



ASML

Integration and test strategies

Current research and future opportunities

Ivo de Jong, Roel Boumen, Asia van de Mortel-Fronczak, Koos Rooda



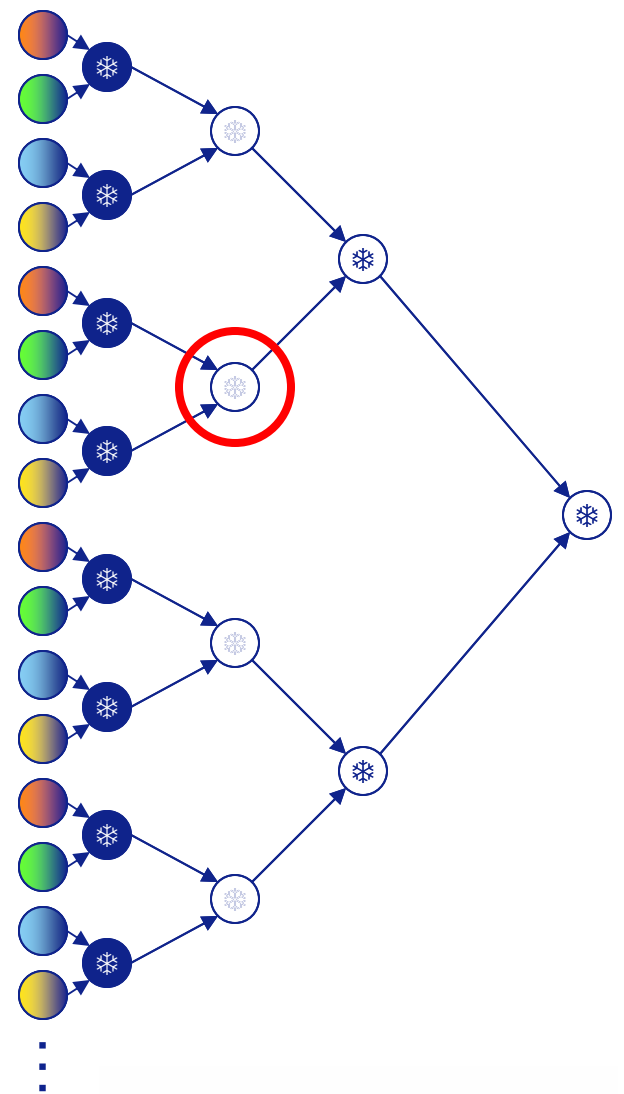
Contents

- Introduction
- Problem statement and objective
- Integration and test planning
- Current research: methods and results
- Detailed method: Next-best-test-case algorithm
- Future opportunities
- Conclusions

