Embracing Requirements Variety for e-Governments based on Multiple Product-Lines Frameworks

Mikio Aoyama  
Kenichiro Watanabe  
Yu Nishio  
Yasuyuki Moriwaki  
Nanzan University  
Fujitsu Communication Systems Limited  
Japan  
Japan  
mikio.aoyama@nifty.com  

All Rights Reserved, Copyright Mikio Aoyama, 2003
Scenario

- Problem Domain
- Approach: Reuse of System
- Framework for Multiple Product Lines
- Scenario-Based Framework Composition and Collaboration Patterns
- Experience and Evaluation
- Future Work
- Conclusions
**Problem Domain: e-Government**

- **e-Government Systems:** Ministry of Land, Infrastructure and Transport Government of Japan (MLIT)

- **Domain:**
  - (Total) Water Management
  - Dam, River Gate, Water Supply, etc.
  - Road Management
  - Environment Management

- **Government Hierarchy**
  - Central (MLIT) System:
    - Huge Nationwide Network System
  - Local System: Prefecture, City, etc.
Problem Domain: An Example:
A Part of River Information Mgmt System

Rainfall Telemetry Station
Water Level Telemetry Station at River
Water Level Telemetry Station at Dam

Central Monitor System
Application Server
PDP/Controller
Monitor Client (Browser)
Web Server
Router
Flood Monitor

Control Station
Problem Domain: Nature

- Large Number of Systems in a Short Time
  - 20+ Systems: 80% Delivered at 4Q
  - Very Short Time to Delivery (3 Months)
    - By Small Team (Can’t Retain Large Team)
- Variety over Similar Product Functionality
  - Multiple Product Lines
  - Levels of Government Hierarchy
  - System Scale & Complexity (Not Functionality)
    - From 4~5 Clients to 100+ Clients per System
    - Variety of Software Requirements
- Assuring High Quality for Public Infrastructure
  - Architecture for Simplex and Duplex
  - Nationwide Networking
Approach: Reuse of System via Multiple Product Lines Frameworks

- **Framework + Scenario-Based Composition**
- **Entirely New Framework Based on Domain/Product Line Analysis (one year)**
  - Government Standard and Domain Knowledge
  - Pluggable Framework Customizable with Scenario and Component
  - Framework Evolution from Single Domain to Multiple Domains
- **Scenario-Based Composition and Collaboration Pattern**
  - Event-Based Scenario Manager for Loosely Coupled Framework Composition
Frameworks for Multiple Product Lines
Platform Independent Hierarchical Architecture

For a Product
- River Information Mgmt.
- Water Supply Monitoring
- Dam Monitoring
- Environment Monitoring

For a Product-Line
- Waters
- Lands

For All Product-Lines
- Base Framework
- Platform (for Linux/UNIX and Windows NT/2000)

A Product-Line Framework for Water Information Mgmt.
- River Information Mgmt.
- Waters
- Base Framework
- Platform

Products

A Product-Line Framework for Dam Monitoring
- Dam Monitoring
- Waters
- Base Framework
- Platform

Products
Frameworks for Multiple Product Lines
Framework Selection and Customization

From Framework Repository to Product Framework
Promoting Reuse of Standard Composition Set of Frameworks to Narrow Down the Variety of Products
## Frameworks for Multiple Product Lines
### Framework Selection and Customization

Mapping Framework to Specific Platform via MVC

<table>
<thead>
<tr>
<th>Layer</th>
<th>View</th>
<th>Controller</th>
<th>Model</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Product Line</strong></td>
<td>River Info Mgmt</td>
<td>- River Water Calc</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Collection</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>- Fault Info Mgmt</td>
<td></td>
</tr>
<tr>
<td><strong>Common Business</strong></td>
<td>Waters</td>
<td></td>
<td>Temporal DBMS</td>
</tr>
<tr>
<td><strong>Base Frwk</strong></td>
<td>- HMI</td>
<td>- Scenario Mgmt</td>
<td>DBMS</td>
</tr>
<tr>
<td></td>
<td>- Reporting</td>
<td>- Event Mgmt</td>
<td></td>
</tr>
<tr>
<td></td>
<td>- Web Presentation</td>
<td>- Time Mgmt</td>
<td></td>
</tr>
<tr>
<td><strong>Platform</strong></td>
<td></td>
<td>RAS (Duplex)</td>
<td></td>
</tr>
</tbody>
</table>

All Rights Reserved, Copyright Mikio Aoyama, 2003
Scenario-Based Component Composition

Problems

Direct Collaboration between Components

Add System Specific Component

- Change of Collaborating Components
- Assuring Consistency
- Increase Complexity of Components Collaboration

