An Early MDA Project … in Hindsight

Ernest Stambouly, Frank Truyen
Cephas Consulting Corp

635 E. 1st Street, Suite 317
Tustin, CA 92780
www.cephas.cc

Preview

- eCRM Enterprise Application Case Study.
- A Model-driven Approach.
- The architecture and UML models.
- The Approach.
- Approach Benefits.
- Lessons Learned.
- Conclusion.

Context

- **Timeframe (1997-2001)**
  - Pre-MDA; Middleware Battles; Young Java; Young UML™.
- **Industry**
  - eCRM
- **Application**
  - Management of communication channels for contact management.

Environment

- Small Software Development Team.
- Existing successful 2-tier Application.
- ***Greenfield*** Development
  - No Legacy integration upfront;
  - No Reverse Engineering;
  - No Harvesting of Design or Code.

Enterprise Requirements

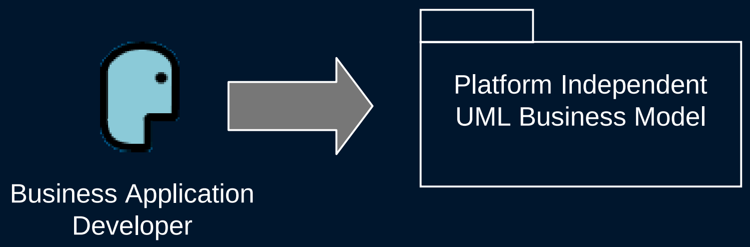
- Large-Scale, distributed & heterogeneous targets.
- Standard-based.
- Ease of adding business functionality.
- Capture intellectual assets separate from application.
- Data interchange with external entities.
- Independence of underlying technologies.
- Development in ***Internet Time***.

Enterprise Decisions

- Invest in Enterprise Architecture
  - Distributed, scalable, platform independent;
  - Separates Business and Technical concerns;
  - Uses formal specification language (UML);
- Restructure Development Organization
  - Architecture/Infrastructure Developer;
  - Business Application Developer;
  - UI & General Developer;

Separation of Concerns
Business

- Standalone specification of Business Functionality
  - Formal Business Model;
  - Platform Independent;
  - Business code evolves independently of infrastructure;



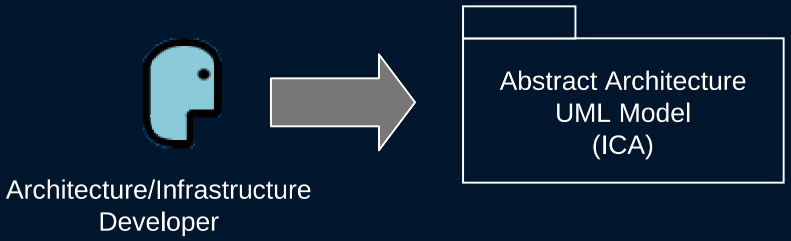
Business Application Developer

Platform Independent UML Business Model

Separation of Concerns

Technology

- Standalone UML specification of Abstract Architecture (dubbed ICA)
  - Defines a formal computing model;
  - Abstract service-based architecture;
  - Independent of underlying technologies;
  - Can be projected to different underlying technologies;



Abstract Architecture
UML Model
(ICA)