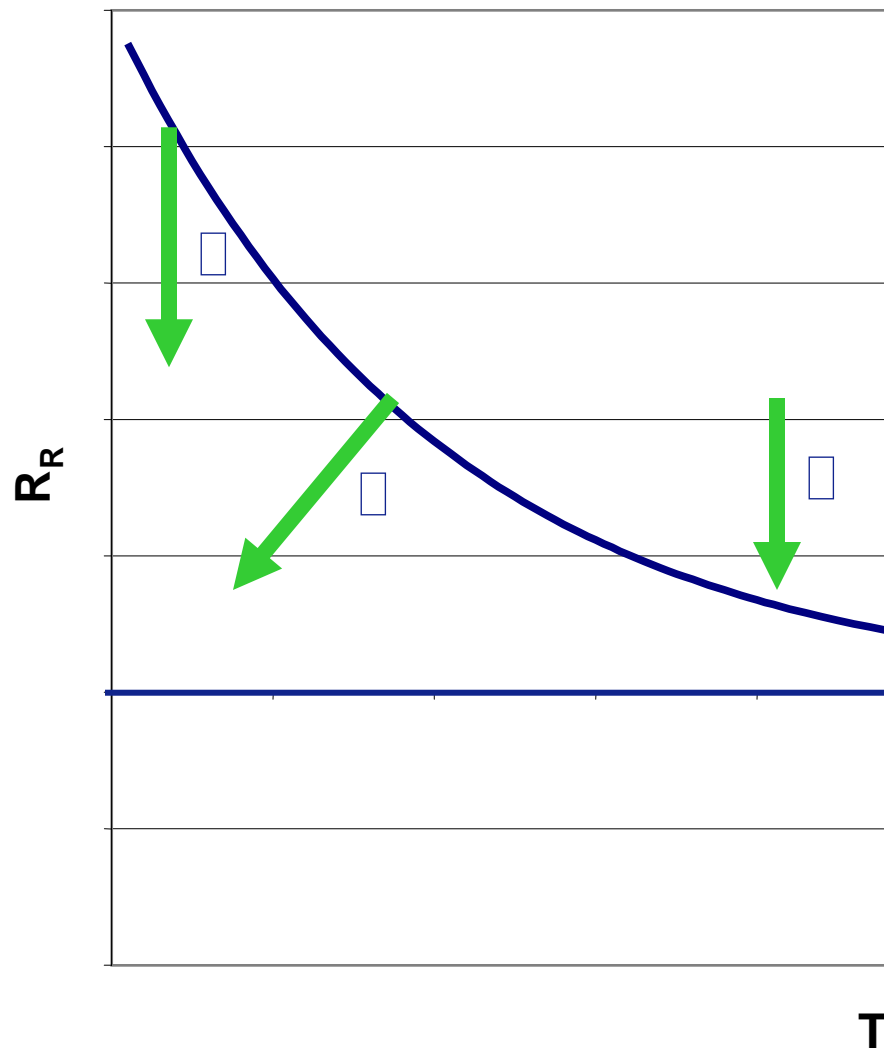


Introduction

- Tight specification
 - ▶ Many components (1000+)
 - ▶ Multi disciplinary components
 - ▶ Incomplete designs
- Time-to-market
 - ▶ Concurrent engineering
 - ▶ Incomplete test phases



Product quality (risk) vs test duration (1/2)

