Compatibility Testing
using
Patterns-based Trace Comparison

Venkatesh-Prasad Ranganath, Kansas State University, USA
Pradip Vallathol, University of Wisconsin-Madison, USA
Pankaj Gupta, Microsoft Corporation, USA

29th IEEE/ACM International Conference on Automated Software Engineering (ASE 2014)
September 15 - 19, 2014
Västerås, Sweden

Thanks to Microsoft Research (India) and USB team (at Microsoft) for supporting this effort.
Compatibility Testing: Syntactic Changes

**client.java**

```java
service.getProperty("name")
```

**service.java**

```java
void getProperty(PropertyId id) {
    ...
}
```
Compatibility Testing: Semantic Changes

client1.c
s.q = c;
f(&s);
s.q = c;
g(&s);

client2.c
s.q = c;
f(&s);
g(&s);

serviceV1.c
@pre s.q == c
void f(Record *s) {
    ...  
    // no changes to s.q
}

@pre s.q == c
void g(Record *s) {
    ...
}
Compatibility Testing: Semantic Changes

client1.c
s.q = c;
f(&s);
s.q = c;
g(&s);

client2.c
s.q = c;
f(&s);
g(&s);

Incorrect!!

serviceV2.c
@pre s.q == c
void f(Record *s) {
...
s.q = 0;
}

@pre s.q == c
void g(Record *s) {
...
}
Common Reasons for Semantic Incompatibilities

• Breaking semantic changes
• Observational dependences and influences
• Weak specifications
• Assumptions
Compatibility Testing of Windows USB drivers

When a USB 2.0 device is plugged into a USB 3.0 port on Win8, will USB 3.0 driver in Win8 behave similar to the USB 2.0 stack in Win7 (along both software and hardware interfaces)?
Why is it hard?

- Clean room implementation of USB 3 driver
- No part of USB 2 driver was reused
- Regression tests were insufficient
- Large testing surface
- Number of unique USB devices
- Possibilities in USB protocol
- Multiple layers of variability
- Device drivers, Controllers, & ASIC in devices
Compatibility Testing of Windows USB drivers

When a USB 2.0 device is plugged into a USB 3.0 port on Win8, will USB 3.0 driver in Win8 behave similar to the USB 2.0 stack in Win7 (along both software and hardware interfaces)?
Compatibility Testing using Patterns-based Trace Comparison

IOCTLType=URB_FUNCTION_BULK_OR_INTERRUPT_TRANSFER(0x09) && IoCallDriverReturn && IoCallDriverReturn.irql=2 && IoCallDriverReturn.status=0xC000000E

DispatchIrp *forward alternates with* IrpCompletion && PreIoCompleteRequest *when*

IOCTLType=IRP_MJ_PNP(0x1B), IRP_MN_START_DEVICE(0x00), irpID=SAME, and IrpSubmitDetails.irp.ioStackLocation.control=SAME
USB 2.0 driver completed isochronous requests at DISPATCH_LEVEL IRQ while USB 3.0 driver completed similar request at PASSIVE_LEVEL IRQ.

```c
21 = fopen("passwd.txt", "r")
```
USB 3.0 driver failed to communicate the corresponding interface when serving a request to select a configuration of a device. *interfaceHandle* attribute remained unchanged.

```
(h!=0 && h=fopen) .... fclose(h)
```

```
21=fopen(, “r”) .... fclose(21)
```

```
21=fopen .... fclose(21)
```

```
h=fopen .... fclose(h)
```

```
21 = fopen(“passwd.txt”, “r”) .... fclose(21)
```
What is reported?

Presence of previously unobserved patterns

$$USB3(dev_k) - \bigcup_i USB2(dev_i)$$

Absence of previously observed patterns

$$\bigcup_i USB2(dev_i) - USB3(dev_k)$$

Comment: This should be intersection
Is it effective?

We detected 14 unique bugs (25 bugs) by testing 14 devices with regression tested USB 3.0 driver.
Is it expensive?

