Integration Testing
Functional Decomposition Based

Chapter 13
Integration Testing

- Test the interfaces and interactions among separately tested units

- Three different approaches
  - Based on functional decomposition
  - Based on call graphs
  - Based on paths
Functional Decomposition

- Create a functional hierarchy for the software
- Problem is broken up into independent task units, or functions
- Units can be run either
  - Sequentially and in a synchronous call-reply manner
  - Or simultaneously on different processors
- Used during planning, analysis and design
# SATM Units

<table>
<thead>
<tr>
<th>Unit</th>
<th>Level</th>
<th>Name</th>
<th>Unit</th>
<th>Level</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>SATM system</td>
<td>14</td>
<td>1.3.1</td>
<td>Screen door</td>
</tr>
<tr>
<td>A</td>
<td>1.1</td>
<td>Device sense &amp; control</td>
<td>15</td>
<td>1.3.2</td>
<td>Key sensor</td>
</tr>
<tr>
<td>D</td>
<td>1.1.1</td>
<td>Door sense &amp; control</td>
<td>16</td>
<td>1.4.1</td>
<td>Validate card</td>
</tr>
<tr>
<td>2</td>
<td>1.1.1.1</td>
<td>Get door status</td>
<td>17</td>
<td>1.4.2</td>
<td>Validate PIN</td>
</tr>
<tr>
<td>3</td>
<td>1.1.1.2</td>
<td>Control door</td>
<td>18</td>
<td>1.4.2.1</td>
<td>Get PIN</td>
</tr>
<tr>
<td>4</td>
<td>1.1.1.3</td>
<td>Dispense cash</td>
<td>19</td>
<td>1.4.3.1</td>
<td>New transaction request</td>
</tr>
<tr>
<td>E</td>
<td>1.1.2</td>
<td>Slot sense &amp; control</td>
<td>20</td>
<td>1.4.3.2</td>
<td>Print receipt</td>
</tr>
<tr>
<td>5</td>
<td>1.1.2.1</td>
<td>Watch card slot</td>
<td>21</td>
<td>1.4.3.3</td>
<td>Post transaction local</td>
</tr>
<tr>
<td>6</td>
<td>1.1.2.2</td>
<td>Get deposit slot status</td>
<td>22</td>
<td>1.4.4</td>
<td>Manage transaction</td>
</tr>
<tr>
<td>7</td>
<td>1.1.2.3</td>
<td>Control card Roller</td>
<td>23</td>
<td>1.4.4.1</td>
<td>Get transaction type</td>
</tr>
<tr>
<td>8</td>
<td>1.1.2.4</td>
<td>Control Envelope Roller</td>
<td>24</td>
<td>1.4.4.2</td>
<td>Get account type</td>
</tr>
<tr>
<td>9</td>
<td>1.1.2.5</td>
<td>Read card strip</td>
<td>25</td>
<td>1.4.4.3</td>
<td>Report balance</td>
</tr>
<tr>
<td>10</td>
<td>1.2</td>
<td>Central bank comm.</td>
<td>26</td>
<td>1.4.4.4</td>
<td>Process deposit</td>
</tr>
<tr>
<td>11</td>
<td>1.2.1</td>
<td>Get PIN for PAN</td>
<td>27</td>
<td>1.4.4.5</td>
<td>Process withdrawal</td>
</tr>
<tr>
<td>12</td>
<td>1.2.2</td>
<td>Get account status</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>1.2.3</td>
<td>Post daily transactions</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>B</td>
<td>1.3</td>
<td>Terminal sense &amp; control</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Table 1: SATM Units and Abbreviated Names

<table>
<thead>
<tr>
<th>Unit</th>
<th>Level</th>
<th>Unit Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1.1</td>
<td>Device Sense &amp; Control</td>
</tr>
<tr>
<td></td>
<td>1.1.1</td>
<td>Door Sense &amp; Control</td>
</tr>
<tr>
<td></td>
<td>1.1.1.1</td>
<td>Get Door Status</td>
</tr>
<tr>
<td></td>
<td>1.1.1.2</td>
<td>Control Door</td>
</tr>
<tr>
<td></td>
<td>1.1.1.3</td>
<td>Dispense Cash</td>
</tr>
<tr>
<td>D</td>
<td>1.1.2</td>
<td>Slot Sense &amp; Control</td>
</tr>
<tr>
<td></td>
<td>1.1.2.1</td>
<td>WatchCardSlot</td>
</tr>
<tr>
<td></td>
<td>1.1.2.2</td>
<td>Get Deposit Slot Status</td>
</tr>
<tr>
<td></td>
<td>1.1.2.3</td>
<td>Control Card Roller</td>
</tr>
<tr>
<td></td>
<td>1.1.2.5</td>
<td>Control Envelope Roller</td>
</tr>
<tr>
<td>B</td>
<td>1.2</td>
<td>Central Bank Comm.</td>
</tr>
<tr>
<td></td>
<td>1.2.1</td>
<td>Get PIN for PAN</td>
</tr>
<tr>
<td></td>
<td>1.2.2</td>
<td>Get Account Status</td>
</tr>
<tr>
<td></td>
<td>1.2.3</td>
<td>Post Daily Transactions</td>
</tr>
<tr>
<td>C</td>
<td>1.3</td>
<td>Terminal Sense &amp; Control</td>
</tr>
<tr>
<td></td>
<td>1.3.1</td>
<td>Screen Driver</td>
</tr>
<tr>
<td></td>
<td>1.3.2</td>
<td>Key Sensor</td>
</tr>
<tr>
<td></td>
<td>1.4</td>
<td>Manage Session</td>
</tr>
<tr>
<td></td>
<td>1.4.1</td>
<td>Validate Card</td>
</tr>
<tr>
<td></td>
<td>1.4.2</td>
<td>Validate PIN</td>
</tr>
<tr>
<td></td>
<td>1.4.3.1</td>
<td>GetPIN</td>
</tr>
<tr>
<td></td>
<td>1.4.3.2</td>
<td>New Transaction Request</td>
</tr>
<tr>
<td></td>
<td>1.4.3.3</td>
<td>Print Receipt</td>
</tr>
<tr>
<td></td>
<td>1.4.3.4</td>
<td>Post Transaction Local</td>
</tr>
<tr>
<td></td>
<td>1.4.3.5</td>
<td>Manage Transaction</td>
</tr>
<tr>
<td></td>
<td>1.4.4</td>
<td>Process Deposit</td>
</tr>
<tr>
<td></td>
<td>1.4.4.1</td>
<td>Get Transaction Type</td>
</tr>
<tr>
<td></td>
<td>1.4.4.2</td>
<td>Get Account Type</td>
</tr>
<tr>
<td></td>
<td>1.4.4.3</td>
<td>Report Balance</td>
</tr>
<tr>
<td></td>
<td>1.4.4.4</td>
<td>Process Withdrawal</td>
</tr>
</tbody>
</table>