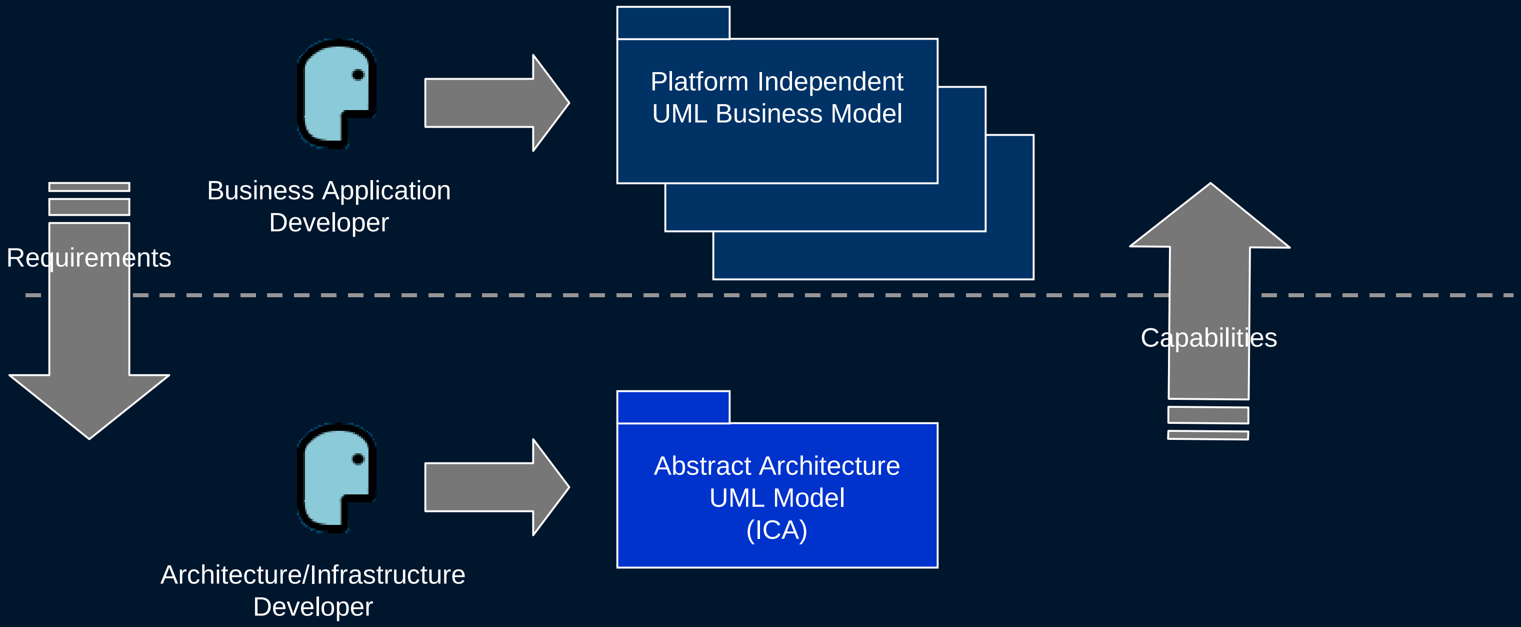
Architecture/Infrastructure
Developer

Formal Specification

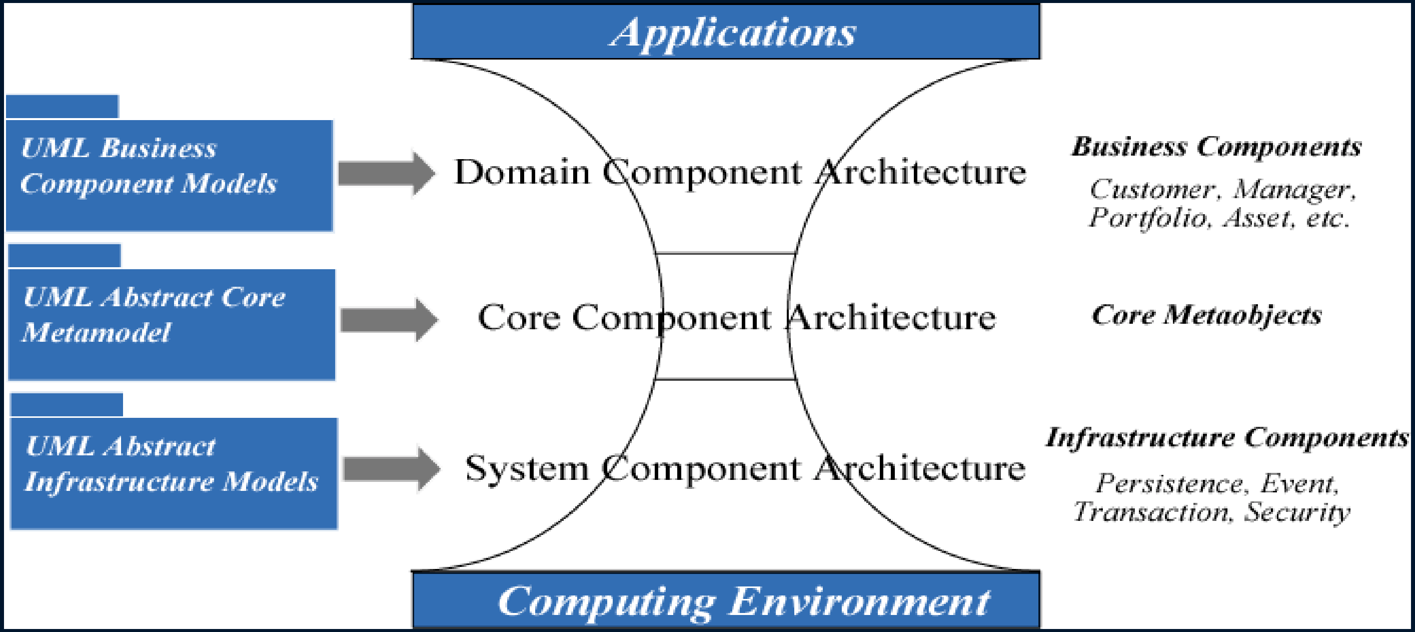
- Uses formal specification language: UML.
- Everything is Model-Driven.
- Standard/Popular OOA&D Methods.
- Development Environment
  - Integrates suite of development and model transformation tools;
  - Promotes model-driven as the “natural” approach to development;

