Qualitative ROI for MDA Projects

Ken Sayers - Chubb and Son, Inc.

OMG UML Workshop

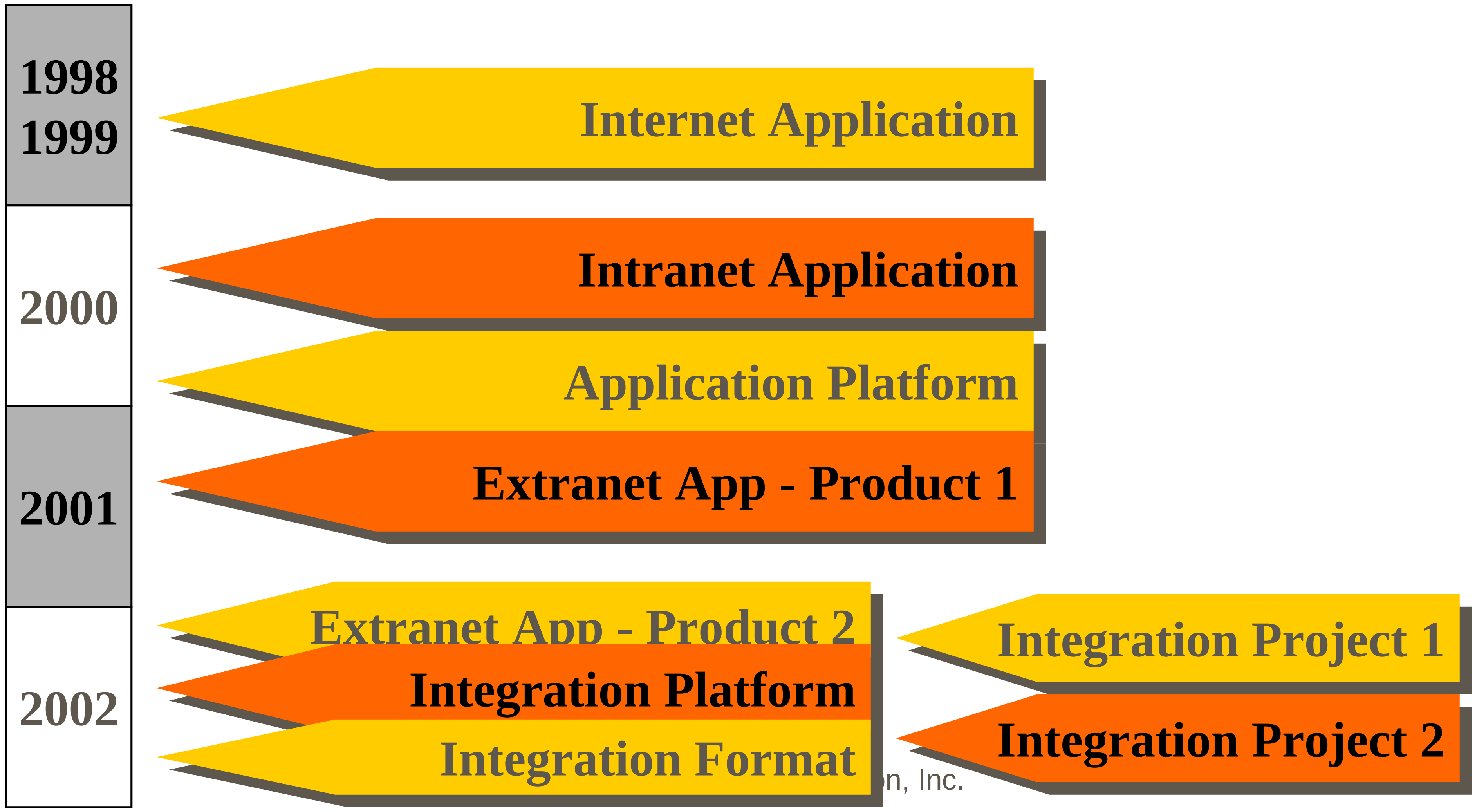
San Francisco, CA

October 21-24, 2002

Introduction

- We started doing MDA before the term MDA was coined
- We wanted to generate code to improve productivity of developers, quality of code and consistency of architecture
- We have anecdotal evidence of good return on our investment