SATS functional decomposition tree
Decomposition-based integration strategies

- What are the decomposition-based integration strategies?
Decomposition-based integration strategies – 2

- Top-down
- Bottom-up
- Sandwich
- Big bang
Big bang integration process

- What is the big bang integration process.
Big bang integration process – 2

- All units are compiled together
- All units are tested together
Big bang integration issues

- What are the issues (advantages and drawbacks)?
Failures will occur!

- No clues to isolate location of faults
- No stubs or drivers to write
Top-down integration

- What is the top-down integration process?
Top-Down integration example

- **Top Subtree**
  - Sessions 1-4

- **Second Level Subtree**
  - Sessions 5-8

- **Bottom Level Subtree**
  - Sessions 9-13
Top-Down integration process

- **Strategy**
  - Focuses on testing the top layer or the controlling subsystem first
    - The main, or the root of the call tree

- **General process is**
  - To gradually add more subsystems that are referenced/required by the already tested subsystems when testing the application
  - Do this until all subsystems are incorporated into the test
Top-Down integration process – 2

- **Stubs** are needed to do the testing

- A program or a method that simulates the input-output functionality of a missing subsystem by answering to the decomposition sequence of the calling subsystem and returning back simulated data
Top-Down integration issues

- What are the issues?
Top-Down integration issues – 2

- Writing stubs can be difficult
  - Especially when parameter passing is complex.
  - Stubs must allow all possible conditions to be tested

- Possibly a very large number of stubs may be required
  - Especially if the lowest level of the system contains many functional units
Top-Down integration issues – 3

- One solution to avoid too many stubs
  - Modified top-down testing strategy
  - Test each layer of the system decomposition individually before merging the layers
  - Disadvantage of modified top-down testing
    - Both stubs and drivers are needed
Bottom-up integration

- What is the bottom-up integration process?
Bottom-up integration example

Bottom Level Subtree
Sessions 1-5

Second Level Subtree
Sessions 6-9

Top Subtree
Sessions 10-13
Bottom-Up integration process

- Bottom-Up integration strategy
  - Focuses on testing the units at the lowest levels first
  - Gradually includes the subsystems that reference/require the previously tested subsystems
  - Do until all subsystems are included in the testing
Drivers are needed to do the testing

- A driver is a specialized routine that passes test cases to a subsystem

- Subsystem is not everything below current root module, but a sub-tree down to the leaf level
Bottom-up integration issues

- What are the issues?
Bottom-Up Integration Issues

- Not an optimal strategy for functionally decomposed systems
  - Tests the most important subsystem (user interface) last
- More useful for integrating object-oriented systems
- Drivers may be more complicated than stubs
- Less drivers than stubs are typically required
Sandwich integration

- What is the sandwich integration process?
Sandwich integration example

Sandwich 1
Sessions 1-3

Sandwich 2
Sessions 4-13

Sandwich 3
Sessions 14-15
Sandwich integration process

- Combines top-down strategy with bottom-up strategy
  - Doing big bang on a subtree
Sandwich integration issues

- What are the issues?
Sandwich integration issues – 2

- Less stub and driver development effort
- Added difficulty in fault isolation
Integration test session

- A session is a test suite that tests one edge in the tree
  - Each session tests the combining of two parts
    \( \text{#sessions} = \text{#edges} \)
  - This is different from the textbook
    \( \text{#sessions} = \text{#nodes} - \text{#leaves} + \text{#edges} \\
    = 2 \text{#edges} - \text{#leaves} + 1 \)

  Alternately

    \( \text{#sessions} = \text{#internal_nodes} + \text{#edges} \)
Integration work numbers

- For top-down integration
  - \(\#\text{nodes} - 1 = \#\text{edges}\) stubs are needed

- For bottom-up integration
  - \(\#\text{nodes} - \#\text{leaves} = \#\text{internal\_nodes}\) drivers are needed

- The number integrated units for top-down and bottom-up
  \[
  \#\text{integrated\_units} = \#\text{internal\_nodes}
  \]
Integration work numbers

- For SATM have 32 integration test sessions
  - Correspond to 32 separate sets of test cases
- For top-down integration
  - 32 stubs are needed
- For bottom-up integration
  - 10 drivers are needed
- For top-down and bottom-up
  - 10 integration units
Decomposition-based drawback

- What is the major drawback of decomposition-based integration?
Decomposition-based drawback – 2

- It is functionally based
  - Has the problems of all functional testing

- How do we overcome the problems?
Decomposition-based drawback – 3

- It is functionally based
  - Has the problems of all functional testing

- How do we overcome the problems?
  - Move to structural-based testing