- Worst case mining time was 115 minutes
- Worst case diffing time was 48 minutes
- *Non-empty* reports analysis took ~2 hours
- Few reports required 24 hours
Domain Knowledge

# of attributes: 361
# of ignored attributes: 108 (361 - 108 = 253)
# of necessary attributes: 29 (253 - 29 = 224)
# of NULL abstracted attributes: 23
# of unquantifiable attributes: 75
# of quantifiable attributes: 150
# of data flows: 17 (between 26 attributes)
### User Feedback

<table>
<thead>
<tr>
<th>Device</th>
<th>Known</th>
<th>Detected</th>
<th>Simplified</th>
<th>Compacted</th>
<th>Reported</th>
<th>False +ve</th>
<th>Structural</th>
<th>Temporal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>9844</td>
<td>932</td>
<td>478</td>
<td>478</td>
<td>11 + 454</td>
<td>6 / 9</td>
<td>4 / 4</td>
</tr>
<tr>
<td>2*</td>
<td>932</td>
<td>2545</td>
<td>121</td>
<td>63</td>
<td>15</td>
<td>0 + 11</td>
<td>1 / 1</td>
<td>1 / 3</td>
</tr>
<tr>
<td>3</td>
<td>965</td>
<td>743</td>
<td>41</td>
<td>21</td>
<td>4</td>
<td>1 + 0</td>
<td>0 / 0</td>
<td>1 / 3</td>
</tr>
<tr>
<td>4</td>
<td>965</td>
<td>1372</td>
<td>67</td>
<td>34</td>
<td>2</td>
<td>1 + 1</td>
<td>0 / 0</td>
<td>0 / 0</td>
</tr>
<tr>
<td>5*</td>
<td>2141</td>
<td>26118</td>
<td>1114</td>
<td>571</td>
<td>55</td>
<td>26 + 29</td>
<td>0 / 0</td>
<td>0 / 0</td>
</tr>
<tr>
<td>6</td>
<td>2141</td>
<td>26126</td>
<td>1054</td>
<td>541</td>
<td>0</td>
<td>0 + 0</td>
<td>0 / 0</td>
<td>0 / 0</td>
</tr>
<tr>
<td>7</td>
<td>2141</td>
<td>2320</td>
<td>84</td>
<td>44</td>
<td>0</td>
<td>0 + 0</td>
<td>0 / 0</td>
<td>0 / 0</td>
</tr>
<tr>
<td>8</td>
<td>2141</td>
<td>27804</td>
<td>1185</td>
<td>608</td>
<td>2</td>
<td>1 + 0</td>
<td>1 / 1</td>
<td>0 / 0</td>
</tr>
<tr>
<td>9</td>
<td>2141</td>
<td>34985</td>
<td>413</td>
<td>217</td>
<td>115</td>
<td>2 + 96</td>
<td>2 / 14</td>
<td>2 / 3</td>
</tr>
<tr>
<td>10</td>
<td>2141</td>
<td>51556</td>
<td>429</td>
<td>231</td>
<td>59</td>
<td>15 + 41</td>
<td>1 / 1</td>
<td>2 / 2</td>
</tr>
<tr>
<td>11</td>
<td>2141</td>
<td>695</td>
<td>35</td>
<td>18</td>
<td>0</td>
<td>0 + 0</td>
<td>0 / 0</td>
<td>0 / 0</td>
</tr>
<tr>
<td>12</td>
<td>2141</td>
<td>1372</td>
<td>67</td>
<td>34</td>
<td>0</td>
<td>0 + 0</td>
<td>0 / 0</td>
<td>0 / 0</td>
</tr>
<tr>
<td>13</td>
<td>2141</td>
<td>3315</td>
<td>122</td>
<td>72</td>
<td>24</td>
<td>19 + 4</td>
<td>1 / 1</td>
<td>0 / 0</td>
</tr>
<tr>
<td>14*</td>
<td>2141</td>
<td>9299</td>
<td>103</td>
<td>54</td>
<td>3</td>
<td>0 + 0</td>
<td>2 / 3</td>
<td>0 / 0</td>
</tr>
</tbody>
</table>
## Smart Presentation