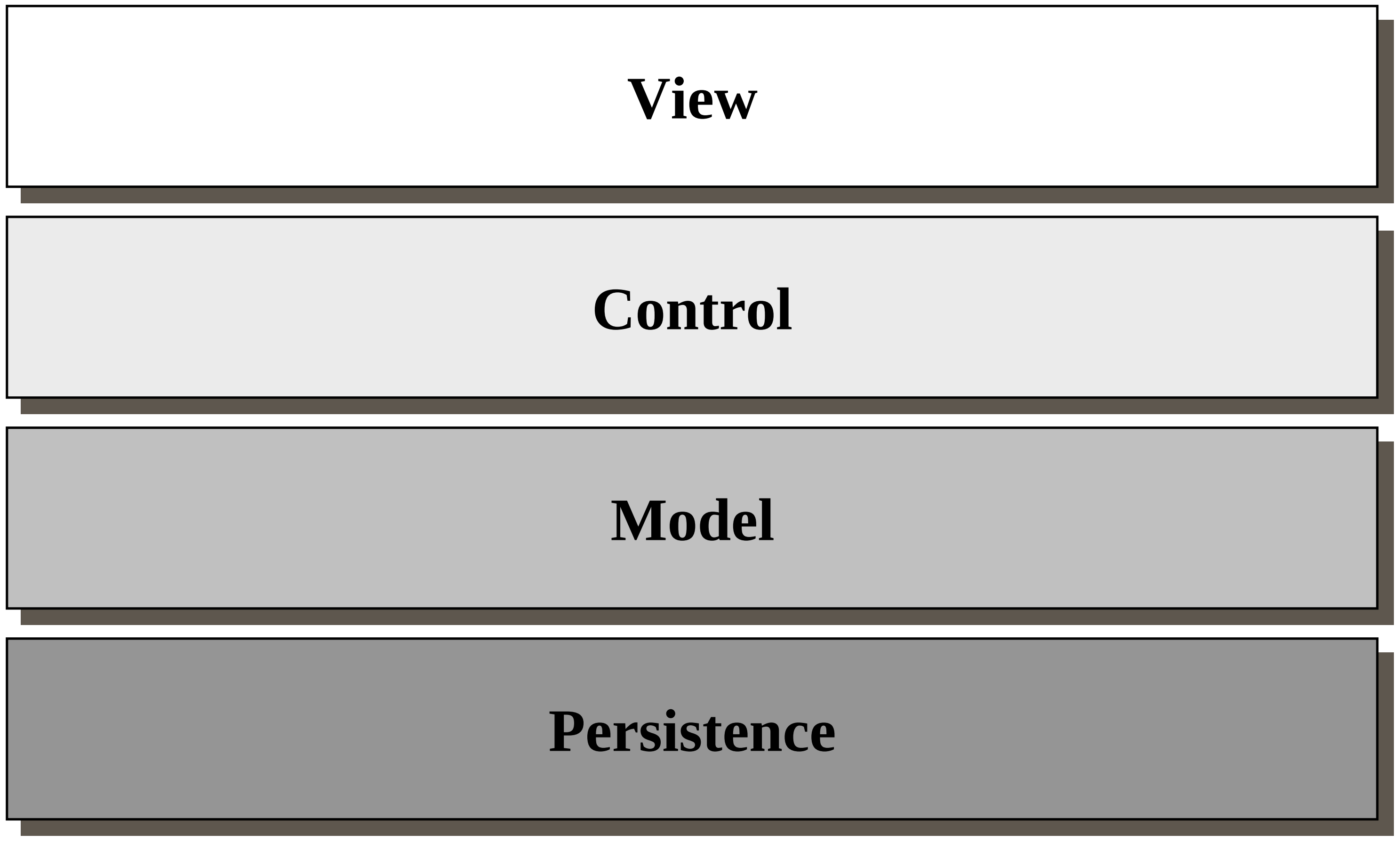
Chubb MDA Experience



Smalltalk Applications

- Large and Small
- Learned Object Technology
- Learned Patterns
- Evolved Architecture
  - Model - Domain
  - View - User Interface
  - Controller - Process Flow
  - Persistence - Save and Load from Store