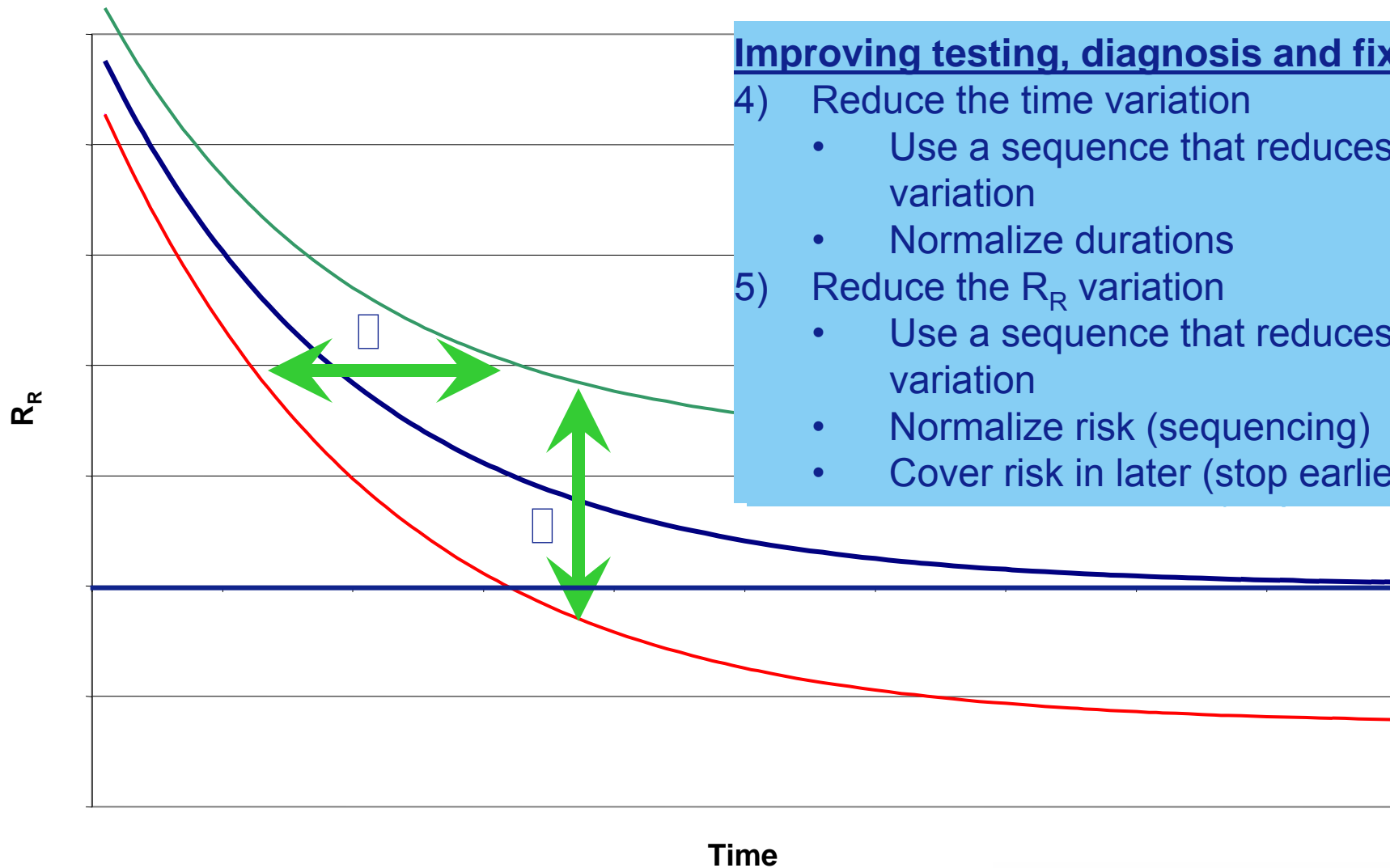


Improving testing, diagnosis and fixing

- 1) Reduce the initial risk
 - Higher product quality
 - Model-based integration
- 2) Improve testing
 - Sequencing
 - Select a better sequence
 - Optimal sequencing
 - Improve coverage
 - Add test cases
 - Improve coverage
 - Execute testing faster
 - Automation
 - Model-based testing
 - Model-based diagnosis
 - Speed up fixing
- 3) Stop testing earlier



Product quality (risk) vs test duration (2/2)



Improving testing, diagnosis and fixing

- 4) Reduce the time variation
 - Use a sequence that reduces the variation
 - Normalize durations
- 5) Reduce the R_R variation
 - Use a sequence that reduces the variation
 - Normalize risk (sequencing)
 - Cover risk in later (stop earlier)



Problem statement and objective

- Problem:
 - Test-diagnose-fix (TDF) tasks are often considered in isolation, not as part of an integration and test sequence
 - Improvement techniques for test-diagnose-fix-tasks are often:
 - Applicable for components of a single discipline
 - Focused on improving one single aspect (coverage, automation, etc.)

