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

Mikio Aoyama

Nanzan University
Japan
mikio.aoyama@nifty.com

Kenichiro Watanabe
Yu Nishio
Yasuyuki Moriwaki

Fujitsu Communication Systems Limited
Japan

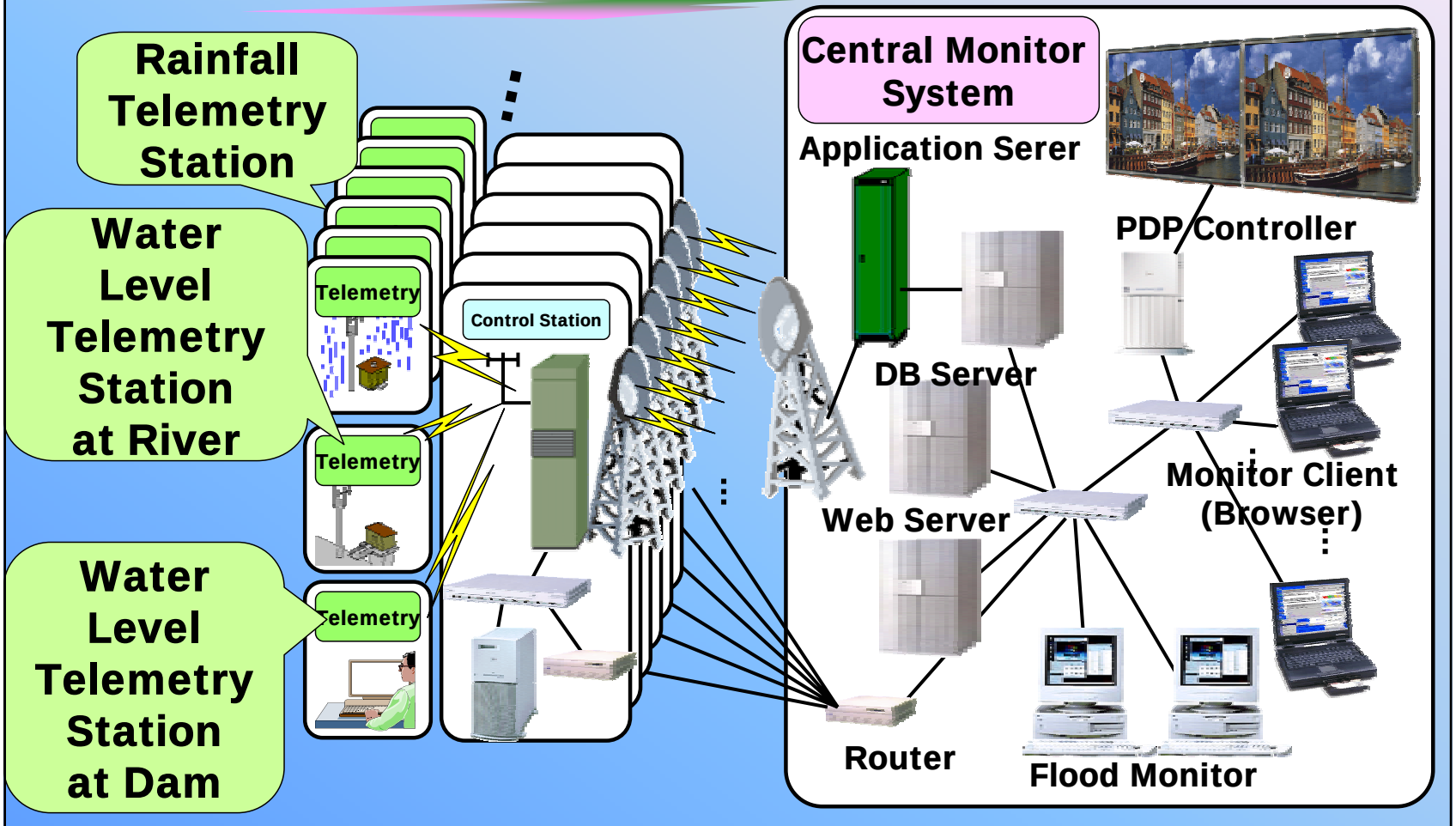
Scenario

- 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



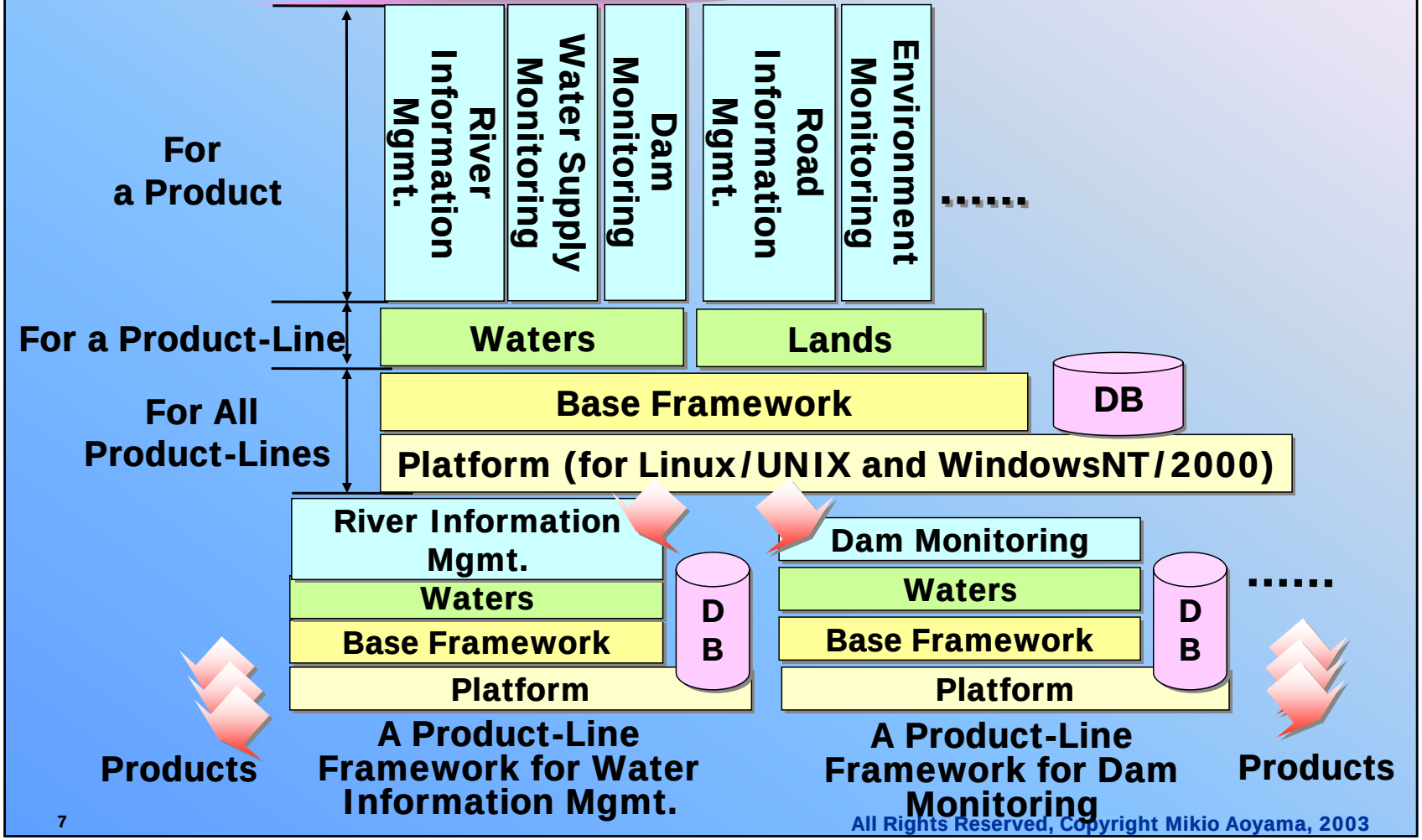
Problem Domain: Nature