Architecture



Internet Application

- 1998-2000
- Internet Travel Insurance Application
- Sell directly to consumer
- Team of One Architect, 2-4 Developers
- Homegrown Code Generation
- Architecture Leveraged from Smalltalk

Internet Application

- UML Models
  - Domain - Class Diagrams
  - Process Model - State Machines

Internet Application

- Code Generation
  - Homegrown Tool
    - Vendor Scripting Language
    - Visual Basic
    - Preserve Custom Code on Regeneration
  - Process Controllers - From State Machines
  - Domain Model - From Class Diagrams

Internet Application

- Hand Coded
  - Persistence - Vendor Helper Classes
  - User Interface - Java Server Pages
  - Specific Business Rules
    - Field Validation
    - Cross Object Validation
    - Process Exceptions

Internet Application

- Results
  - Generated OUR Architecture
  - Generation can be changed quickly
  - Consistent Architecture throughout system
  - Productive Developer (only one using generation. The MDA Architect.)
  - High Quality
  - Knowledge of the tool left project with Architect

Intranet Application

- 1999 to present
- Intranet System
- Team of Local and Distributed Architects with 4-5 Developers
- Modeled/Generated Domain and Process - Same As for Internet Application

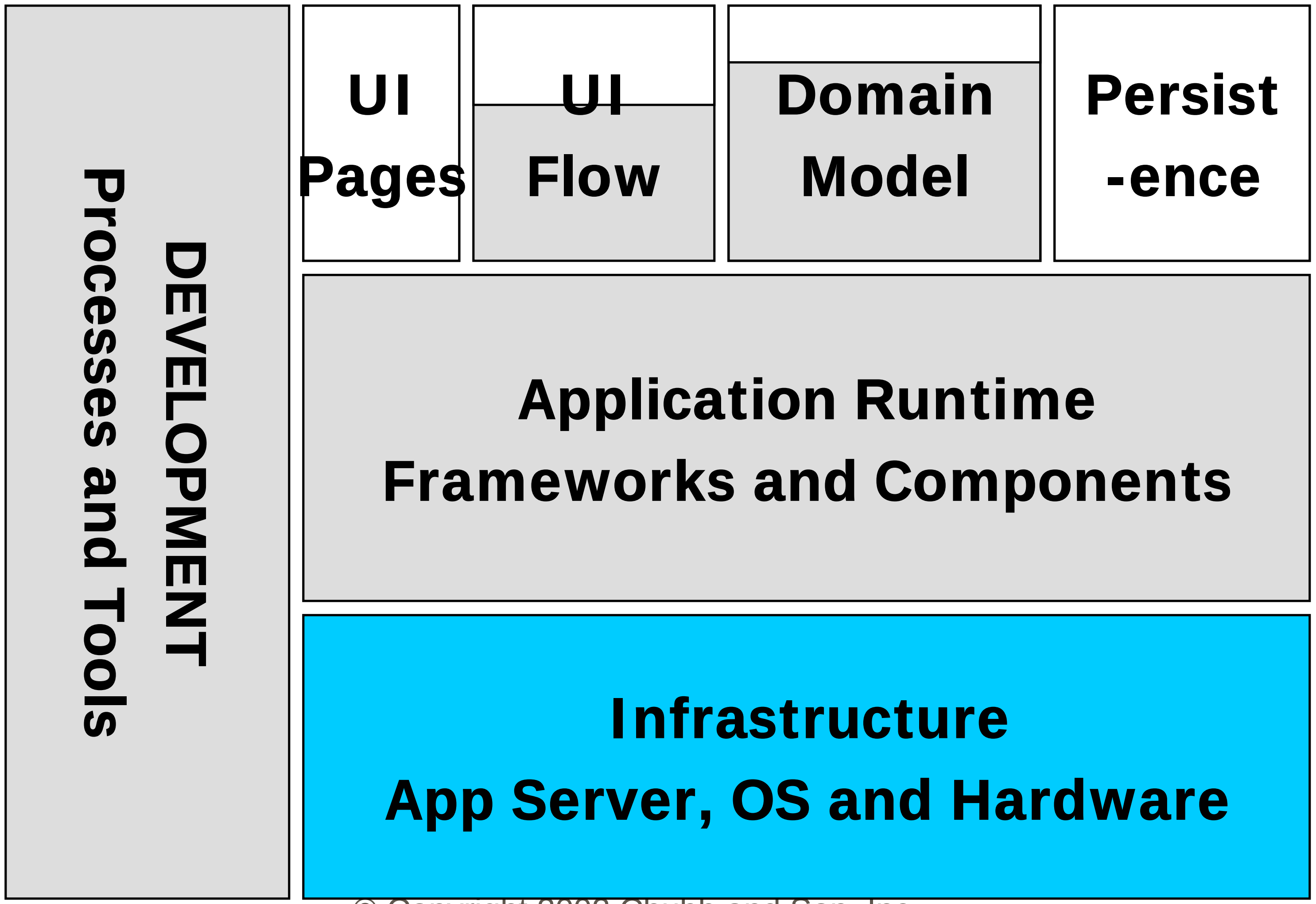
Intranet Application

- Results
  - Amended Architecture
  - Generated Amended Architecture
  - Generation was changed quickly
  - Consistent Architecture throughout system
  - New Developers using Generation
  - Only One MDA Architect
  - Input from Architects on System

Application Platform

