Modeling and Analysis for Software Installation Testing
by Jerry Gao, Karen Kwok and Todd Fitch

San Jose State University

Presented by: Jerry Zeyu Gao
San Jose State University
Email: jerrygao@email.sjsu.edu
URL:/www.engr.sjsu.edu/gaojerry
June, 2007

Presentation Outline

• Introduction
• Understanding Software Installation
• Software Installation Test Models
• Test Criteria and Metrics
• Strategic Test Plan and Analysis
• Test Planning and Using Weighted Test Sequences
• Related Work
• Conclusion and Future Work
Introduction

Software installation testing
- one of the important types of system testing
- one of the most complicated types of testing to plan and execute

In past years, researchers have not made enough effort to tackle the related issues and challenges in software installation testing.

According to QA and test engineers in the real world, there are the following challenges in software installation validation:

- A software product should be installed on a diverse system environment with many different configurations.
- A software product installation should be validated under various system running conditions.
- Software installation testing needs to be performed under a very tight schedule without well-defined test models, test criteria, and tools.

Major Problem:
- Many software makers experience their highest abandonment rates during the installation phase where users are giving up before even using the product

The current practice in the real world is summarized below:

- Using various ad-hoc approaches
- Lack well-defined test models, standards, and test coverage criteria
- Design test cases manually using an ad-hoc approach without models and criteria
- Most test executions are done manually
- Apply some GUI-based executable test scripts generated in ad-hoc approach.
- Installation test planning is carried out using a fully populated matrix.
Introduction

The Major Needs:

- A well-defined software installation test process with cost-effective test criteria and standards,
- Well-defined test models for software installation planning, testing and analysis to model and present:
  - Diverse system environments and configurations
  - Various system running conditions
  - Validate system installation functions,
- Well-defined metrics measuring and predicting test complexity, costs, and coverage.
- Cost-effective systematic test generation methods and tools to support automatic software installation testing.

Understanding Software Installation

What is software installation testing?

Its major purpose is to validate the given software product to see if it can be correctly installed in a specified system environment with proper system configurations and running conditions.

Its goal is to demonstrate the installation functions and behaviors work correctly.

The major focus of software installation validation is to answer the following questions:

- Can the software be properly installed on all specified system configurations?
- On the specified system configuration environment, can the software be successfully installed under each of the validated running conditions?
- Does the software demonstrate that its installation functions and behaviors behave correctly?
Understanding Software Installation

A software installation test process must include the following steps:

1. Understand, identify, and document all of the system configurations and possible environment settings.
2. Understand, identify, and document all of the system installation conditions for each configured system environment.
3. Plan software installation testing by:
   - Identifying and selecting the right test standards and test criteria.
   - Defining tasks and a working schedule.
   - Selecting a cost-effective test strategy by focusing on the important and popular system configurations and running conditions.
   - Selecting (and adopting) a proper installation test tool.
4. Identify and design installation test cases and develop automated test scripts.
5. Execute the specified installation tests and report results.
Software Installation Test Model

We have proposed a model, known as a semantic tree model, which is formally defined as 3-tuple = (N, E, R), where

- N is a set of tree nodes. Three types of nodes exist: a) a single root node, b) intermediate nodes, and c) leaf nodes.
- E is a set of tree edges. Each edge is a link, which connects a parent node and child node in a tree.
- R is a set of relations, and each item in R has a semantic label that presents one semantic relation between a parent node and its child nodes. Five types of semantic labels exist: OR, AND, NOT, NAND, and Select-1.
Software Installation Testing Space

Three derived models:
- System Environment Configuration (SEC) model
- System Installation Condition (SIC) model
- System Installation Function (SIF) model

A Sample of System Configuration Environment Model

OS Configuration

- EOR: P-Node must be provided and set up with only one of its exclusive parts, which are denoted as two child nodes. In other words, the two parts can't be set up at the same time.
- AND: P-Node must be provided without setting up its specific part, denoted as its only child node.
- NOT: P-Node must be provided without setting up its specific part, denoted as its only child node.
- NAND: P-Node must be provided without setting up its specific part, denoted as its only child node.
- Select-1: P-Node can be set up with any one of its child nodes.
A Sample of Installation Condition Model

<table>
<thead>
<tr>
<th>Relations</th>
<th>Semantics in an Installation Condition Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>EOR</td>
<td>P condition holds true only when one of its child conditions holds true.</td>
</tr>
<tr>
<td>AND</td>
<td>P condition holds true only when all of its child conditions hold true.</td>
</tr>
<tr>
<td>NAND</td>
<td>P condition holds true only when all of its child conditions are not hold true.</td>
</tr>
<tr>
<td>Select-1</td>
<td>P condition holds true only when none of its child conditions holds true.</td>
</tr>
</tbody>
</table>

