Increasing Client-side Confidence in Remote Component Implementations

Ramesh Jagannathan
Paul A.G. Sivilotti
Dept. of Computer & Info. Science
The Ohio State University

Distributed Autonomous Peer-to-peer Systems

“computational tapestry”

Client-side Confidence in Remote Components

Testing Components

- Component interface
  - Behavioral specification
  - Abstract state
  - Both safety and progress properties
- From interface, automatically generate testing harness
  - Unit-testing
  - Monitors/records component behavior
  - Reports violations (and trace information)

Contrast Developer and Client-side Testing

- Developer-side:
  - Compare spec with actual behavior
  - Includes state transitions
- Client-side:
  - Compare spec with observed behavior
  - Only visible behavior matters
  - Messages sent & received
  - There must exist some justification for behavior

Testing Remote Components (Challenges)

- No access to implementation
  - Executable resides on remote machine
  - Truly black box
- Heterogeneity of clients
  - Different clients may have different:
    - Confidence requirements / priorities
    - Performance requirements / priorities
- Multiple threads of control
  - Concurrent invocations of shared components
  - Precondition paradox

Example: Auctioneer

```java
interface Auctioneer {
    void bid (int amount); // submit bid
    void inc (int amount); // increase current // high bid
    int getBid (); // return value of current // high bid
};
```
**Example: Auctioneer**

```java
interface Auctioneer {
    state int price = 0;
    void bid (int amount);
        mod: price;
        post: price = max(price, amount);
    void inc (int amount);
        pre: amount > 0;
        post: price < price <= price + amount;
    int getBid () {
        post: getBid() = price;
    }
}
```

**A Simple Example**

![Diagram of a simple example involving a client and an Auctioneer component.]

**A Faulty Component: Unilateral Detection**

- Some faults can be detected by a single client.

![Diagram illustrating unilateral fault detection.]

**A Faulty Component: Collective Detection**

- Some faults can only be detected by a collection of clients.

![Diagram illustrating collective fault detection.]

**A Faulty Component: Limitation on Detection**

- Some faults cannot be detected even by a collection of clients.

![Diagram illustrating limitations in detection.]

**Solution: Checking Wrapper**

- Intercept messages to/from component.

![Diagram illustrating the addition of a checking wrapper.]

---

*Client-side Confidence in Remote Components*
**Design Issue I: Opacity**
- Component is a black box
  - Eg., value of current high bid is not known outside of Auctioneer implementation
- Observation:
  - Specification is in terms of abstract state
- Design decision:
  - Component must provide abstract state
  - Intercept all messages and compare values with current (and new) abstract state
  - Provided state is a "cover story" that justifies the observed messages

**Opacity Example I**

**Design Issue II: Privacy**
- Component does not trust the checking framework
  - Wrapper might leak private information (abstract state) to the clients
  - Eg., game of mastermind
- Observation:
  - Many different abstract states may justify the same visible behavior
- Design decision:
  - Component may provide a set of possible abstract states
  - Cardinality of set is decided by component

**Opacity Example II**

**Privacy Example I**
Design Issue III: Performance
- Checking abstract states incurs overhead
- Effect is amplified by multiple cover stories
- Observation:
  - Not all clients may want this level of checking
- Design decision:
  - Distinguish between different confidence levels
  - Selectively synchronize for high-confidence

Design Issue IV: Specifying Behavior
- Components exhibit autonomous reactive behavior
- Non-terminating
- Can initiate behavior (active)
- Observation
  - Ubiquity of interface description languages
- Design decision
  - Behavioral specifications given in temporal logic
    (next, transient, stable, ...)
  - IDL augmented with specification constructs

Next Property: P next Q
- Example: (A = 4) next (A = 4)

<table>
<thead>
<tr>
<th>Value of A</th>
<th>4</th>
<th>6</th>
<th>3</th>
<th>4</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>P</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
<tr>
<td>Q</td>
<td>✓</td>
<td>✓</td>
<td>✗</td>
<td>✗</td>
<td>✗</td>
</tr>
</tbody>
</table>

- Observation:
  - (P next Q) ⇒ [P ⇒ Q]

Example Specification
- Using pre- and post-conditions
  - void bid (int amount)
    - mod: price
      - post: price = max(price, amount)
  - Using next (quantified)
    - (∀i,j: (n_bid = i ∧ price = j) next
      - (n_bid = i+1 ⇒ price = max(j, amount)))
  - Using functional next
    - (i := n_bid, j := price) in
      - (n_bid = i ∧ price = j) next
      - (n_bid = i+1 ⇒ price = max(j, amount))

Enriched Interface Description
```java
interface Auctioneer {
  #pragma state int price;
  #pragma initially(price := 0);
  #pragma fn
  (i := n_bid, j := price) in
    (n_bid := i+1 ⇒ price := max(j, amount));
  ...
  void bid (int amount); // place a bid
  void inc (int amount); // increases the bid value
  int getBid();
}
```

Architecture of Framework
Pruning Abstract States

- Current set of possible states
  \[ S = \{s_1, s_2, \ldots, s_n\} \]
- Set of possible new states reported by component
  \[ T = \{t_1, t_2, \ldots, t_m\} \]
- Collection of next properties
  \[ N = \{n_1, n_2, \ldots, n_p\} \]