- 2000 to Present
- Platform for Building and Deploying Thin-Client Applications
- Leveraged Previous Architecture
- Team of One Architect, Several Technical Analysts
- Support for UI Page Flow, Process and Domain Models

Application Platform



Application Platform

- UML Models
  - UI Page Flows - Class Diagrams
  - UI Pages - Class Diagrams
  - Domain - Class Diagram
- Generated
  - Strategies
  - Whole Values
  - XML Conduits

Application Platform

- Code Generation
  - Started With Homegrown Tool
  - Evaluated Vendor Tool
  - Adopted Vendor Tool
  - Converted Architecture to Vendor Tool
- Created Generalized Domain Model
- Developed Playbooks for Development Process

Application Platform

- Results
  - More Complete Generation
  - Vendor Generation Tool
  - Development Process
  - One Lead MDA Architect and One Back-up
  - Others learning tool and MDA thinking

Extranet App - Product 1

- 2001
- Accept and Pass App to Legacy System
- Team - One Architect and Two Technical Analysts (platform), 4 to 6 Developers
- Modeled/Generated UI Pages and Flow and Domain Model
- Concurrent with Platform Development
- Re-wrote hastily built system

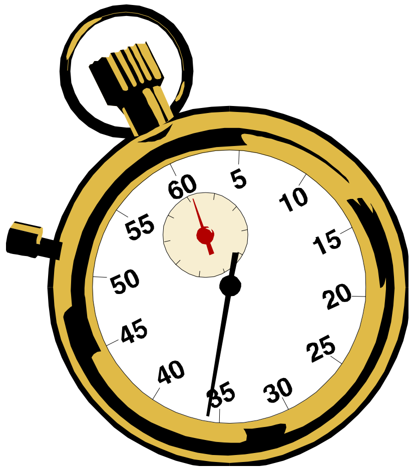
Extranet App - Product 2

- 2002
- Second Product Implemented on Platform
- Same Team
- Application Team Takes Control of Domain Model

Extranet Application

- Results
  - Successful Use of Platform
  - Consistent Architecture
  - Architecture/Platform Evolution did NOT Significantly Disrupt Development
  - On-Time and High Quality
  - Started Hearing the Term MDA
  - Back-up Personnel Learning More MDA