- **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

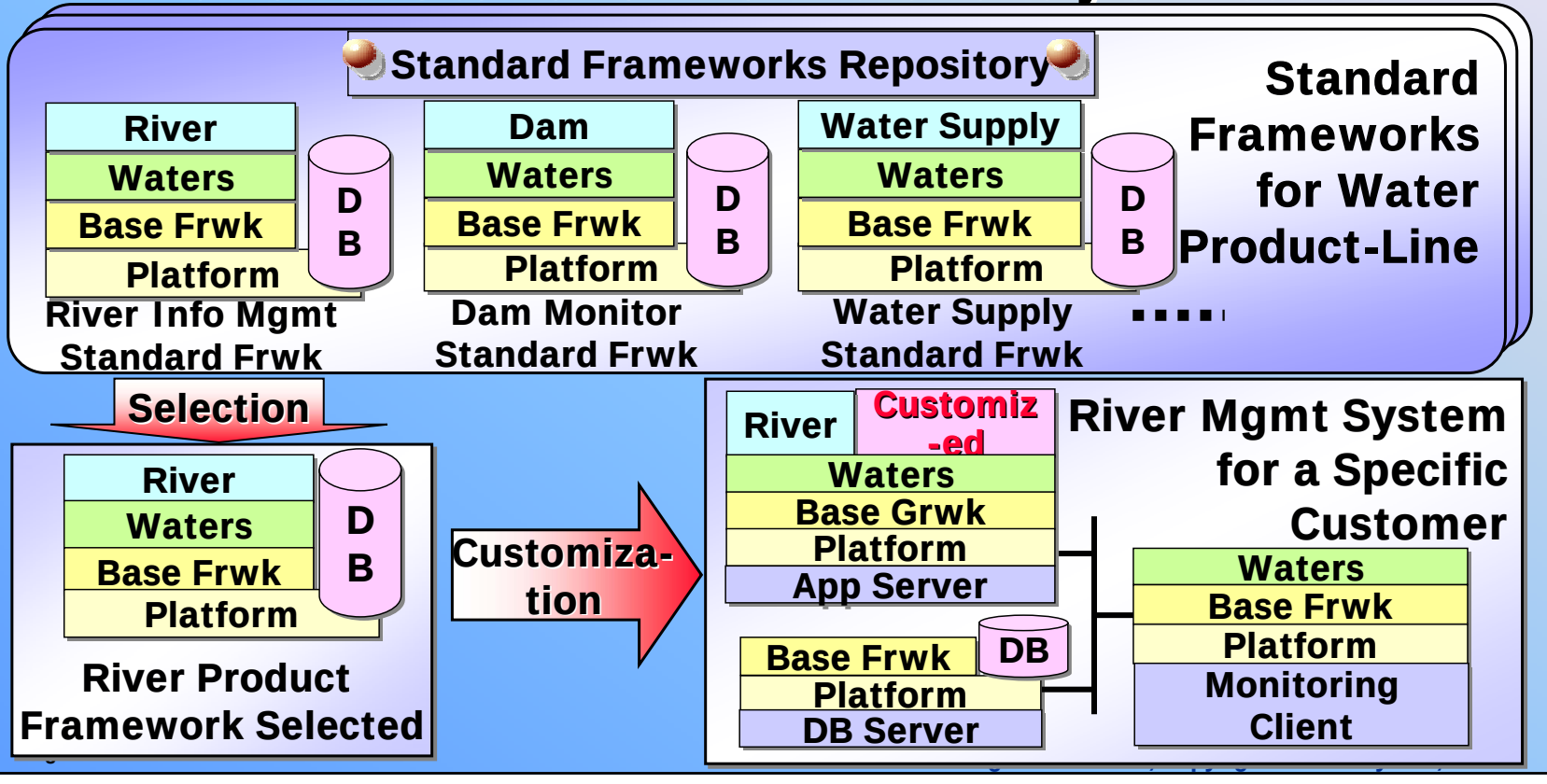
Frameworks for Multiple Product Lines
Platform Independent Hierarchical Architecture



