Middleware Reliability Implementations and Connector Wrappers

Authors: Jesse Sowell and Kurt Stirewalt

Presented by: Jesse Sowell

Overview

- **Context:** Adding new features to distributed applications by extending middleware
  - Behavioural reflection [FaPe'98]
  - Selective behavioural reflection [Yan+'02]
  - *Feature-oriented* design [Ren+'98]
    - *Feature* a product characteristic used in distinguishing programs in a family [BaSR'03]
- **Problem:** Evaluating feature decompositions
- **Hypothesis:** *Connector wrappers* [SpGa'03] useful for evaluating feature decomposition

Feature Composition in Theseus

- **Theseus:** framework for asynchronous request—reply communication
- Framework component = composable feature
  - Theory: roles/collaborations [VaNo'96,BaOM'92]
  - E.g. transport, FIFO vs. prioritized scheduling
- **Extensions:**
  - Reliability, e.g., retry, bounded retry, failover
  - Implemented as *wrappers*

Connectors and Wrappers

- **Given:** In software architecture, middleware functionality modelled by *connectors*
  - Distributed systems implement components
  - Middleware connectors guide their interaction
  - Formalized in CSP [AlGa'97]
    - Configuration = composition of components & connector
- **Given:** *Connector wrappers* a principled basis for creating and understanding wrappers [SpGa'03]
- **Key Idea:** Use connector wrappers to evaluate design of Theseus extensions

Enabling Evaluation

- **Idea:** Establish correspondence between connector wrappers and features, e.g.,
  - parallel composition = collaboration synthesis
  - action = operation invocation
- **What we did:**
  - Formalized Theseus core as a connector
  - Checked that connector wrappers compose
  - Refactored Theseus' reliability extensions
    - Made them "traceable" to connector wrappers
  - Evaluated implementation against specification

Details of Our Evaluation

- **Given:** Connector specification, implementation, connector wrapper, feature implementation
- **Procedure:**
  1. Establish traceability relationship T
  2. Compose wrapper with core
  3. Simultaneously interpreting spec and code
     - Use T to check conformance
     - Designer decides sufficiency
     - Results:
       - Refactoring for traceability helped improve design of both the core and reliability features
       - Process suggested further feature composition

Conclusions

- **Benefits:** Evaluation has potential for:
  - Validating/improving implementation
  - Discovering more effective decompositions
- Currently this is done by hand
- **Future Work:**
  - Perform additional case studies
  - Techniques and tools to:
    - Automate evaluation
    - Generate configuration code from specifications
    - Analyze configurations