Development Organization

- Reflects development approach


Platform Independent UML Business Model
Business Application Developer
Requirements
Capabilities
Abstract Architecture UML Model (ICA)
Architecture/Infrastructure Developer

The Architecture



Applications

UML Business Component Models

UML Abstract Core Metamodel

UML Abstract Infrastructure Models

Domain Component Architecture

Core Component Architecture

System Component Architecture

Computing Environment

Business Components

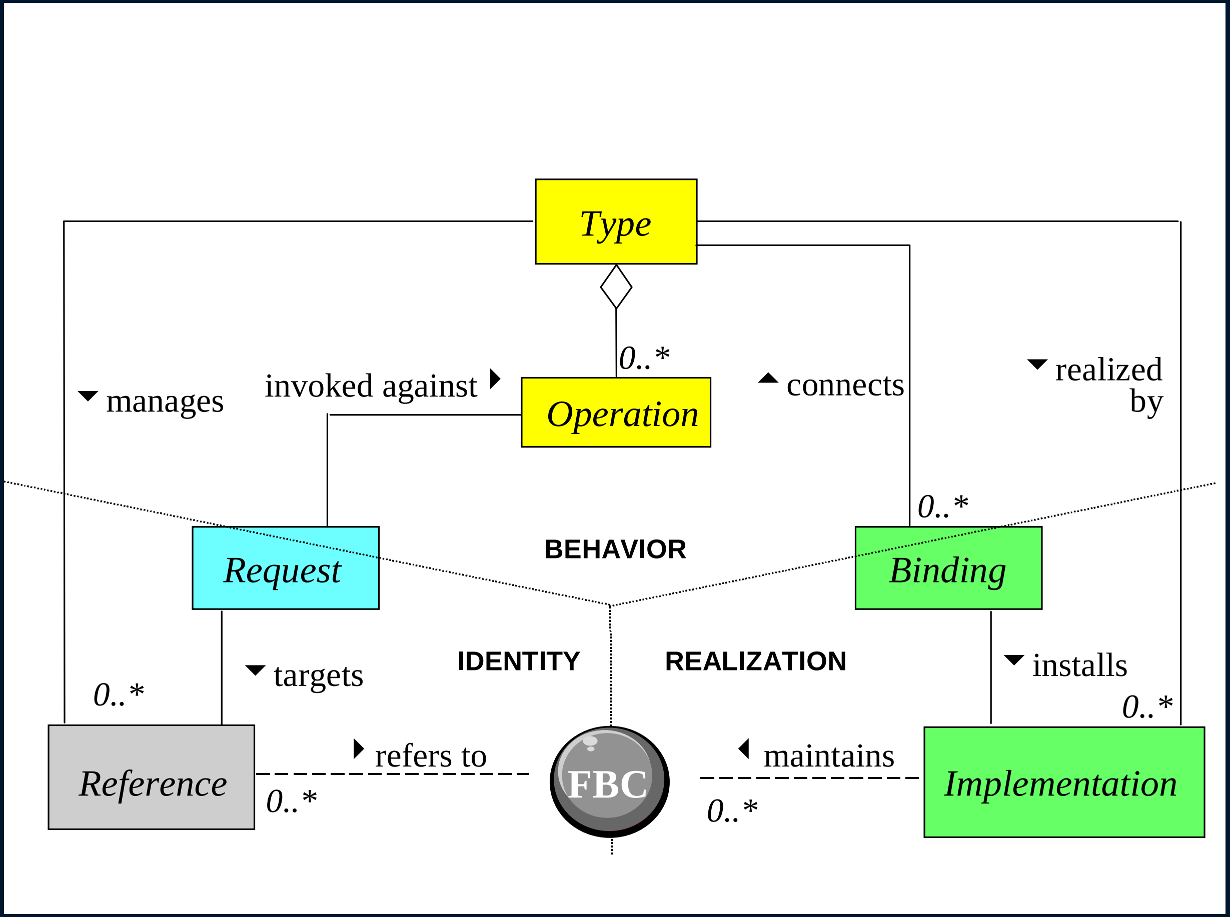
Customer, Manager, Portfolio, Asset, etc.