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

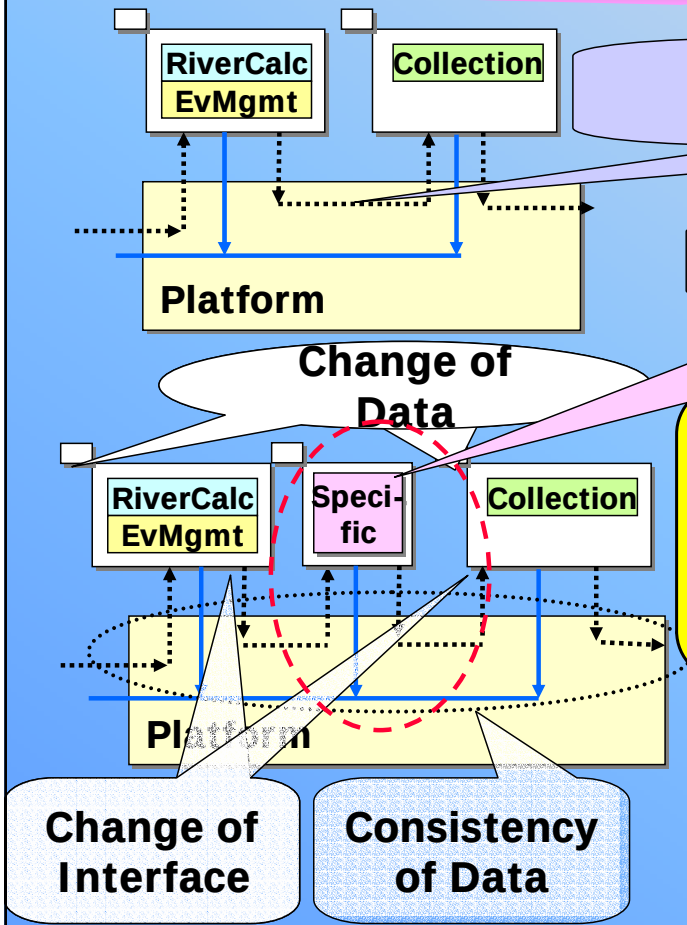


Frameworks for Multiple Product Lines Framework Selection and Customization

- Mapping Framework to Specific Platform via MVC

Layer		View	Controller	Model
Product Line	River Info Mgmt		- River Water Calc - Collection - Fault Info Mgmt	
Common Business	Waters			Temporal DBMS
Base Frwk		- HMI - Reporting - Web Presentation	- Scenario Mgmt - Event Mgmt - Time Mgmt	DBMS
Platform			RAS (Duplex)	

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

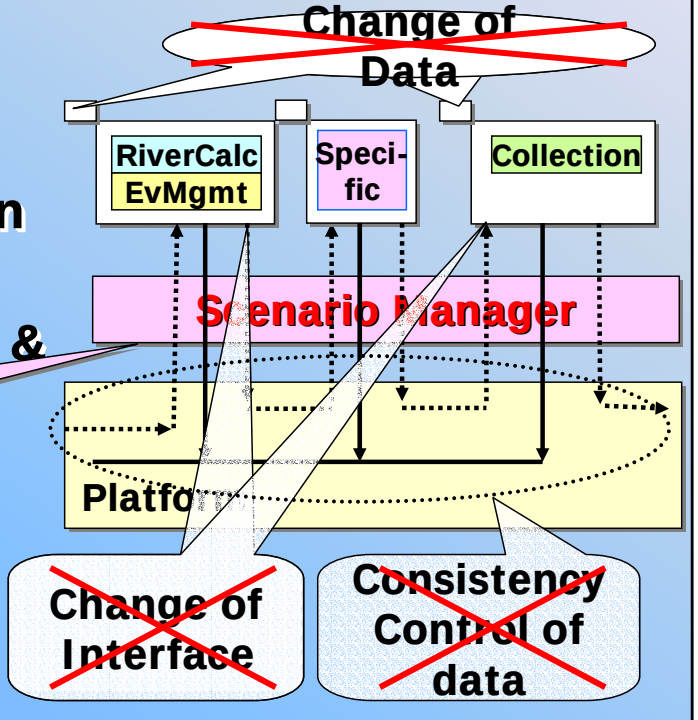
Scenario-Based Component Composition Approach: Scenario Manager

- **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

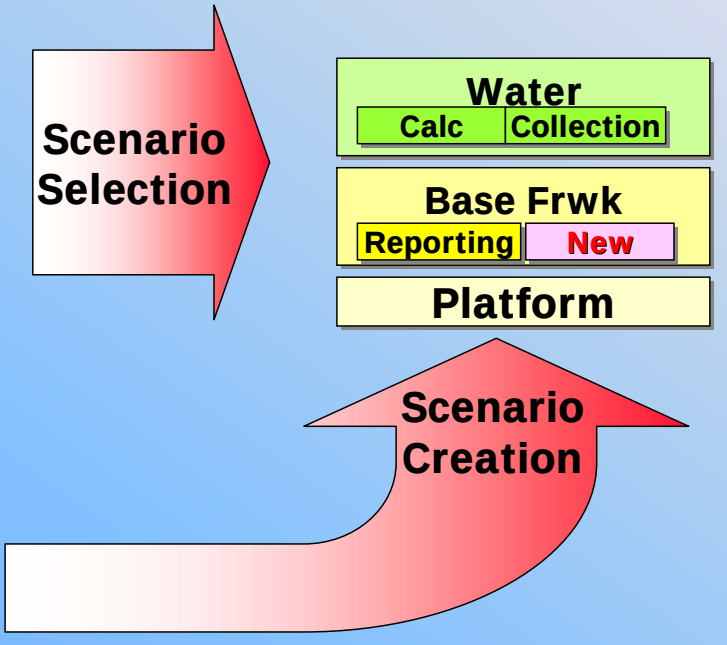
Use Scenario-Based Collaboration Patterns



Scenario-Based Component Composition
Scenario Manager: Collaboration Patterns

- Problem: Variety of Invocation via Scenario Manager
- Solution: Scenario-Based Collaboration Patterns
  - Reuse of Existing Scenarios