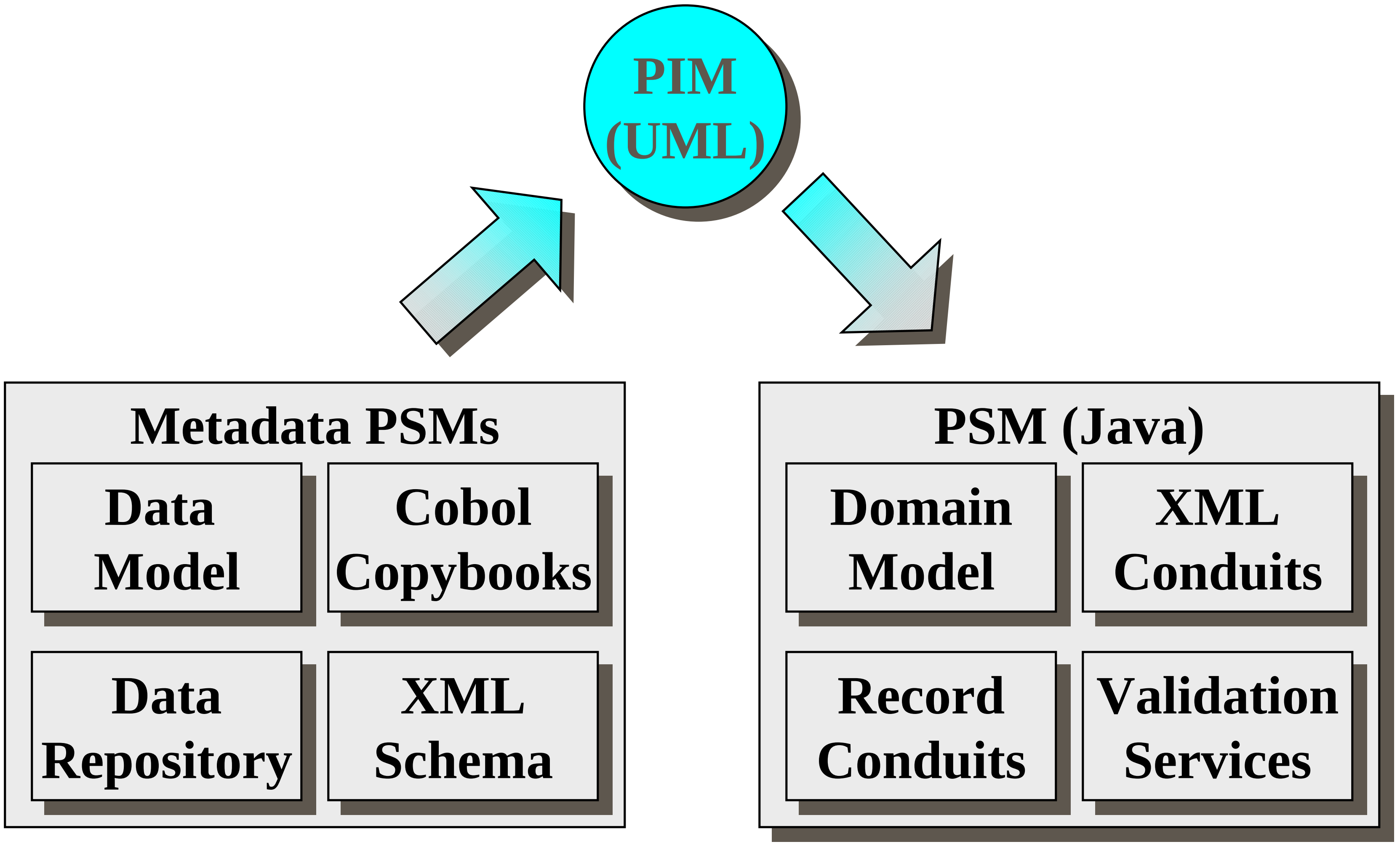
Current Projects



Integration Platform

- Development Process
  - Metadata to UML to Java
- Integration Model (CIM)
- Architecture
- Runtime Code
- Code Generation