<table>
<thead>
<tr>
<th>Device</th>
<th>Known</th>
<th>Detected</th>
<th>Simplified</th>
<th>Compacted</th>
<th>Reported</th>
<th>False +ve</th>
<th>Structural</th>
<th>Temporal</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>9844</td>
<td>932</td>
<td>478</td>
<td>478</td>
<td>11 + 454</td>
<td>6 / 9</td>
<td>4 / 4</td>
</tr>
<tr>
<td>2*</td>
<td>932</td>
<td>2545</td>
<td>121</td>
<td>63</td>
<td>15</td>
<td>0 + 11</td>
<td>1 / 1</td>
<td>1 / 3</td>
</tr>
<tr>
<td>3</td>
<td>965</td>
<td>743</td>
<td>41</td>
<td>21</td>
<td>4</td>
<td>1 + 0</td>
<td>0 / 0</td>
<td>1 / 3</td>
</tr>
<tr>
<td>4</td>
<td>965</td>
<td>1372</td>
<td>67</td>
<td>34</td>
<td>2</td>
<td>1 + 1</td>
<td>0 / 0</td>
<td>0 / 0</td>
</tr>
<tr>
<td>5*</td>
<td>2141</td>
<td>26118</td>
<td>1114</td>
<td>571</td>
<td>55</td>
<td>26 + 29</td>
<td>0 / 0</td>
<td>0 / 0</td>
</tr>
<tr>
<td>6</td>
<td>2141</td>
<td>26126</td>
<td>1054</td>
<td>541</td>
<td>0</td>
<td>0 + 0</td>
<td>0 / 0</td>
<td>0 / 0</td>
</tr>
<tr>
<td>7</td>
<td>2141</td>
<td>2320</td>
<td>84</td>
<td>44</td>
<td>0</td>
<td>0 + 0</td>
<td>0 / 0</td>
<td>0 / 0</td>
</tr>
<tr>
<td>8</td>
<td>2141</td>
<td>27804</td>
<td>1185</td>
<td>608</td>
<td>2</td>
<td>1 + 0</td>
<td>1 / 1</td>
<td>0 / 0</td>
</tr>
<tr>
<td>9</td>
<td>2141</td>
<td>34985</td>
<td>413</td>
<td>217</td>
<td>115</td>
<td>2 + 96</td>
<td>2 / 14</td>
<td>2 / 3</td>
</tr>
<tr>
<td>10</td>
<td>2141</td>
<td>51556</td>
<td>429</td>
<td>231</td>
<td>59</td>
<td>15 + 41</td>
<td>1 / 1</td>
<td>2 / 2</td>
</tr>
<tr>
<td>11</td>
<td>2141</td>
<td>695</td>
<td>35</td>
<td>18</td>
<td>0</td>
<td>0 + 0</td>
<td>0 / 0</td>
<td>0 / 0</td>
</tr>
<tr>
<td>12</td>
<td>2141</td>
<td>1372</td>
<td>67</td>
<td>34</td>
<td>0</td>
<td>0 + 0</td>
<td>0 / 0</td>
<td>0 / 0</td>
</tr>
<tr>
<td>13</td>
<td>2141</td>
<td>3315</td>
<td>122</td>
<td>72</td>
<td>24</td>
<td>19 + 4</td>
<td>1 / 1</td>
<td>0 / 0</td>
</tr>
<tr>
<td>14*</td>
<td>2141</td>
<td>9299</td>
<td>103</td>
<td>54</td>
<td>3</td>
<td>0 + 0</td>
<td>2 / 3</td>
<td>0 / 0</td>
</tr>
</tbody>
</table>
Lessons Learned

• If domain knowledge is available, use it
• If a feedback loop can be established, set it up
• Presentation matters
• Embrace the unorthodox
Limitations

• Detects a class of incompatibilities
Threats to Validity

- Generalization needs more experiments
- Effect of latent factors need to be studied
Key Takeaways

• An approach to compatibility testing via patterns-based trace comparison.

• The use of structural and temporal patterns as trace abstractions to enable software engineering and maintenance tasks.

• Of course, the lessons learned :)