Core Metaobjects

Infrastructure Components

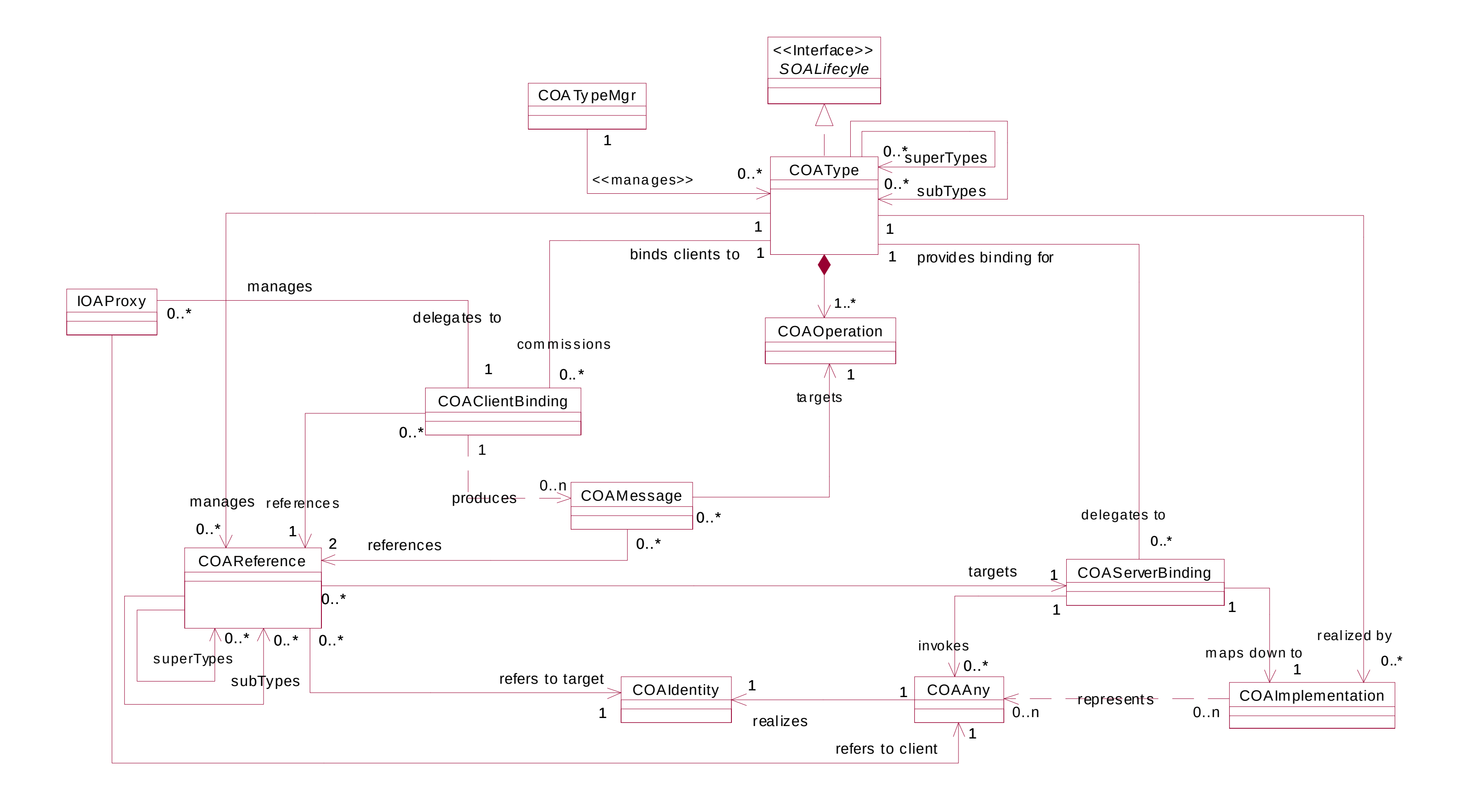
Persistence, Event, Transaction, Security

UML Abstract Core Metamodel



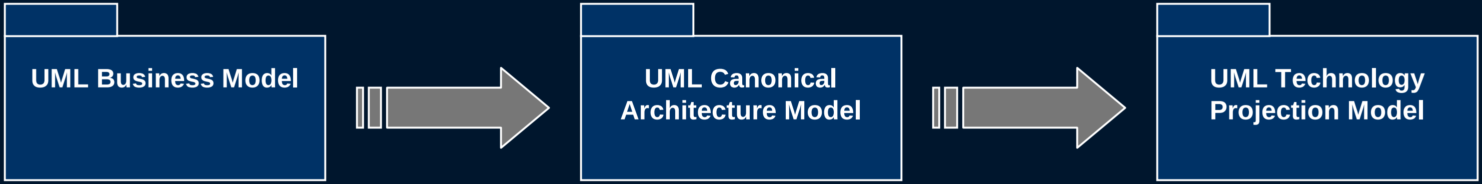
9/20/2002

Refined Core Metamodel



9/20/2002

Model Transformation Stages



UML Business Model

UML Canonical Architecture Model

UML Technology Projection Model

(PIM)

