Rapid Application Development via Instrumented Connectors

David Wile
(for Robert Balzer)
USC
Information Sciences Institute
wile@isi.edu

Architecture Levels

- Reference Architecture
  - Logical flows of data and control
  - Conceptual diagram
- Application Architecture
  - Physical flows of data and control
  - Lost logical flows through compilation and design choices
  - "All we've got" in legacy systems
  - All we can count on, with COTS
Rapid Application Development

- COTS vendor types (Wasserman):
  - Small, startup:
    - good to work with
    - high risk (almost like designing both products)
  - Medium, hungry, with track record:
    - ideal in that you can influence API design
    - responsive, until they grow
  - Big, entrenched:
    - stuck with their API
    - insensitive
    - BUT, polished product is almost irresistible

Rapid Application Development

- Architecture issues:
  - Logical architecture must be imposed on physical architecture
  - Usually done by mapping the ideal API primitives onto the physical architecture
- Sometimes impossible:
  - Style mismatches (Ahmed’s work)
  - Information completely missing from the API, e.g. event notification, focus information
Instrumented Connectors

- Enhance existing applications:
  - monitor performance and assumptions
  - improve efficiency
  - provide security
  - enforce constraints

- Leverage COTS, *especially Big Company COTS*, to enhance application functionality
  - plan logical architecture around instrumented connectors: as an integration framework
  - interface adaptation and filtering
  - infrastructure "cocoon"

Software Architectures
Software Architectures

GeoServer

Doctrine Authoring

Testbed Controller

Network Sockets

Corba

RPC

Architecture Connectors

Conduit for all inter-module interactions

- Network Sockets
- Event Broadcast
- Corba
- RPC
**Instrumented Connectors**

Conduit for all inter-module interactions

- Network Sockets
- Event Broadcast
- Corba
- RPC

Inserted Mediators enable

- Instrumentation
- Interface adaptation
- Filtering
- Value Added Infrastructure

Uniform Mediator
Interface Spanning
Integration Frameworks
Augmented Infrastructure Through Indirect Invocation

Dynamic Link Library (DLL)
(UNIX and Windows)

Uniform mechanism for Intermodule Interactions
- OS Services
- Network Sockets
- CORBA
- ...

Module

Commercial Infrastructure
Value-Added Infrastructure

Indirect Invocation Mediators

Directs

OS Services
Network Sockets
CORBA
**Dynamic Link Library (DLL)**

(UNIX and Windows)

Uniform mechanism for Intermodule Interactions
- OS Services
- Network Sockets
- CORBA
- ...

- Mediator added between Module & DLL component
- Mediator maintains DLL component API

---

**Instrumented Connectors Status**

Conduit for all inter-module interactions

- Network Sockets
- Event Broadcast
- CORBA
- RPC

Inserted Mediators enable
- Instrumentation
- Interface adaptation
- Filtering
- Value Added Infrastructure
Instrumented Connector Demos

- Network Sockets (UNIX)
  - Architecture Animator
  - Architecture Driver
- Graphic User Interface (Win95/NT)
  - Compose Eudora Replies in EMACS
- Extend PowerPoint into Architecture Editor (Win95/NT)
- OS Services (Win95/NT)
  - Encryption Archive
  - Virtual File System
- Infrastructure (Win95/NT)
  - Interactive Probes
  - Architecture Instrumentation Controller
  - Architecture Animation
  - Validating Architecture Constraints
Dynamic Link Library (DLL) (UNIX and Windows)

Uniform mechanism for Intermodule Interactions
- OS Services
- Network Sockets
- CORBA
- ...

