2006 exam solution
Reuse Policy

• Defining a policy for acquiring and validating components from the outside.

• Putting in place a corporate component library.

• Updating the software process model to account for a generalization activity.

• Devising a reward structure for developers who contribute to the corporate library.
Active data structures

Treating data structures as machines with a state, characterized by a current position, e.g. through a cursor that can be moved back and forth.
Alternative way: passive data structures
Treat the structure as just a collection.

Search and insertion in passive style:

```plaintext
pos := l.search (v)
l.insert_after (w, pos)
```

Search and insertion in active style:

```plaintext
l.search (v) -- Command, not query
l.put_right (v)
```
# Active vs. passive data structure

<table>
<thead>
<tr>
<th></th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Active</td>
<td>Simplifies interface</td>
<td>Restricted iteration method</td>
</tr>
<tr>
<td></td>
<td>Simplifies implementation</td>
<td>Don’t go well with concurrent access</td>
</tr>
<tr>
<td></td>
<td>More scalable</td>
<td></td>
</tr>
<tr>
<td>Passive</td>
<td>Various traversal methods through external iterators</td>
<td>More heavy interface</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Code duplication</td>
</tr>
</tbody>
</table>
Test case design – using partition testing

For every type of arguments, define its partitions. For example:

- `%0, ‘a’, ‘Z’` for CHARACTER
- `0, minimum, maximum, integer larger than the string length` for INTEGER
- `Void, some empty string, some random string` for STRING

And then after invocation of the tested feature, check state of the target object by checking the validity of queries such as `item, safe_item`. 
Test case evaluation

• Measure coverage

• Introduce mutation
Visitor pattern vs. visitor component

<table>
<thead>
<tr>
<th></th>
<th>Pros</th>
<th>Cons</th>
</tr>
</thead>
<tbody>
<tr>
<td>Visitor pattern</td>
<td>Performance</td>
<td>Hard to add new types to be visited</td>
</tr>
<tr>
<td>Visitor component</td>
<td>Off-the-shelf reusability</td>
<td>Slower</td>
</tr>
<tr>
<td></td>
<td>Easy to add new types</td>
<td>Bigger clients</td>
</tr>
</tbody>
</table>
Visitor component

visitor.extend (agent process_string_general)
visitor.extend (agent process_plain_text_file)
visitor.extend (agent process_string_32)
visitor.extend (agent process_file)
visitor.extend (agent process_any)

visitor.visit (a_string_32)

One topological sort:
1 2 3 4 5