(PIM)

(PSM)

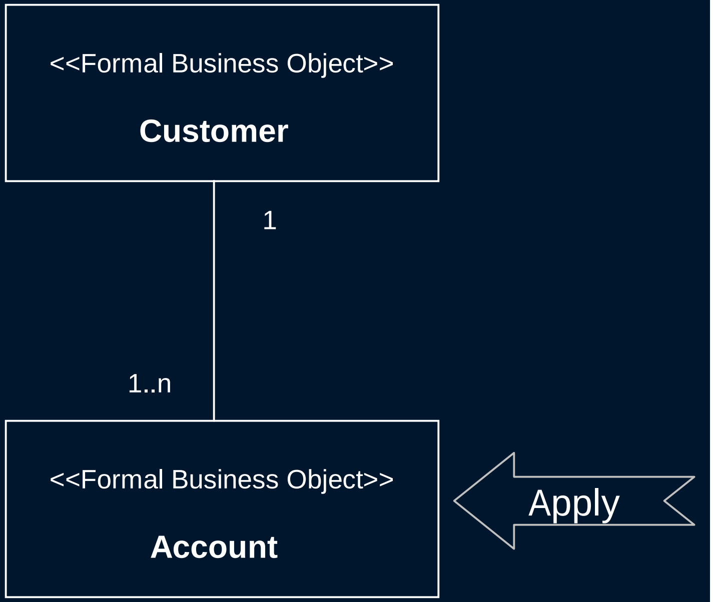
- Build Business Model;
- Designate Business Components using UML stereotypes & property sheets;
- Transform;

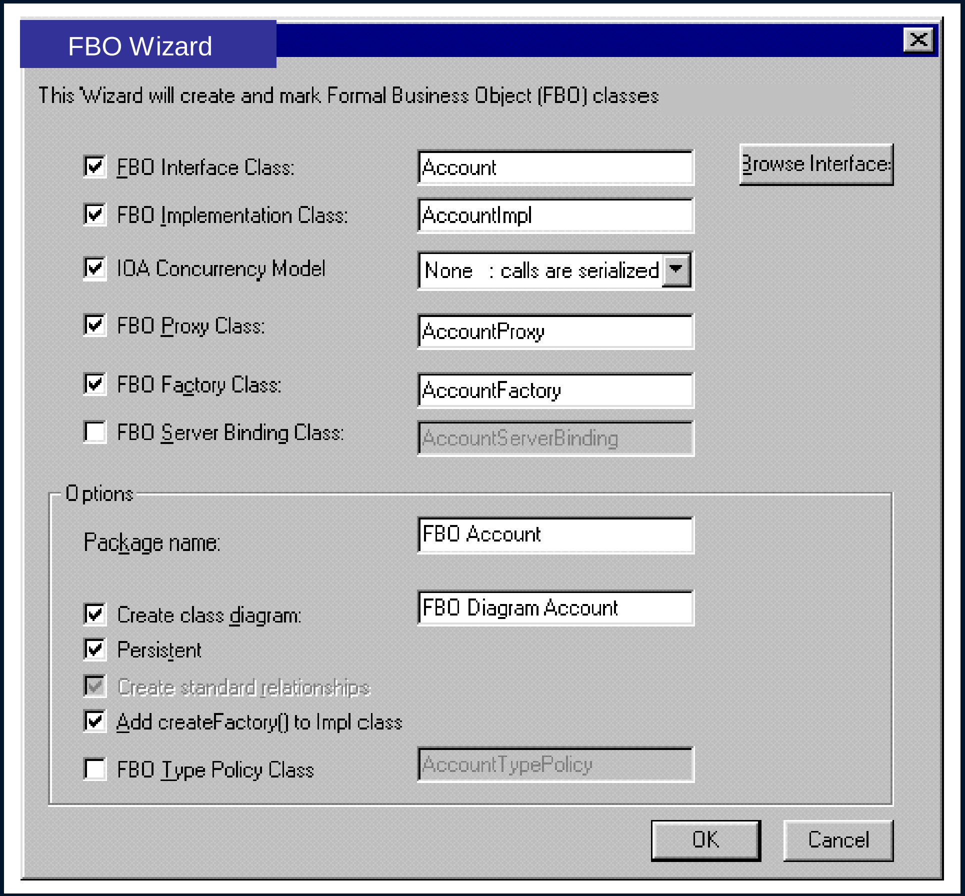
- Refined Business Model;
- Augmented with instances of the Core metamodel;
- Independent of underlying technology;
- Generate canonical model repository (XML);

- Platform-specific, e.g., CORBA, COM, J2EE, …
- Generate code from this model;
- Implement business code in dedicated “Impl” generated classes.

