system model

- Dynamic scheduling (e.g. EDF)
- Admission control
- Observability
- Mixed task load
- Known
  - Bound on number of instances of task types
  - Task dependencies (e.g., output – input)
  - Relative deadlines
Approach to increase testability

✦ Adapt features that:
  – Increase testability
  – While maintaining semantics of dynamic real-time systems

✦ Examples
  – Predefined observation points
    • Fewer time points of events to test
  – Designated preemption points
    • Reduces the number of interleavings
Testing: Onion model

- ET Real-time
- Constrained ET Real-time
- TT Real-time
- Concurrent
- Seq. software

Distributed

Multi-processor
Observation of events (continued)

EDF order: $x_0, x_1, x_2$

Event occurrence  Task  Observation point

State $s_0$
Events $e_1, e_2$
Event-triggered semantics

- Tasks are triggered by events (although delayed)
  - Not executed in predefined time-slots
  - The delay must be considered during analysis
- Dynamic scheduling and preemptions support
  - Overload resolution
  - Graceful degradation
Approach to test case generation

- Formalized model to describe assumptions about environment and task characteristics
  - Criticality
  - Estimated execution times (best/worst case)
  - Event patterns (external and internal)

- Testing criteria specify what tests should cover
  - Stress admission control and scheduling scheme
    - e.g., try to make critical task break their deadlines
Test case generator

Real-time System

System model

Execution environment constraints

Test cases

Test case execution outcomes

Evaluator

Birgitta

Robert Mats

Jonas (EM)
Approach to test case execution

- **Transaction processing platform**
  - Allow us to focus on timeliness testing
  - Isolation maintained by the transaction manager
    - E.g. strict conservative locking

- **Non-intrusive observability**
  - Built in event-monitor

- **Support constraints on transaction behavior**
  (Birgisson, Mellin and Andler, 1998)
  - Bound number of designated preemption points
    - Decrease number of execution orders
    - Simplify specifying state of active transactions
Complete Test-Case Specifications

- Include system-wide state information in test-case specifications
  - State of each active transactions
    - Preempted / Blocked / Executing
  - State of resources
    - Equivalence classes
  - Current dynamic schedule
    - Deadlines and criticality of active tasks

- Include data parameters with input event-sequence
  - E.g. sensor reading

→ More deterministic behavior during test execution
→ Possible to automatically select initial states with interesting temporal behavior
Test case execution method

Test case setup phase(1)
- Lock resources
- Execute transactions to a specified preemption point
- Create current schedule
- Save system-state

Test – execution phase(2)
- Start execution
- Insert event-sequences

Parameterized Event-sequence
Monitor and Log
Interesting temporal behavior
Testing in a development project

- Quality of the system
  - Many system properties important
  - Some types of faults need to be avoided

- Quality of the testing
  - Measure total coverage

- Time and Cost
  - Available resources are bounded
Complementary Test methods

- Algorithms for identifying test cases
- Different focus, strengths and weaknesses
  - Quality attributes
  - Types of faults
  - Cost
  - …
- How evaluate test methods
  - Suitable combinations for specific systems
Current status

★ Current work
  – 2 Thesis proposals (Robert, Birgitta)
    • One problem, initial solutions, concrete time-plan
  – 1 Research/Thesis Proposal (Mats)

★ Previous work
  – 2 research proposals completed
    • Literature surveys, Initial Approach/Experimentation
    • Identification of open problems
  – 6 articles
  – 1 MSc thesis (Ragnar Birgisson)
  – 2 third year final project reports (Robert, Birgitta)
Potential future works

❖ Automatic generation of ECA-based applications
  – Formal spec. used for testing applications
    • Why not generate application-skeletons / design from the same notation?
    • ECA-rules are interesting on Tx-based platforms
    • Preliminary work initiated

❖ Self-reflecting components for testing
  – Built-in meta-information for testing
    • Extract parts of formal spec. from applications
    • Standardized test-interfaces (in e.g. corba/com objects)
    • Self-testing RT-systems

❖ Testing distributed dynamic systems