Removing a State

- Rule:
  A new state, \( t_j \), is valid if it has support from a predecessor in the old set, \( S \)

\[ \exists s_i \in S : P_j.s_i \Rightarrow Q.t_j \]

Removing a State II

- Rule:
  At least one predecessor must conform to all next properties for \( t_j \) to be valid

\[ \exists s_i \in S : (\forall k: 1 \leq k \leq p : P_k.s_i \Rightarrow Q_k.t_i) \]

Pruner Algorithm

for each new state \( t_j \)
for each old state \( s_i \)
  assume \( s_i \) supports \( t_j \)
  for each next property of the form \( (P_k \Rightarrow Q_k) \)
    if \( \neg (P_k.s_i \Rightarrow Q_k.t_j) \)
      then \( s_i \) does not support \( t_j \)
      endfor
  endfor
if no \( s_i \) was found to support \( t_j \)
  remove \( t_j \) from the new state set
endfor
if the new state set is empty, report violation

Optimization for Pruning

- Two ways to satisfy \( P.s_i \Rightarrow Q.t_j \)
  - \( \neg P.s_i \) : so \( s_i \) provides support for all \( t_j \)
  - \( Q.t_j \) : so \( t_j \) is supported by all \( s_i \)

<table>
<thead>
<tr>
<th>( ns )</th>
<th>( s_1 )</th>
<th>( s_2 )</th>
<th>( s_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_1 )</td>
<td>( \neg P.s_1 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( T_2 )</td>
<td>( \checkmark )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( T_3 )</td>
<td>( \checkmark )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( T_4 )</td>
<td>( \checkmark )</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>( \neg ns )</th>
<th>( s_1 )</th>
<th>( s_2 )</th>
<th>( s_3 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_1 )</td>
<td>( P.s_1 \wedge \neg Q.t_1 )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( T_2 )</td>
<td>( \checkmark )</td>
<td></td>
<td></td>
</tr>
<tr>
<td>( T_3 )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( T_4 )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>( T_5 )</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Optimization for Pruning

- If \( P.s_i \Rightarrow Q.t_j \) is not satisfied, no need to check remaining next properties

<table>
<thead>
<tr>
<th>( P_1 ) next ( Q_1 )</th>
<th>( P_2 ) next ( Q_2 )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( T_1 )</td>
<td>( \neg P.s_1 \wedge Q.t_1 )</td>
</tr>
<tr>
<td>( T_2 )</td>
<td>( \checkmark )</td>
</tr>
<tr>
<td>( T_3 )</td>
<td></td>
</tr>
<tr>
<td>( T_4 )</td>
<td></td>
</tr>
<tr>
<td>( T_5 )</td>
<td></td>
</tr>
<tr>
<td>( T_6 )</td>
<td></td>
</tr>
</tbody>
</table>

\[ \neg (P_1.s_i \Rightarrow Q_1.t_j) \]

\[ \neg (P_2.s_i \Rightarrow Q_2.t_j) \]
Confidence Levels

- High confidence
  - Pruning, including all prior interactions, completed prior to reply
  - Potentially high overhead (delay) for client
- Low confidence
  - Cover story appended to queue
  - Reply forwarded to client prior to any pruning
  - Very little overhead (delay) for client
- Medium confidence
  - Bound on number of replies optimistically forwarded

Confidence Guarantees

- High confidence
  - All validated replies represent correct behavior (up to that point)
- Low confidence
  - All violations are reported eventually, provided the client continues to interact
- Medium confidence
  - All validated replies, save the last n, represent correct behavior to that point

Related Work

- Confidence in proprietary implementations
  - Proof-carrying code, acceptance checking
- Inferring remote state
  - Control theory, model checking
- Dynamic testing
  - Eiffel, iContract, AssertMate, Biscotti, cidl
- Component wrappers to separate checking
  - RESOLVE
- Security and trust

Conclusions

- Prototype for CORBA components
- CIDL for behavioral specification
- Component implementation language neutral
- See: http://www.cis.ohio-state.edu/~paolo
- Utility of CORBA interceptors
  - Automatic tracking incoming/outgoing messages
  - Unfortunately, ORB dependent (ORBocus)
- Ongoing evaluation
  - Banking, auction, telephone activation system
  - Web services (WSDL)
- Natural extensibility of notation

Acknowledgements

- Distributed Components research group at Ohio State
  - Prakash Krishnamurthy, Scott Pike, Nigamanth Sriram, Murat Demirbas, Karuna Annadajala, Bob Nolan, Charlie Giles
- Funding sources:
  - National Science Foundation (ITR)
  - Lucent Technologies
  - Ohio Board of Regents
Increasing Client-Side Confidence in Remote Component Implementations

Ramesh Jagannathan
Paul A.G. Sivilotti
Dept. of Computer & Info. Science
The Ohio State University

http://www.cis.ohio-state.edu/~paulo

Pruner Algorithm - Example

- \( \text{top} = k \) next \( \text{top} = k \lor \text{top} = k + 1 \)

current abstract state set

\[
\begin{array}{c}
\text{K} \\
6
\end{array}
\]

new state set

\[
\begin{array}{c}
4 \\
5 \\
6 \\
7
\end{array}
\]