Sample Models

UML Business Model





FBO Wizard

This Wizard will create and mark Formal Business Object (FBO) classes

☑ FBO Interface Class:	Account	Browse Interface
☑ FBO Implementation Class:	AccountImpl	
☑ IOA Concurrency Model	None : calls are serialized	
☑ FBO Proxy Class:	AccountProxy	
☑ FBO Factory Class:	AccountFactory	
☐ FBO Server Binding Class:	AccountServerBinding	

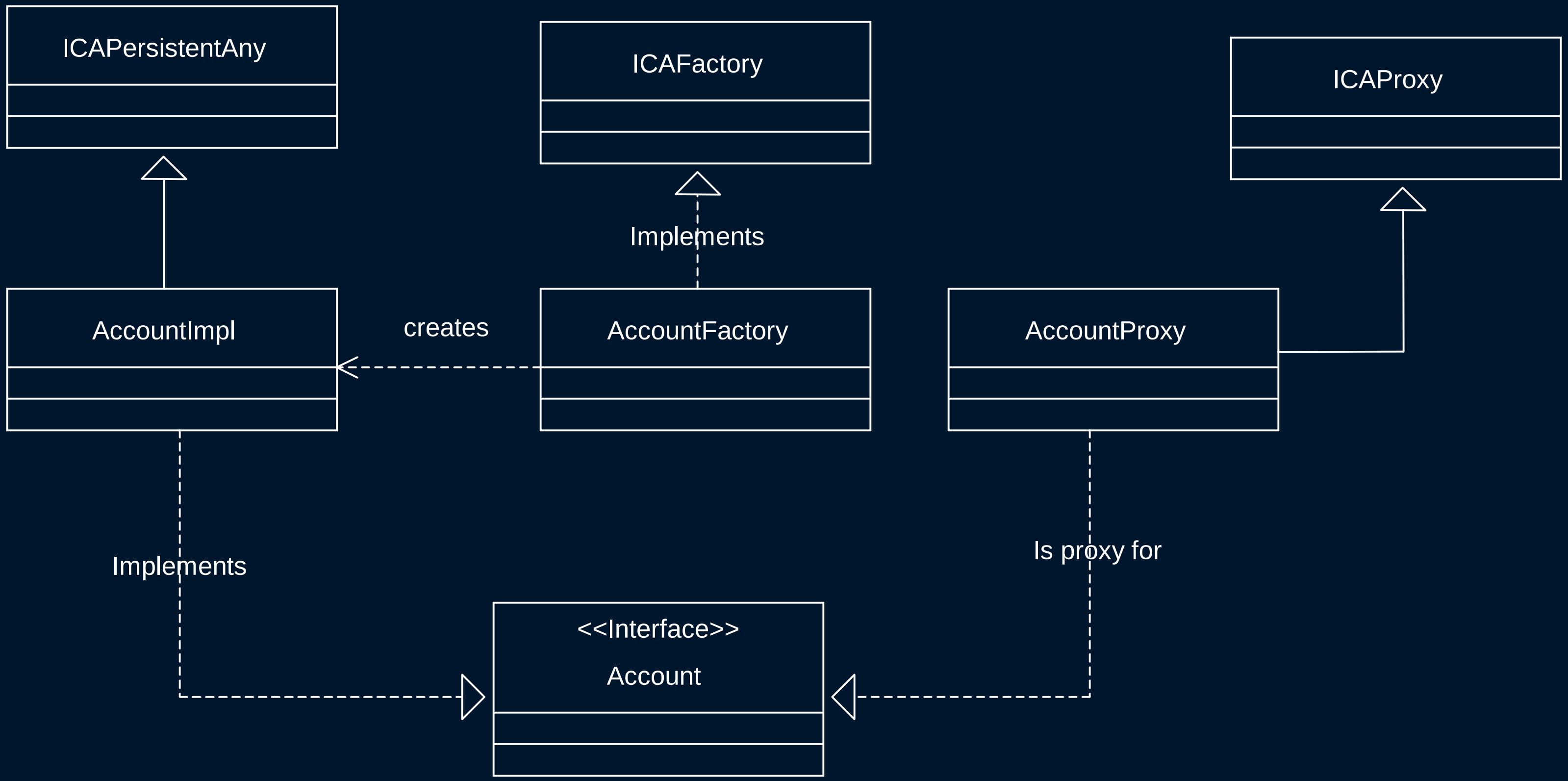
Options

Package name:	FBO Account
☑ Create class diagram:	FBO Diagram Account
☑ Persistent	
☑ Create standard relationships	
☑ Add createFactory() to Impl class	
☐ FBO Type Policy Class	AccountTypePolicy