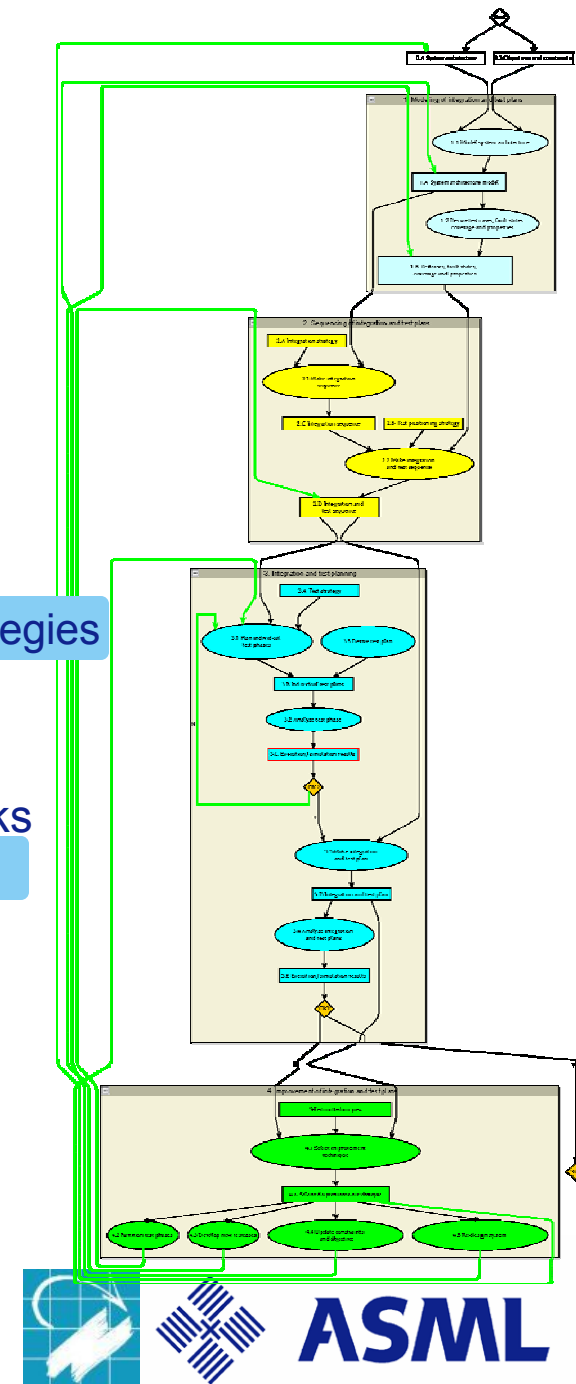
Objective: Improve test-diagnose-fix tasks such that the overall integration and test sequence is improved

- Methods:
 - Modeling of: system, test cases, integration and test tasks
 - Integration and test planning process (incl. strategies)
 - Analysis of (integration and) test plans
 - Improvement techniques



Integration and test planning

- Modeling
 - System architecture model
 - System test model
- Sequencing
 - Integration sequence **Integration strategies**
 - Test phase positioning **Test positioning strategies**
 - Integration and test sequencing
- Planning
 - Planning and optimizing individual TDF tasks **Test strategies**
- Improvement
 - Develop new test cases
 - Split up a test phase
 - Change the system architecture
- Execution
 - Performance analysis



Integration and testing

- Integration sequence:
 - Sequence of integration (assembly) and test-diagnose-fix tasks
 - dev – asm – tdf – das – cpy
 - Based on an integration strategy and test positioning strategy
 - Concurrent engineering, risk-based
- Test-diagnose-fix task (TDF task)
 - Sequence of test, diagnose and fix tasks
 - Based on a test strategy:
 - Test sequencing method
 - Test stop criteria
 - Test process configuration



Contents

- Introduction
- Problem statement and objective
- Integration and test planning
- Current research: methods and results
- Detailed method: Next-best-test-case algorithm
- Future opportunities
- Conclusions



Current research: methods

- Sequencing: AND/OR graph search algorithm
- (integration and) test sequence analysis: simulator
 - Test sequencing: risk-based, IG, random, ordered, AO, etc.
 - Test process configurations: test first-fix later, interleaved T-D-F, parallel T-D-F
 - Test stop criteria
 - System test model and integration and test sequence
- Integration and test plan improvement
 - Use another test strategy
 - TDF task partitioning: hypergraph partitioning algorithm
 - Next-best-test-case: IG based algorithm
 - Architectural guidelines



Current research: methods and results

- Sequencing: AND/OR graph search algorithm ~10% - ~40% improvement (duration)
- (integration and) test sequence analysis: simulator
 - Test sequencing: risk-based, IG, random, ordered, AO, etc.
 - Test process configurations: test first-fix later, interleaved T-D-F, parallel T-D-F
 - Test stop criteria
 - System test model and integration ~30% - ~70% improvement (duration) (case dependent)
- Integration and test plan improvement
 - Use another test strategy
 - TDF task partitioning: hypergraph partitioning algorithm
 - Next-best-test-case: IG based algorithm ~30% improvement (duration) at ~30% additional cost
 - Architectural guidelines