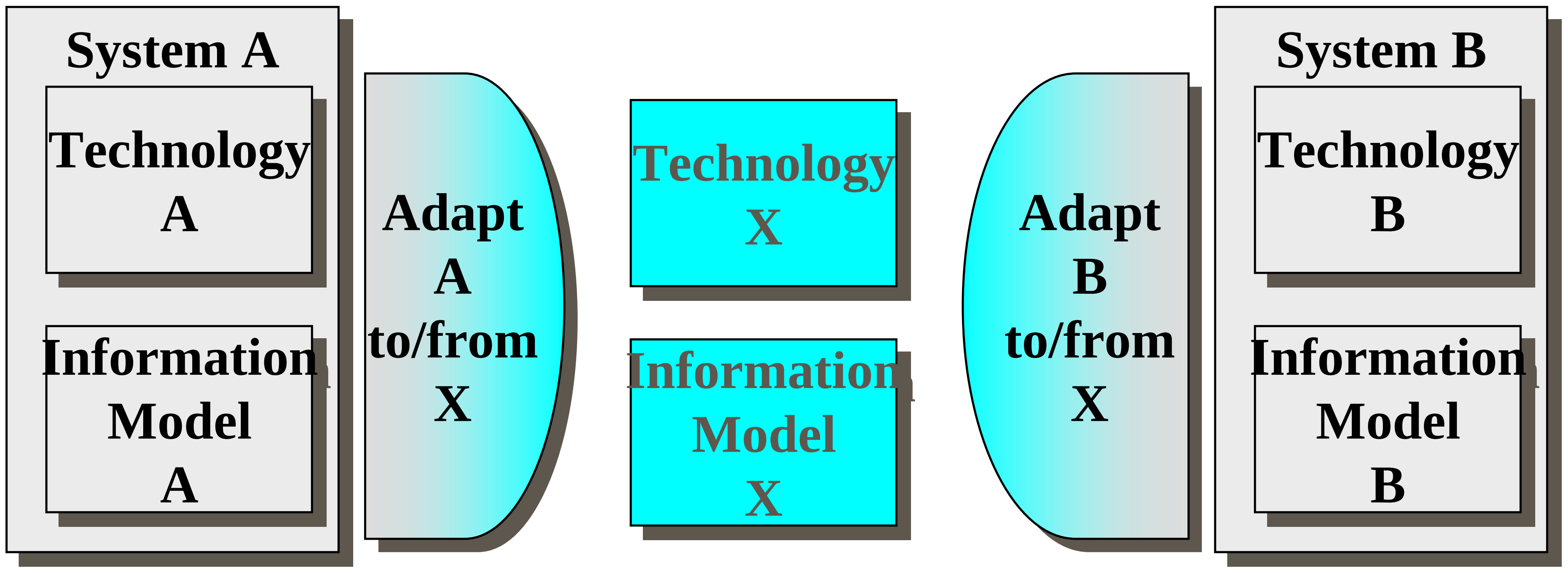
Integration Platform Development Process



Integration Model

- CIM Intermediate Document Format
  - Model based on an internal Operational Datastore (ODS) Format
  - Simplified for use in integration
  - Usable across Products
  - XML Schema Built Together with Data Group
  - Java implementation handles XML
  - Other implementations are possible