Pattern	Component in Frwk
Data Flow	Calc, Collection, Schedule, Update
Call Return	Reporting, etc
Event	UI Collaboration, Data Distribution
Shared Data	Calc, Collection, Retrieve
Layered	Peripheral Control



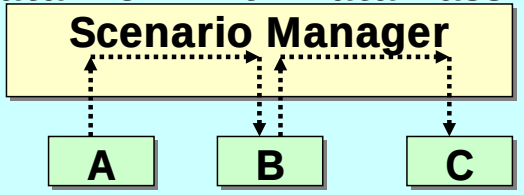
  - Creation of New Scenarios Based on Scenario in the Frwk

Scenario-Based Component Composition
Scenario Manager: Collaboration Patterns

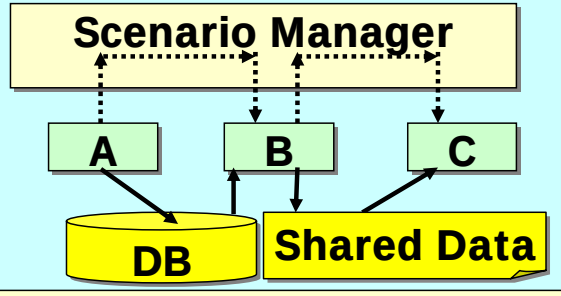
☞Examples of Collaboration Patterns