To be defined



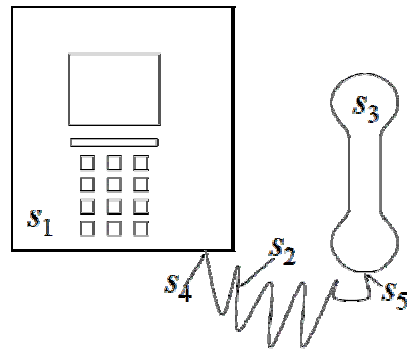
Contents

- Introduction
- Problem statement and objective
- Integration and test planning
- Current research: methods and results
- Detailed method: Next-best-test-case algorithm
- Future opportunities
- Conclusions



Next-best-test-case algorithm – some detail

- Goal: Given a test set, determine the coverage of the next best test case that could be developed
- Input: System test model



| S / T | t_0 | t_1 | t_2 | t_3 | t_4 | t_5 | P | I |
|---------|-------|-------|-------|-------|-------|-------|-----|-----|
| s_1 | 1 | 1 | 0 | 0 | 1 | 0 | 0% | 1 |
| s_2 | 1 | 0 | 1 | 0 | 1 | 1 | 8% | 1 |
| s_3 | 1 | 0 | 0 | 1 | 0 | 1 | 8% | 1 |
| s_4 | 1 | 0 | 0 | 0 | 1 | 0 | 8% | 1 |
| s_5 | 1 | 0 | 0 | 0 | 0 | 1 | 70% | 1 |
| s_6 | 1 | 0 | 0 | 0 | 0 | 1 | 80% | 1 |
| IG | 0.26 | 0.40 | 0.40 | 0.40 | 0.76 | 0.29 | | |

- Objective function: information gain

$$IG(j) = -(p_p(j) \log_2 p_p(j) + p_f(j) \log_2 p_f(j))$$



Details: Next-best-test-case algorithm 1/3

| <i>S / T</i> | <i>t</i> ₀ | <i>t</i> ₁ | <i>t</i> ₂ | <i>t</i> ₃ | <i>t</i> ₄ | <i>t</i> ₅ | <i>P</i> | <i>I</i> |
|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|-----------------------|----------|----------|
| <i>s</i> ₁ | 1 | 1 | 0 | 0 | 1 | 0 | 0% | 1 |
| <i>s</i> ₂ | 1 | 0 | 1 | 0 | 1 | 1 | 8% | 1 |
| <i>s</i> ₃ | 1 | 0 | 0 | 1 | 0 | 1 | 8% | 1 |
| <i>s</i> ₄ | 1 | 0 | 0 | 0 | 1 | 0 | 8% | 1 |
| <i>s</i> ₅ | 1 | 0 | 0 | 0 | 0 | 1 | 70% | 1 |
| <i>s</i> ₆ | 1 | 0 | 0 | 0 | 0 | 1 | 80% | 1 |
| <i>IG</i> | 0.26 | 0.40 | 0.40 | 0.40 | 0.76 | 0.29 | | |

- Assumptions:
 - A test covers a set of fault states: signature
 - A good test is a test with a 50% failure probability
- Basic (naïve) algorithm:
 - Generate all possible test signatures: i.e. all possible combinations of fault states (2^l-1)
 - Remove signatures already present in the test set T
 - Calculate the IG for all signatures
 - Add the signature with max(IG) to the test set T



Details: Next-best-test-case algorithm 2/3

NBTC(D):

- For all signatures in $\text{theta}(S) \setminus \emptyset$: determine $\text{IG}(\text{signature}, D.S_C)$
- return signature with $\max(\text{IG}(\text{signature}, D.S_C))$

$\text{theta}(S)$:

- Generate all permutations of s in S

ClusteredNBTC(D):