A Sample of System Installation Function Model

<table>
<thead>
<tr>
<th>Relations</th>
<th>Semantics in a System Installation Function Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>EOR</td>
<td>P function is provided when any of its sub-functions (denoted as child nodes) is provided.</td>
</tr>
<tr>
<td>AND</td>
<td>P function is provided when all of its sub-functions (denoted as child nodes) are provided.</td>
</tr>
<tr>
<td>NOT</td>
<td>P function is provided without its specific sub-function (denoted as the only child node).</td>
</tr>
<tr>
<td>NAND</td>
<td>P function is supported without the support of some parts, denoted as the only child node.</td>
</tr>
<tr>
<td>Select-1</td>
<td>P function is provided when none of its sub-functions (denoted as child nodes) is provided.</td>
</tr>
</tbody>
</table>
A semantic spanning tree GSPT is a sub-tree of a given semantic tree GST, its holds the following properties:

- GSPT must include all parent nodes in GST.
- For each parent node Npi with an AND (or NAND) relation, GSPT must includes all of its child nodes and its links connected them.
- For each parent node Npi with an EOR relation, GSPT must include only one of its child nodes and the corresponding link.
- For each parent node Npi with a Select-1 relation must include only one of its child nodes and the corresponding link.
- For each parent node Npi with a NOT relation, GSPT must include the only child node and its corresponding link.
Test Criteria for System Environment Configuration

For a given SEC model $G_{SEC} = (N_{SEC}, E_{SEC}, R_{SEC})$, let’s define the system test criteria for the system installation environment and its various configurations.

Single System Environment Configuration Test Criterion:
• This test criterion only can be achieved when the given test case set $T_{SEC_i}$ has been exercised under the $SEC_i$ setting.

All System Environment Configuration Test Criteria:
• This test criterion only can be achieved when the given test case set $T_{SEC}$ have been exercised under all system environment configurations. This implies that all elements of SEC have been tested. They refer to all of the spanning trees of $G_{SEC}$.

System Environment and Configuration Test Complexity:
• For given software product $P$, its system environment and configuration test complexity ($SECTComplexity$) can be computed as follows:

$$SECTComplexity = \text{No. of elements in SEC} = |SEC|$$

$$= \text{No. of different semantic spanning trees in } G_{SEC}$$
Test Criteria for System Installation Conditions

For a given SIC model, \( G_{SIC} = (N_{SIC}, E_{SIC}, R_{SIC}) \), let’s define the test criteria and test metrics to address the diverse software running conditions under a given system environment and configuration.

**Single-System Installation Condition Test Criterion:**
- This test criterion only can be achieved when the given test case set \( T_{IS} \) has been exercised under a system installation condition (say SIC) when \( P \) is configured as SECi.

**All-System Installation Condition Test Criteria:**
- For a product \( P \) configured as SECi, this test criterion only can be achieved when the given test case set \( T_{IS} \) have been exercised under all system installation conditions in SIC. This implies that all elements of SIC have been tested. They actually are the spanning trees of \( G_{SIC} \).

**System Installation Condition Test Complexity:**
- For given software product \( P \) under a given configured system environment SECi, its system installation condition test complexity (\( \text{SICT}_{\text{Complexity}} \)) can be computed as follows:

\[
\text{SICT}_{\text{Complexity}} = \text{No. of items in SIC} = |\text{SEC}|
\]

\[
= \text{No. of different semantic spanning trees in } G_{SIC}
\]

Test Criteria for System Installation Functions

For a given product \( P \) and its software installation function model (SIF), \( G_{SIF} = (N_{SIF}, E_{SIF}, R_{SIF}) \), let’s define the function test criteria and test metrics to cover system installation functions under a given system installation condition SIC for a configured system environment SECi as follows.

**Leaf Node Function Test Criterion:**
- For any leaf node \( N_i \) in \( G_{SIF} \), this criterion is achieved when the given \( T_{IS} \) includes at least one test case, which exercise the corresponding function of \( N_i \).

**Adequate Leaf Node Function Test Criterion:**
- For any leaf node \( N_i \) in \( G_{SIF} \), this criterion is achieved when the given \( T_{IS} \) includes an adequate test set, which exercise the corresponding function of \( N_i \).

**Adequate Parent Node Function Test Criterion:**
- For any parent node \( N_{pi} \) in \( G_{SIF} \), including the root node and intermediate nodes, this criterion is achieved only when the given \( T_{IS} \) includes an adequate test set for each child node. In other words, all of its child nodes have achieved its adequate test criterion.
Test Complexity Analysis for Installation Functions

For any parent node $N_p$ of $G_{SI}$, its function test complexity can be computed based on its
relation with its child nodes and the test complexity of its child nodes.

Let $C_j$ stands for a child node of $N_p$.

- If its semantic relation with its child nodes is SELECT-1, then its complexity can be
  computed with the following.
  \[ N_p's \text{ FTComplexity} = (C_j's \text{ FTComplexity}) \]
  \( (5) \)
  Where $j = 1, \ldots, n$, $n$ is the number of its child nodes, and $C_j$ is a child node of $N_p$.

- If its semantic relation with its child nodes is EOR, then its complexity can be computed
  with the following.
  \[ N_p's \text{ FTComplexity} = (C_j's \text{ FTComplexity}) \]
  \( (6) \)
  Where $j = 1$ or 2, and $C_j$ is a child node of $N_p$.