(1)Data Flow Pattern

1-a) Data Flow with Data Passing

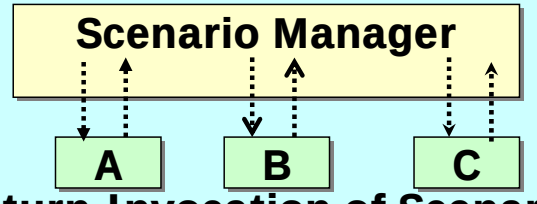


1-b) Data Flow with Data Reference

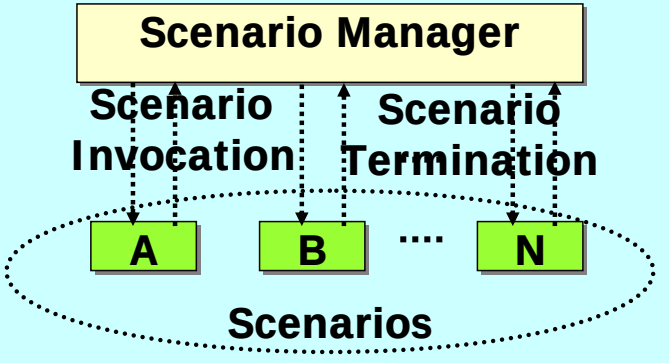


(2) Call-Return Pattern

2-a) Call-Return Invocation of Component

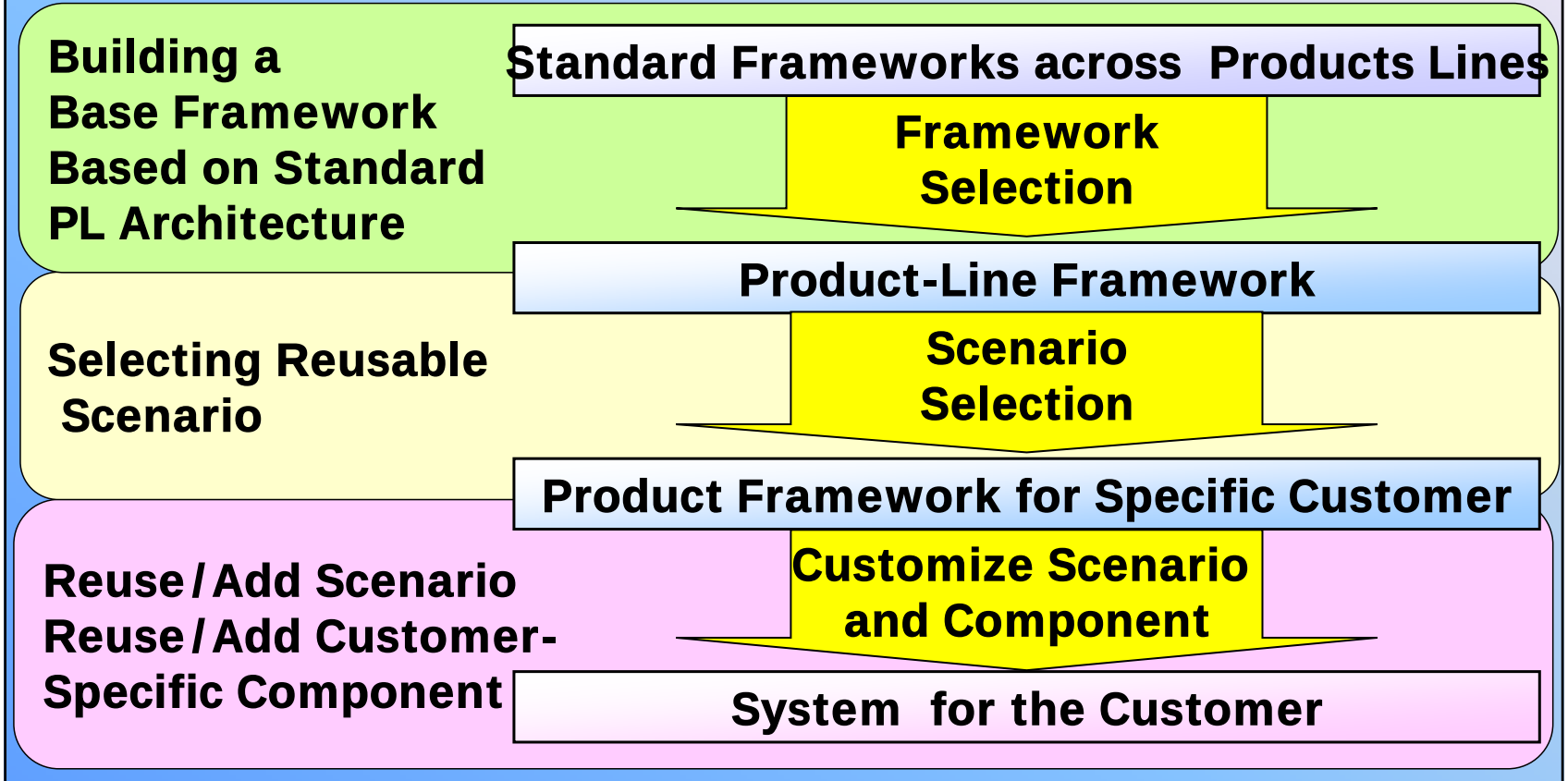


2-b) Call-Return Invocation of Scenario



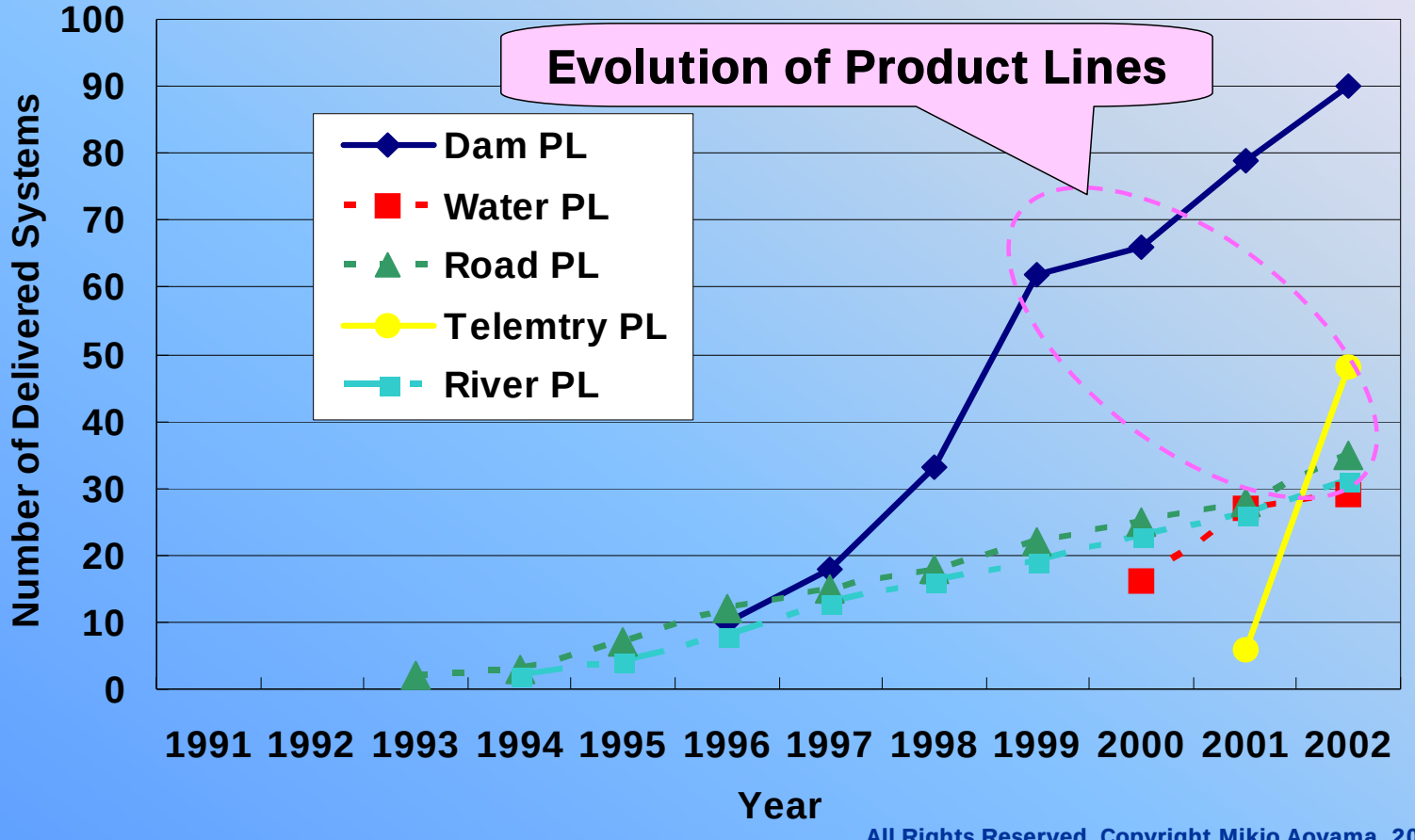
Scenario-Based Component Composition Framework-Based Development Process