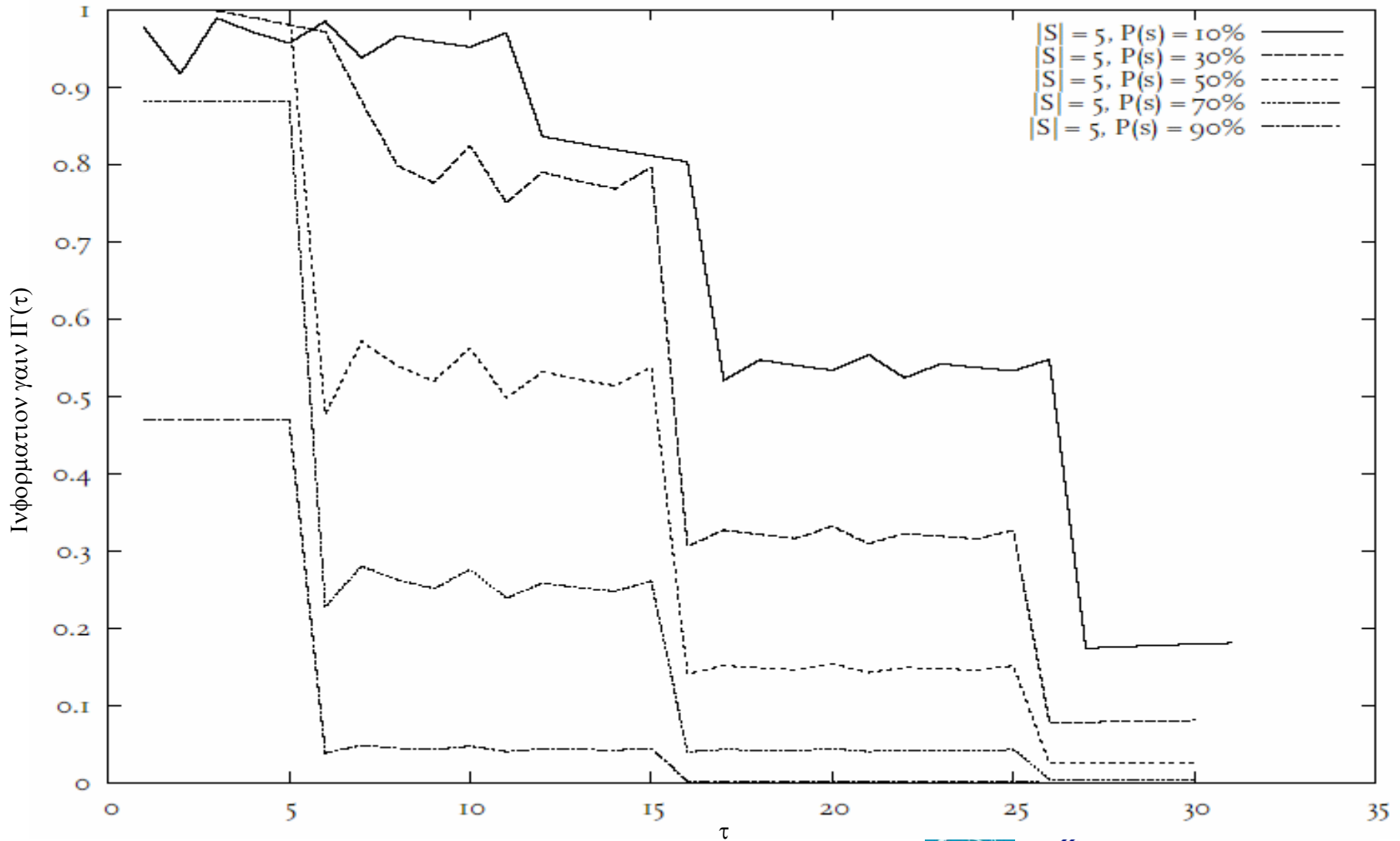
- If $|S| > \text{maxS}$:
 - $(\text{signature}', \text{signature}_{\text{clus}}) := \text{uncluster}(\text{NBTC}(\text{cluster}(D)))$
 - $\text{signature}_{\text{sub}} := \text{ClusteredNBTC}(\text{remove_clean}(\text{signature}' \setminus \text{signature}_{\text{clus}}))$
 - return $\text{signature}' \setminus (\text{signature}_{\text{clus}} \Delta \text{signature}_{\text{sub}})$
- If $|S| \leq \text{maxS}$: return $\text{NBTC}(D)$