Need Loosely Coupled Composition

Change of Interface

Consistency of Data

Change of Data

Platform

RiverCalc
EvMgmt

Collection

Speci-

fic

All Rights Reserved, Copyright Mikio Aoyama, 2003
Scenario-Based Component Composition Approach: Scenario Manager

**Scenario**
- A Sequence of Standard Commands
  - \{Scenario, Component\}
- Derived from Operational Requirements

**Scenario Management**
- Consistent Control of Collaboration
- Interface Marshalling
- Consistency Control of Processing & Data

Event-Driven Scenario Manager
- Absorb Change by Indirect Collaboration
- Use Scenario-Based Collaboration Patterns
Scenario-Based Component Composition

Scenario Manager: Collaboration Patterns

Problem: Variety of Invocation via Scenario Manager

Solution: Scenario-Based Collaboration Patterns

Reuse of Existing Scenarios

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Component in Frwk</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Flow</td>
<td>Calc, Collection, Schedule, Update</td>
</tr>
<tr>
<td>Call Return</td>
<td>Reporting, etc</td>
</tr>
<tr>
<td>Event</td>
<td>UI Collaboration, Data Distribution</td>
</tr>
<tr>
<td>Shared Data</td>
<td>Calc, Collection, Retrieve</td>
</tr>
<tr>
<td>Layered</td>
<td>Peripheral Control</td>
</tr>
</tbody>
</table>

Creation of New Scenarios Based on Scenario in the Frwk
Scenario-Based Component Composition
Scenario Manager: Collaboration Patterns

Examples of Collaboration Patterns

(1) Data Flow Pattern
1-a) Data Flow with Data Passing

Scenario Manager

A ----> B ----> C

1-b) Data Flow with Data Reference

Scenario Manager

A ----> B ----> C

DB

Shared Data

Legend:

: Event/Message Flow

: Data Access

(2) Call-Return Pattern
2-a) Call-Return Invocation of Component

Scenario Manager

A ----> B ----> C

2-b) Call-Return Invocation of Scenario

Scenario Manager

Scenario Invocation

Scenario Termination

A ----> B ----> N

Scenarios

Legend:

: Event/Message Flow

: Data Access
Scenario-Based Component Composition Framework-Based Development Process

Framework + Scenario-Based Composition

Building a Base Framework Based on Standard PL Architecture

Selecting Reusable Scenario

Reuse/Add Scenario Reuse/Add Customer-Specific Component

Standard Frameworks across Products Lines

Framework Selection

Product-Line Framework

Scenario Selection

Product Framework for Specific Customer

Customize Scenario and Component

System for the Customer
Experience and Evaluation
Product-Line Systems Delivered

Accumulated Number of Delivered Products

Evolution of Product Lines

- Dam PL
- Water PL
- Road PL
- Telemetry PL
- River PL

Number of Delivered Systems

Year


100 90 80 70 60 50 40 30 20 10 0
Experience and Evaluation
Component Reuse

Examples in 3 River Information Mgmt Systems

- 60~90% Reuse of Product-Line Frameworks
- Larger-Scale Systems Demand More Customizations

Reason: Increase the Variety of Input/Output

<table>
<thead>
<tr>
<th>System</th>
<th>Scale</th>
<th>Additional Component</th>
<th>Standard Framework Reused</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Large</td>
<td>40%</td>
<td>60%</td>
</tr>
<tr>
<td>B</td>
<td>System</td>
<td>20%</td>
<td>80%</td>
</tr>
<tr>
<td>C</td>
<td>Small</td>
<td>0%</td>
<td>100%</td>
</tr>
</tbody>
</table>
Experience and Evaluation
Reuse of Scenario Collaboration Patterns

Reuse of Scenario Collaboration Patterns at 3 River Information Mgmt System

Similar Reuse Ratios across Different Scale Systems over Product Line implies Reuse of Common Architecture Set

- System A
- System B
- System C

Large
Small

Bar chart showing data flow, call-return, interpreter, event-driven, shared data, and layer for systems A, B, and C.
Future Work

- **Reengineering Platform**
  - XML Web Service

- **Application to Other Domains**
  - Evaluation of Experience
  - Materialize Lessons Learned

- **Adoption to New Initiative: e-Japan II**
  - New Initiatives for Next 3 Years
Conclusions

Variety and Extremely Short Delivery

Changed the Style of Development:
- From Development to System Reuse
- Mechanism for System Reuse
  - Multiple Product-Line Development
  - Scenario-Based Composition
- Orchestrating the Best Practice
  - Serious Domain Analysis and Framework Architecture Design
  - Plain Vanilla Approach (No Surprise) Practiced by Average (But Serious) Engineers with Serious Architects