- If its semantic relation with its child nodes is AND, then its complexity can be computed
  with the following.
  \[ N_p's \text{ FTComplexity} = (C_j's \text{ FTComplexity}) \]
  \( (7) \)
  Where $j = 1, \ldots, m$, $m$ is the number of its child nodes, and $C_j$ is a child node of $N_p$.

Test Complexity Analysis for System Installation Conditions

- Tcomplexity: 1+1+1+3 = 6
  Tcomplexity: 1+1+2 = 4
  Tcomplexity: 1+2+1 = 4
- SICTcomplexity: 3 * 3 * 4 = 36
Test Complexity Analysis for System Installation Functions

Test Planning Using Weighted Test Sequences

In many cases, engineers don’t have enough time and resources to cover adequate sets of installation function test cases on all of the identified configurations and installation conditions.

It is very useful to provide them with a systematic technique to find out the following ranked test sequences in the 3-dimensional software installation test space:

- A ranked validation sequence for diverse system environment configurations from the most important one to the least important one or an optional one.
- A ranked validation sequence for different system running conditions from the most common one to the rarely encountered one.
- A ranked validation test sequence for various system installation function tests.
Weight Calculations for Different Types of Nodes

\[
\begin{align*}
 w_k &= \frac{w_{k1}}{\sum_{j=1}^{m} w_{j1}} = \frac{w_{k1}}{w_{11} + \ldots + w_{m1}}, \text{ where } j = 1, \ldots, m
\end{align*}
\]  

Weight Classification Levels

<table>
<thead>
<tr>
<th>Weight Level</th>
<th>System Environment Configurations</th>
<th>System Running Conditions</th>
<th>System Installation Functions</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>Most Important Configurations</td>
<td>Important System Conditions</td>
<td>Important Installation Functions</td>
</tr>
<tr>
<td>4</td>
<td>Required Configurations</td>
<td>Common Conditions</td>
<td>Commonly Used Functions</td>
</tr>
<tr>
<td>3</td>
<td>Necessary Configurations</td>
<td>Required Conditions</td>
<td>Required Functions</td>
</tr>
<tr>
<td>2</td>
<td>Optional Configurations</td>
<td>Non-Common Conditions</td>
<td>Unpopular Functions</td>
</tr>
<tr>
<td>1</td>
<td>Not Supported Configurations</td>
<td>Rarely Encountered Conditions</td>
<td>Optional Functions</td>
</tr>
</tbody>
</table>
A Ranking Method for Test Sequences

The basic idea:
To rank each leaf node in a weighted semantic tree model.
Based on these rankings, we can find out the ranked validation test sequence using the following method based on a given weighted semantic tree model.

Assume \( G_{IT} = (N, E, R, W) \) is a given weighted semantic tree model.

Any leaf node \( N_{li} \) in \( N \) must have only one path \( P = e_1, \ldots, e_k \) to the root node. Since each edge on \( P \) has its own weighted ranking value, the weighted ranking of \( N_{li} \) can be defined as:

\[
\text{Rank}(N_{li}) = \sum_{i=1}^{k} w_{li}^i \cdot (10^{-i})
\]

Where \( w_{li}^i \) refers to the weighted ranking value for \( e_i \).

A Ranking Example for Software Configurations

![Diagram of a weighted semantic tree model with rankings for different configurations.](diagram.png)
A Ranking Method for Test Sequences

For any given weighted semantic tree \( G_{ET} = (N, E, R, W) \), assume \( G_{ET-SP} = (N_{SP}, E_{SP}, R, W_{SP}) \) to present a spanning tree.

According to the properties of the semantic tree, \( N_{SP} \) is a subset of \( N \), \( E_{SP} \) is a subset of \( E \), \( R \) is the same set of \( R \) in \( G_{ET} \), and \( W_{SP} \) is a subset of \( W \) in \( G_{ET} \).

The total ranking of \( G_{ET-SP} \) can be defined below.

\[
\text{Total-Rank} (G_{ET-SP}) = \sum \text{Rank} (N_{Lk}), k = 1, \ldots m \quad (11)
\]

Where \( N_{L1}, N_{L2}, \ldots, N_{Lm} \) are leaf nodes in \( N_{SP} \).

Using the total rankings of weighted spanning trees of a given semantic model, we can sort out a weighted validation sequence as a partial order among various system configurations and system running conditions.

This provides installation test engineers with a very useful guideline to conduct their testing activities.
A Semantic Tree With Rankings

A Semantic Spanning Tree With Rankings
Conclusion and Future Work

Summary:
This paper addresses the needs in modeling and analysis of software installation validation.
- A model-based approach is used to support systematic modeling and analysis for software installation testing.
- Based on a semantic tree model, a set of well-defined test criteria and complexity metrics are defined to support cost-effective test measurement and coverage analysis.

They are useful to help engineers to automatically identify and validate:
   a) system environments and configurations
   b) various installation conditions, and
   c) installation functional features.

Future extension of this research includes two parts:
• Develop more detailed product-oriented test cost metrics to support more effective strategic test planning and measurement.
• Developing a software installation automation tool based on the proposed models to support test modeling and analysis, test generation, and test coverage analysis.