- Mediator added between Module & DLL component
- Mediator maintains DLL component API

Instrumented Connectors Status
Conduit for all inter-module interactions

- Network Sockets
- Event Broadcast
- Corba
- RPC
- Graphic User Interface
- File System
- OS Services

Inserted Mediators enable
- Instrumentation
- Interface adaptation
- Filtering
- Value Added Infrastructure
<table>
<thead>
<tr>
<th>Instrumented Connector Demos</th>
</tr>
</thead>
<tbody>
<tr>
<td>- Network Sockets (UNIX)</td>
</tr>
<tr>
<td>- Architecture Animator</td>
</tr>
<tr>
<td>- Architecture Driver</td>
</tr>
<tr>
<td>- Graphic User Interface</td>
</tr>
<tr>
<td>- (Win95/NT)</td>
</tr>
<tr>
<td>- Compose Eudora Replies in EMACS</td>
</tr>
<tr>
<td>- OS Services (Win95/NT)</td>
</tr>
<tr>
<td>- Encryption Archive</td>
</tr>
<tr>
<td>- Virtual File System</td>
</tr>
<tr>
<td>- Infrastructure (Win95/NT)</td>
</tr>
<tr>
<td>- Interactive Probes</td>
</tr>
<tr>
<td>- Architecture Instrumentation Controller</td>
</tr>
<tr>
<td>- Architecture Animation</td>
</tr>
<tr>
<td>- Validating Architecture Constraints</td>
</tr>
</tbody>
</table>
Effort to Build

- Elapsed Time 2.5 Weeks

Effort to Build

- Elapsed Time 2.5 Weeks
- Development Time
  - PowerPoint Driver 7 Days
  - Instrumented Connectors 1 Day
  - ACME Analyzer 5 Days
  Total 13 Days
# Effort to Build

- **Elapsed Time**: 2.5 Weeks
- **Development Time**
  - PowerPoint Driver: 7 Days
  - Instrumented Connectors: 1 Day
  - ACME Analyzer: 5 Days
  - Total: 13 Days
- **Code Size**
  - PowerPoint Driver: 15 Pages
  - Instrumented Connectors: 2 Pages
  - ACME Analyzer: 10 Pages
  - Total: 27 Pages

---

# ACME Editor

[Diagram of ACME Editor]
ACME Editor

Lessons Learned

- PC COTS Functionally Rich
- PC COTS Highly Adaptive
- Inst. Connectors Crucial (Visibility)
- No Architecture Mismatch
Refining an Architecture

- New diagram contains (empty) subarchitecture for selected component
- Subarchitecture wired to outer architecture through refined component
- Shadow on refined component indicates it has a subarchitecture

Abstracting an Architecture

- Several components abstracted into a single one
- New diagram contains selected components as a subarchitecture
- Subarchitecture wired to outer architecture through abstract component
• Several components abstracted into a single one
• New diagram contains selected components as a subarchitecture
• Subarchitecture wired to outer architecture through abstract component
Encryption Archive

- Password Protected Access
  - Existence projected into target directory
  - Decrypted upon access
- Hidden Persistent Storage
  - Encrypted file contents
  - Encrypted file name
Virtual File System

- Defines file system subset available to Program
- Features:
  - Hidden Directories/Files
    - Existence Undetectable
  - Virtual Directories/Files
    - Projected from elsewhere
  - Access Restrictions
    - By Directory, File, Pattern
Interactive Probes

- Interactive Probe Selection
  - Process to Instrument
  - Interfaces to Inspect

- Interactive Probe Events
  - Displays interface arguments
  - Return Value (if applicable)
  - User may
    - Authorize/Prohibit request
    - Change Arguments
    - Change Return Value
    - Stop further Probes
    - Reselect Probes
C++ Metrics

- Capture Compilation Activity
  - Compilation Errors
  - Compilation Warnings

Architecture
Instrumentation Controller

PowerPoint GUI:

- Instrumentation Specification
  - Instrumentation Shape attached to connector
  - Form specifies Instrumentation
- Instrumentation Installation
  - Selected from mouse menu
  - Installation affects running processes
  - Color indicates installation state
Architecture Animation

PowerPoint GUI:

- **Interaction Animation**
  - Tokens move between components
  - Tokens carry communication

- **Interaction Sources**
  - Live
    - Instrumented Connectors
    - Interaction Aggregators
  - Simulated
    - Architecture Simulator
    - Component Models
Validating Architecture Constraints

- Constraints expressed as compound events
- Event Monitor
  - Monitors primitive events
  - Detects specified compound events
  - Announces their occurrence
- PowerPoint Animator
  - Highlights participating events
Mediator Technology