☞Framework + Scenario-Based Composition



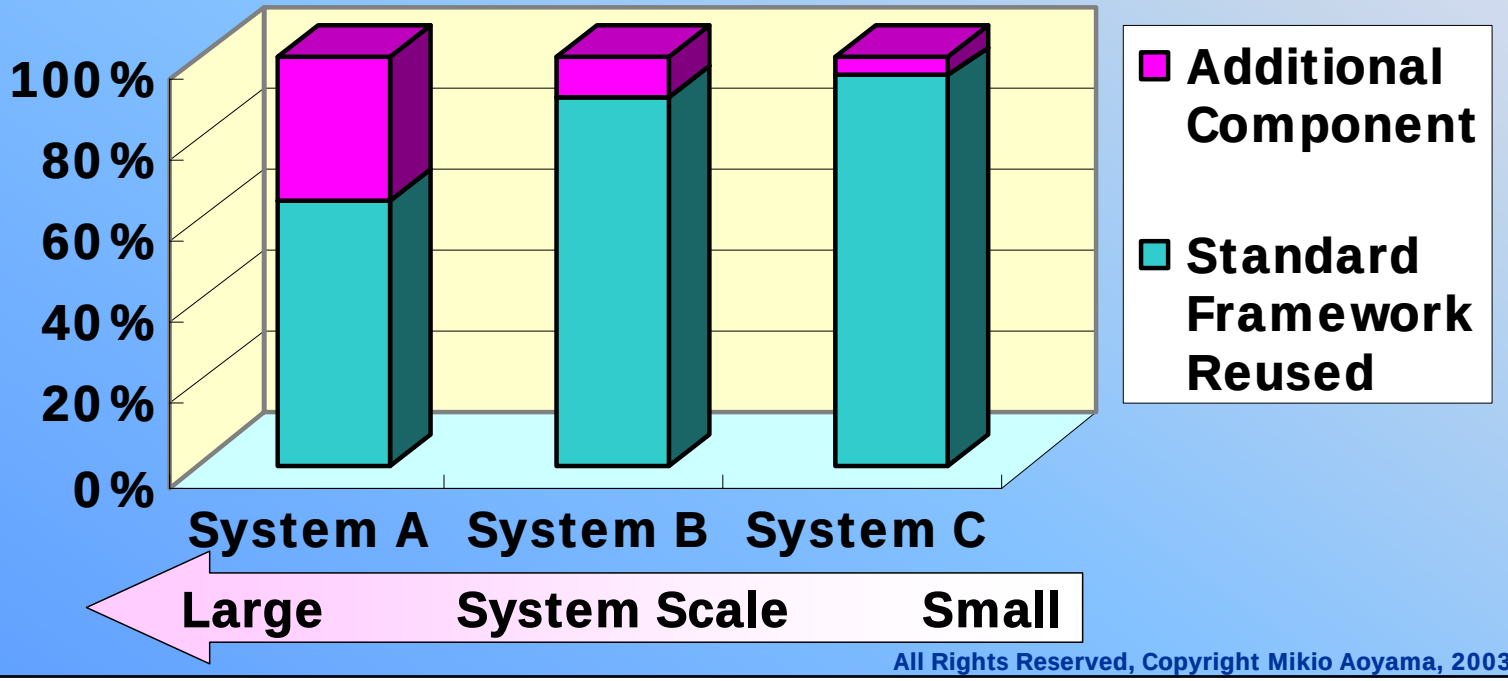
Experience and Evaluation
Product-Line Systems Delivered

