Model-based integration and testing of high-tech multi-disciplinary systems

Niels Braspenninck
Asia van de Mortel-Fronczak
Koos Rooda

Systems Engineering Group
Mechanical Engineering Department
Eindhoven University of Technology
Contents

• Rationale
• Model-based integration and testing (MBI&T)
• Case: EUV machine
  – Modeling
  – Model-based system analysis
  – Model-based system testing
• Conclusions
• Discussion
High-tech multi-disciplinary machines

ASML TWINSCAN wafer scanner

Properties:
• Multi-disciplinary
• >12 M LoC
• 13.8 M € average selling price

Performance (XT:1900i)
• < 40 nm line width
• < 6 nm overlay
• > 131 WPH throughput

Source: www.asml.com
Integration and test problem

- Trends in high-tech multidisciplinary system development
  - Increasing complexity
  - Decreasing time to market
  - Increasing integration and test effort

- Resulting effects
  - Decreasing quality ($Q \downarrow$)
  - Long integration and test time ($T \uparrow$)
  - High costs for fixing integration problems late in the process ($C \uparrow$)
TANGRAM research project

- Develop methods and tools to reduce integration and test effort
- ‘Industry as laboratory’: proof of concept in industrial environment of ASML
- Managed by Embedded Systems Institute (ESI)
- Several industrial and academic partners (60 FTE in total, 5 PhD students)
Research objectives

• Develop methods, theories, and tools for modeling, analysis, and testing of components to increase quality ($Q^\uparrow$) and to reduce the integration and test time and costs ($T\downarrow,C\downarrow$) of these components

• Demonstrate these methods, theories, and tools in industrial case studies
Model-based integration and testing (MBI&T)

Formal executable models enable

- Powerful analysis techniques that give the engineer a better understanding of the behavior of the system
- Early integration of all possible combinations of models and realizations of components

Leading to

- Earlier system level analysis and testing (Q↑)
- Earlier (and thus cheaper) finding and fixing of integration problems (C↓)
- Reduction of integration and test time (T↓)
Current systems development

Requirements $R$, designs $D$, and realizations $Z$ of a system with $n$ components and infrastructure $I$
MBI&T systems development

Requirements $R$, designs $D$, models $M$, realizations $Z$ of a system with $n$ components and infrastructure $I$ that allows integration of models and realizations.
MBI&T instantiation

• Modeling
  – Paradigm: concurrent processes
  – Mathematics: automata (FSM) theory and process algebra
  – Tools: timed \( \chi \) (timed discrete-event, communication, data)

• Analysis
  – Simulation: timed \( \chi \), Python, and C
  – Verification (model checking): Uppaal, Spin, \( \mu \)CRL
  – Testing: Jabber publish/subscribe, Python, CORBA, remote I/O
  – Visualization: animated automata, message sequence charts, Gantt charts
MBI&T instantiation

- **$R$: Requirements**
- **$D$: Design**
- **$M$: Model**
- **$Z$: realiZation**

\[ \chi \text{ simulator} \]

**System model simulation**

- Uppaal, Spin, $\mu$CRL

**System model verification**

**Real-time $\chi$ simulator**

**Model-based system testing**

- Jabber pub/sub, Python, CORBA, remote I/O, …

- **Simulator**

- **Jabber**

- **Visualization**

- **Simulation**

- **Verification**

- **Testing**

- **Integration**

- **Requirements**

- **Design**

- **Model**

- **Integration**
Contents

• Rationale
• Model-based integration and testing (MBI&T)
• Case: EUV machine
  – Modeling
  – Model-based system analysis
  – Model-based system testing
• Conclusions
• Discussion
Case: ASML EUV machine

- Source for extreme ultra violet light

- Focus on vacuum-source interface
  - Modeling
  - Model-based system analysis
  - Model-based system testing
Case: modeling

Concurrent processes (communication, timing)  Vacuum System automaton (internal behavior)
Case: model-based system analysis

*R*: Requirements  
*D*: Design  
*M*: Model  
*Z*: realiZation

1. **Define**  
   *R*: Requirements  
   *D*: Design

2. **Design**  
   *R*: Requirements  
   *D*: Design

3. **Model**  
   *M*: Model  
   *Z*: realiZation

4. **Realize**  
   *M*: Model  
   *Z*: realiZation

5. **Integrate**  
   *R*: Requirements  
   *D*: Design  
   *M*: Model  
   *Z*: realiZation

**System Model Simulation**  
*Uppaal*

**System Model Verification**
Case: model-based system analysis

Simulation and visualization of scenarios

- Animated automata
- Message sequence charts
- Gantt charts
Case: model-based system analysis

Verification of complete model state space with UPPAAL
- Deadlock/livelock freeness
- No errors and no undefined behavior
- Safety properties, e.g. source active while vented
- Temporal, e.g. maximum sequence durations
Case: results

• Modeling
  – Errors and inconsistencies in reviewed design documents
    • Interface errors/incompleteness
    • Redundant and missing states
  – Better system overview
    • Improved communication between engineers
    • Modeling problems indicate design problems

• Analysis
  – Simulation
    • Source raises error during ‘good weather’ venting scenario
  – Verification
    • Two deadlocks caused by non-determinism in interrupt handler
    • One deadlock caused by concurrent events
    • Two modeling mistakes
    • Maximum sequence requirements are not met
Case: model-based system testing

- **R**: Requirements
- **D**: Design
- **M**: Model
- **Z**: realization

### Diagram

- **RS** → **D_S** → **RVS** → **DVS** → **ZVS**
- **RSSRC** → **D_SRC** → **model** → **realize** → **integrate**
- **Jabber pub/sub, Python, remote I/O**
- **timed χ**
- **Uppaal**
- **visual**
- **system model simulation**
- **real-time χ simulator**
- **model-based system testing**

### Relevant Information

- **system model simulation**
- **system model verification**
- **model-based system testing**
Case: test setup
Case: test results

- System testing 20 weeks before real integration
- 6 hours of test time instead of 4 days in clean room
- Found 6 errors that could lead to
  - Source damage (2 days of downtime in cleanroom)
  - Unnecessary waiting (multiple hours per test/sequence)
  - Unnecessary operator intervention
- Fast diagnosis, on the spot repairing, and fast retesting
Conclusions

- A model-based integration method has been developed and instantiated with tools for modeling and analysis.
- Successful proof of concept in case study:
  - Prevent integration problems in most expensive period (C↓)
  - Prevent source damage at real integration (C↓)
  - Prevent long down times at real integration (T↓)
  - Faster diagnostics during model-based system testing (T↓)
  - Modeling gives better system overview (Q↑)
  - Early fixing of source errors (Q↑)
Future work

- Extend EUV case
- Implications of using models and infrastructure for testing in an asynchronous and real-time environment
- Modeling effort vs. potential benefits

- Implications of using models and infrastructure for testing in an asynchronous and real-time environment
Model-based integration and testing of high-tech multi-disciplinary systems

Questions?

Niels Braspennin
Asia van de Mortel-Fronczak
Koos Rooda

Systems Engineering Group
Mechanical Engineering Department
Eindhoven University of Technology