- Can install mediator on any call from a process to a shared library (DLL in NT)
  - Includes all operating system calls
  - As well as third party APIs
  - GUI interception is more ad hoc, based on Microsoft’s advertised API
- Mediator is a function with exactly the same interface as what it mediates
  - Arguments and types
  - Return values
  - Exceptions handled and thrown (problematic)

Wrapper Technology

- Each wrapper is a collection of mediators
  - Most applications require that several mediators be installed at once
  - Constitutes the “cocoon”
  - Running binary code wrapped -- no sources
- Wrapper manager permits installing and uninstalling
  - Multiple wrappers can have functions in common -- "onion skin" metaphor
  - Installed on a per-process basis
  - Each wrapper has a per-thread state variable
Wrapper Issues

- Propagation issues
  - Propagate installed wrappers to spawned process?
  - Mediator functions usually call original function: should it get wrapped version from other wrapper layer?
- Major problems with current scheme
  - Difficult to find out what APIs need to be mediated
  - Writing mediation code and packaging into a DLL is rather tedious and should not need to be learned to use mediators and wrappers
  - Packaging up the mediators into a wrapper is awkward

Future Technical Support

- Spy program for watching API calls
- Design of a high level language to specify:
  - Which functions to mediate in a given wrapper
  - What their definitions are
- Integrated mechanism allowing the user to point to a set of APIs determined using the spy program and then providing the bodies
  - Appropriate interface code will be put around the definitions
  - Code to do the installation will be generated
Future Work

- Generating Components
  - Architecturally Compliant Components

![Diagram showing the process of generating components with component specification and architecture specification leading to a component.]

Future Work

- Generating Components
  - Architecturally Compliant Components

![Diagram showing the process of generating components with component specification, architecture specification, generator, and architecturally adapted component.]

Future Work

- Generating Components
  - Architecturally Compliant Components
  - From specification of instrumented connectors

- Integrating Software
  - Program Interaction Managers

Program Interaction Principles

- All interactions can be detected
- Detected interactions can be mediated
  - Monitored/Logged
  - Allowed/Rejected
  - Coordinated/Tracked
  - Transformed
- Mediation can be performed by separate Interaction Managers
- Interaction Managers can be managed
Security/Safety Manager

- Manages behavior of an application/subsystem
  - Restricts application/subsystem behavior (via rules)
  - Restrictions can be safely nested in multiple layers
  - => Security/Safety Manager can be safely user extended

Authorization Manager

- Manages behavior of environment on subsystem
  - Restricts external interactions (via rules)
  - Restrictions can be safely nested in multiple layers
  - => Authorization Manager can be safely user extended
Coordination Manager

- Manages coordination between applications
  - Remote mediator detects coordinating event
  - Remote mediator initiates coordinating interactions

Integration Manager

- Manages integration of applications
  - Remote mediators detect coordinating events
  - Remote mediators initiate coordinating interactions
  - Integrator translates into receiving context
Quality of Service Manager

- Manages use of external resource(s)
  - Detects quality of service available
  - Initiates adapting interactions in Client
  - Negotiates QoS with Resource (server)

Future Work

- Generating Components
  - Architecturally Compliant Components

- Integrating Software
  - Program Interaction Managers
  - Coordinated Distributed Instrumentation
  - Secure Instrumentation

- Securing Software
Safety Manager

- Automatically Safeguard Persistent Resources
  - Backup Files and Registry settings before Modify
  - UNDO restores saved resources
  - For Buggy, Suspect, & Malicious Programs

NT Security Manager

- Manages behavior of an application/subsystem
  - Restricts application/subsystem behavior (via rules)
  - Restrictions can be safely nested in multiple layers
  - Security Manager can be safely user extended
Future Work

- Generating Components
  - Architecturally Compliant Components
- Integrating Software
  - Program Interaction Managers
  - Coordinated Distributed Instrumentation
  - Secure Instrumentation
- Securing Software
  - Safe Execution Environments
    - Copy Before Modify
  - NT Security Monitor