☞Accumulated Number of Delivered Products



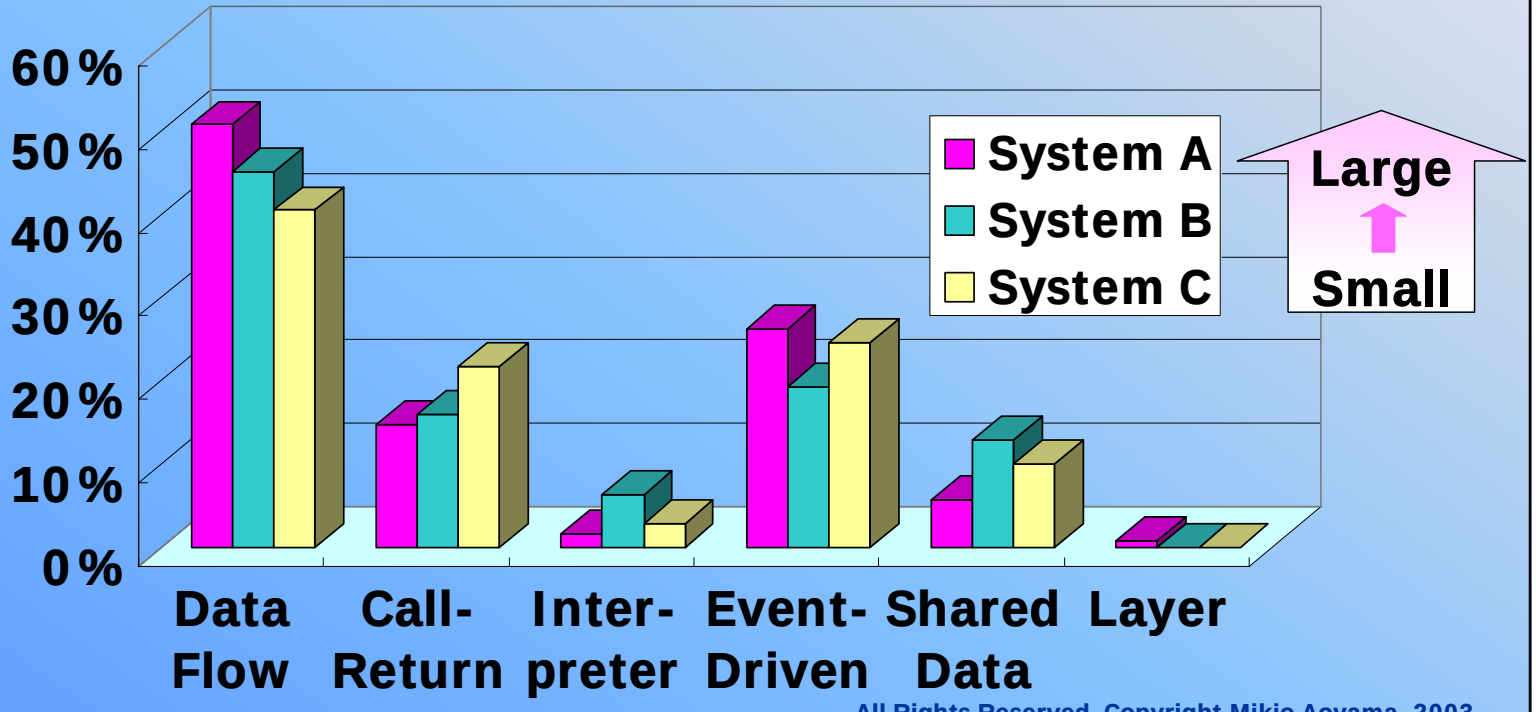
Experience and Evaluation
Component Reuse

- **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



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**



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
- Orchestration of Best Practice
  - Serious Domain Analysis and Framework Architecture Design
  - Plain Vanilla Approach (No Surprise) Practiced by Average (But Serious) Engineers with Serious Architects