Dependability and Accountability for Context-aware Middleware Systems

Andrew Rice
Computer Laboratory
University of Cambridge
Distributed Middleware

- Middleware for sentient computing are inherently distributed
  - Many connected devices with wide-ranging capabilities
  - Multiple sources of context
Complicated Middleware

- Middleware for sentient computing are inherently complicated
  - Computationally intensive tasks are performed on behalf of low-power devices
  - Shared processing steps are made available to many applications
Examples

- SPIRIT + Active Bat system

Distributed access for applications using CORBA

Multi-lateration on raw bat sensor readings to derive a location

Monitors predefined regions for containment events and provide call backs to applications
Examples

- Cantag (image processing framework)
  - Supports distributed image pipelines
    - increase throughput
    - off-load complicated stages
  - May share computation steps when tracking multiple tag types
Validation

- Many operations performed by the middleware are asymmetric
  - difficult to compute but cheap to check
- Multi-lateration, containment monitoring, shape-fitting are all examples of this
Concept

- Define a set of trusted components
- Decompose remainder of system into untrusted components
  - check the outputs against the input data
Positive Validation

- Given an item of data from the middleware we can check that there is supporting evidence.
- Example: Whenever the Bat system reports a sighting we check that it is correct.
Positive Validation

Application: Tell me acr31's position

\[
\begin{align*}
c_1 &: t, d_1 \\
c_2 &: t, d_2 \\
c_3 &: t, d_3 \\
c_4 &: t, d_4 \\
c_5 &: t, d_5 \\
c_6 &: t, d_6 \\
\end{align*}
\]

\[
\begin{align*}
s_1 &: t, 'acr31' \\
l &: t, x, y, z \\
\end{align*}
\]
Positive Validation Shortcoming

Application: Tell me when acr31 enters SN04

Prove: acr31 not in SN04 at time $t_1$

Prove: acr31 in SN04 at time $t_2$

Also must show that acr31 not sighted between $t_1$ and $t_2$
Negative Validation

• Need to demonstrate a lack of sightings
• This can be done by coupling a sequence number when the Bat is polled
  • (we have to trust the polling component)

Application: Tell me when acr31 enters SN04

Prove: acr31 not in SN04 for poll('acr31',n)

Prove: acr31 in SN04 for poll('acr31',n+1)
Negative Validation

- This approach doesn't work for unpolled systems
- Example: Cantag may sight many tags in a single frame (or none at all)
- Negative validation in this scenario requires checking discarded data
- Example: Shape fitting
  - Fitted shapes close to contours? - +ve validation
  - Contours for unfitted shapes should have been discarded? - -ve validation
Costs of Validation

- Implemented validation for Cantag
- Huge variation in the runtime cost for validation
  - this is due to large variation in the number of entities

<table>
<thead>
<tr>
<th>Process</th>
<th>No. for +ve</th>
<th>No. for -ve</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distortion Correction</td>
<td>![Graph]</td>
<td></td>
</tr>
<tr>
<td>Fit Shape</td>
<td>![Graph]</td>
<td>![Graph]</td>
</tr>
<tr>
<td>Transform</td>
<td>![Graph]</td>
<td></td>
</tr>
<tr>
<td>Decode</td>
<td>![Graph]</td>
<td>![Graph]</td>
</tr>
<tr>
<td>Frame</td>
<td>←</td>
<td>←</td>
</tr>
<tr>
<td>Frame</td>
<td>←</td>
<td>←</td>
</tr>
</tbody>
</table>
Costs of Validation

- Cost of checking each stage can be used to classify each function

Positive asymmetric functions are common.

The addition of heuristics in early stages can reduce the number of elements for checking in expensive stages.

Decode

Decode + Heuristic
Evaluation

- Validation is a mechanism for exposing the state of the system to allow applications to validate behaviour.
- Checking each stage requires an application-specific check – no sensor system abstraction.
- Every piece of data might not need checking.
  - Infrastructure required to support this.
- Bugs were caught in Cantag – it does work!
Future Work

- Longterm deployment of validation in a real sentient middleware (as opposed to small tests)
- General interfaces for applications to request validation data and acquire validation algorithms