$\text{IG}(\text{signature}, S_C)$:

- IG (entropy) taking into account signature of previous test cases S_C



Details: Next-best-test-case example 3/3



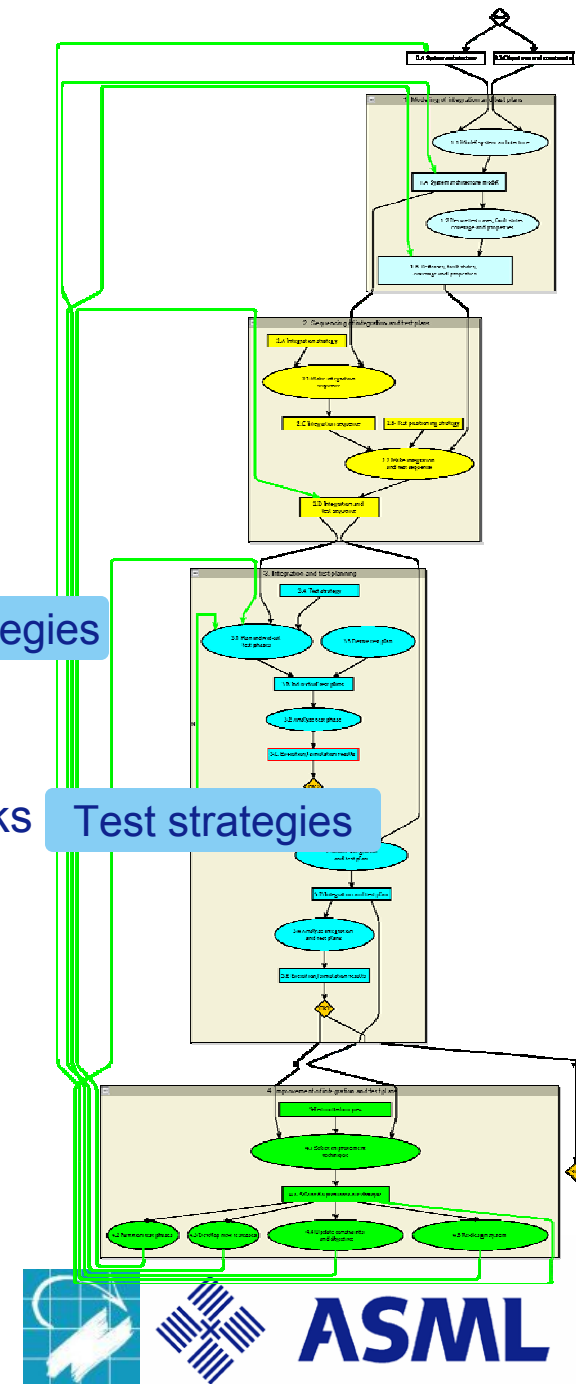
Conclusions

- The next-best-test-case algorithm can be used to calculate the optimal coverage for a new test case
- General conclusions
 - The integration and test sequence should be taken into account when test-diagnose-fix tasks are analyzed and improved
 - A method to create and analyze integration and test sequences has been developed
 - This method uses integration strategies, test positioning strategies and test strategies to obtain an integration and test sequence
 - Several techniques that improve integration and test sequences have been developed



Future opportunities

- Modeling
 - System architecture model
 - System test model
- Sequencing
 - Integration sequence **Integration strategies**
 - Test phase positioning **Test positioning strategies**
 - Integration and test sequencing
- Planning
 - Planning and optimizing individual TDF tasks **Test strategies**
- Improvement
 - Develop new test cases
 - Split up a test phase
 - Change the system architecture
- Execution
 - Performance analysis





ASML

Integration and test strategies

Current research and future opportunities



Tangram:
System Integration and Test Symposium

October 31st, 2007
Eindhoven, Netherlands