Integration Architecture



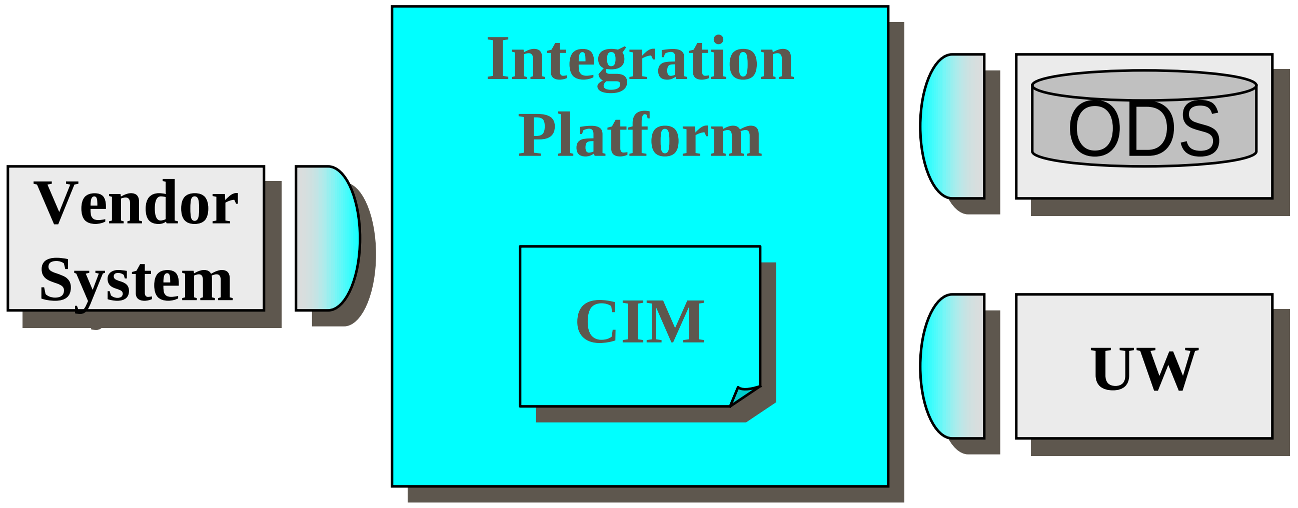
Integration Platform

- Runtime Code to support generated code
- Code Generation
  - Domain Classes - hold/validate information
  - XML Conduits - convert to and from XML
  - Record Conduits - convert to and from records
  - Process Classes - handle flow of processing based on event model

Integration Project 1

- Load ODS
  - Integrate with Extract-Transform-Load (ETL) Tool
  - Utilize CIM and Integration Platform
- PDF Filing
  - Extract from ODS and Vendor System Feed
  - Convert to PDF send to User Workstation App
  - Utilize CIM and Integration Platform

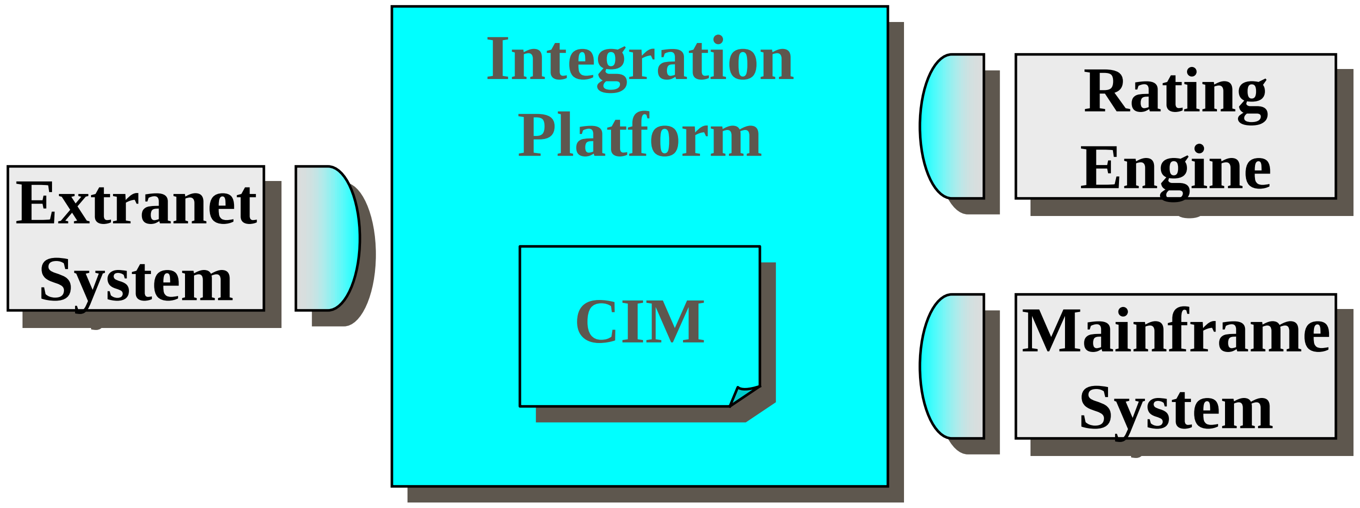
Integration Project 1



Integration Project 2

- Legacy System Upload
  - Replace HLLAPI
  - Load Database using MF Services
  - Use CIM and Integration Platform
- On-line Price Indicator
  - Integrate with Rating Engine
  - Use CIM and Integration Platform

Integration Project 2



Qualitative ROI

- Investment
  - Build Architecture
  - Learn Tools
  - Implement Architectures
  - Maintain Implementations
  - Maintain Tools

Qualitative ROI

- Return
  - Consistency
  - Maintanability
  - Productivity
  - Quality
  - Robustness

Conclusion

- We’ve been doing MDA-like development since 1998
- We’ve worked on several production applications
- We’ve evolved to using standards and vendor tools
- We have seen productivity and quality