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
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
- 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
- (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 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

~10% - ~40% improvement (duration)

~30% - ~70% improvement (duration) (case dependent)

~30% improvement (duration) at ~30% additional cost

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

- 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

- 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 \(\text{max}(\text{IG})\) to the test set \(T\)
Details: Next-best-test-case algorithm 2/3

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

\( \theta(S) \):
- Generate all permutations of \( s \) in \( S \)

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

IG(signature, \( S_C \)):
- IG (entropy) taking into account signature of previous test cases \( S_C \)
Details: Next-best-test-case example 3/3

- **Example system test models**
- Five fault states, 10% failure probability each
- Five fault states, 30% failure probability each
- Five fault states, 50% failure probability each

![Graph showing information gain over time for different fault states with failure probabilities of 10%, 30%, 50%, 70%, and 90%](graph.png)
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
  - Test phase positioning
  - Integration and test sequencing
- Planning
  - Planning and optimizing individual TDF tasks
- Improvement
  - Develop new test cases
  - Split up a test phase
  - Change the system architecture
- Execution
  - Performance analysis
Integration and test strategies

Current research and future opportunities