OK Cancel

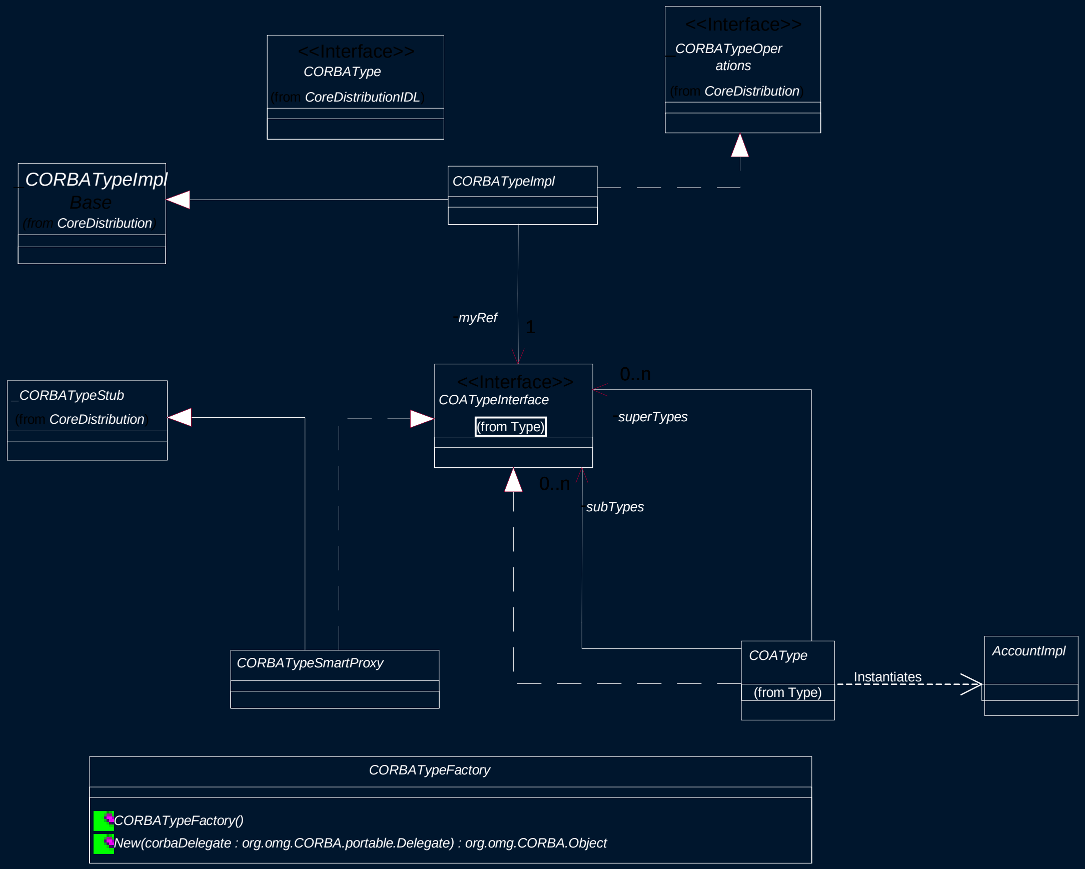
Sample Models

Refined Business Model



Sample Models

CORBA® Specific Model



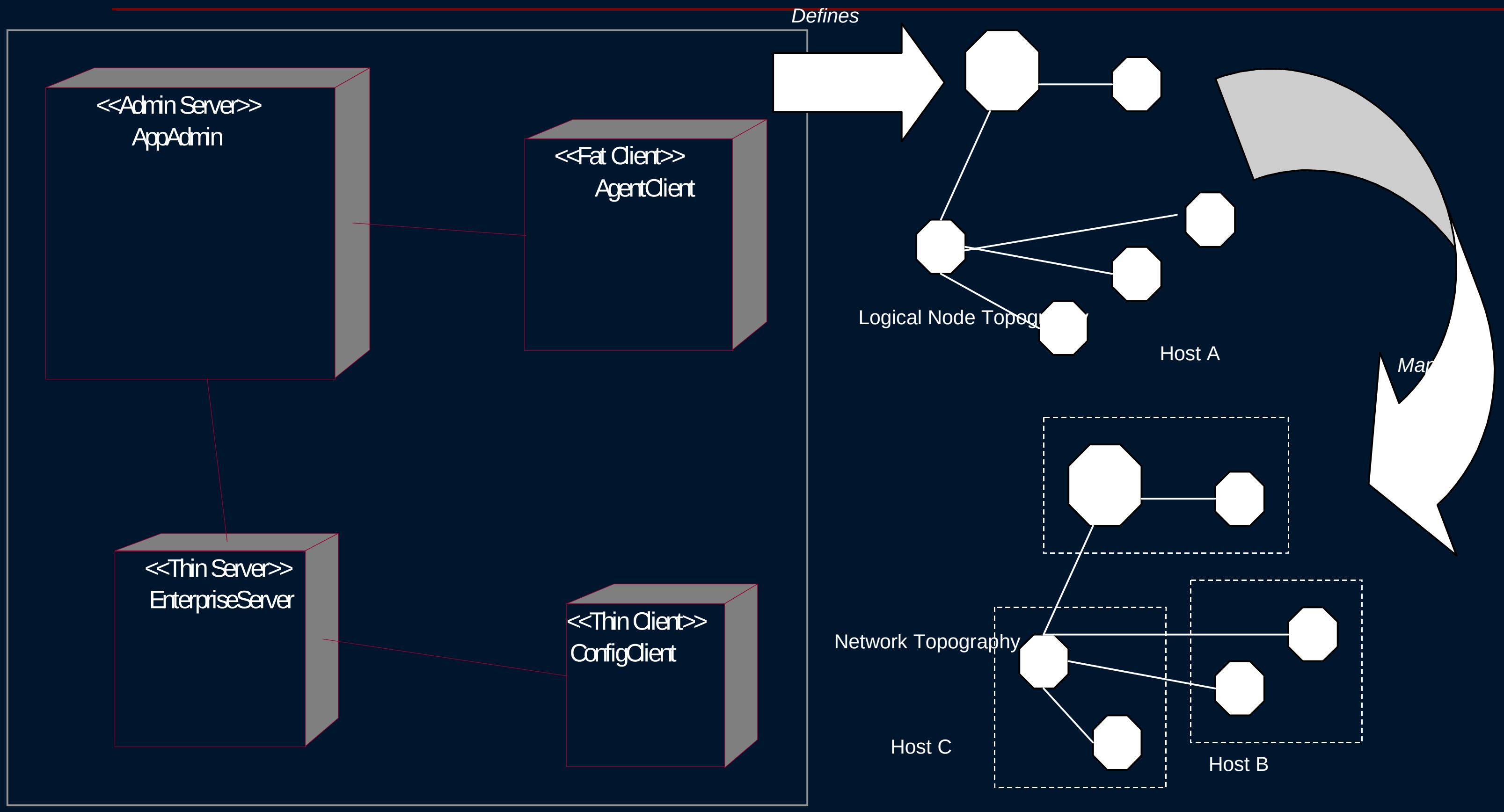
Business Domain Definition
XML Representation

```
<Class> Customer
        <Operation> approvePurchase
            <Returns Type="boolean"/>
            <Parameter Name="purchase" Type="Long" Mode="In"/>
            <Parameter Name="creditLeft" Type="LongHolder" Mode="Out"/>
        </Operation>
          <Operation> getStatus
            <Returns Type="void"/>
            <Parameter Name="nameHold" Type="StringHolder" Mode="Out"/>
            <Parameter Name="creditHold" Type="LongHolder" Mode="Out"/>
            <Parameter Name="totOrdHold" Type="DoubleHolder" Mode="Out"/>
            <Parameter Name="discHold" Type="FloatHolder" Mode="Out"/>
        </Operation>
        …
        <ClientBinding Name="Acme.BusinessTypes.CustomerProxy"/>
        <Implementation Name="Acme.BusinessTypes.CustomerImpl" Concurrency="Full"
Transaction="NotSupported" FactoryImplicit="False" FactoryActivate="Update"
 NodeId="SunServer"/>
        <PersistentDataImpl Name="Acme.BusinessTypes.CustomerData"/>
        <PersistentStateImpl Name="Acme.BusinessTypes.CustomerState"/>
        <Persistent Name="OracleDB1"/>
        …
    </Class>
```

Deployment Model

- UML Deployment View to model logical topography.
- Generate Deployment Descriptors.
- Mapping tools to physical network.

Deployment Transformation



Application-level Deployment Specification

The Reality

- Project is a success and in production today.
- Most of the effort was spent on building tools and the infrastructure.
- Crude homegrown automation and transformation tools.
- Need for more powerful modeling expressiveness of non-functional features.

What This Means

- Key to MDA realization:
  - Tools! Tools! Tools!
  - Upgraded Skill set;
- As an early adopter, experiment with MDA.

Benefits

- High returns from reuse.
- Application developers focus on business development.
- Ease of migration to other underlying technologies.
- Seamless addition of ancillary services
  - Remote Debugging; Instrumentation; Logging;

Lessons Learned

- Invest in tools – don't build from scratch.
- Effort requires management buy-in.
- Performance Overhead of infrastructure layer is negligible.
- Mentoring and Training technical staff
  - Architecture philosophy, approach & patterns;
  - Design by Contract;

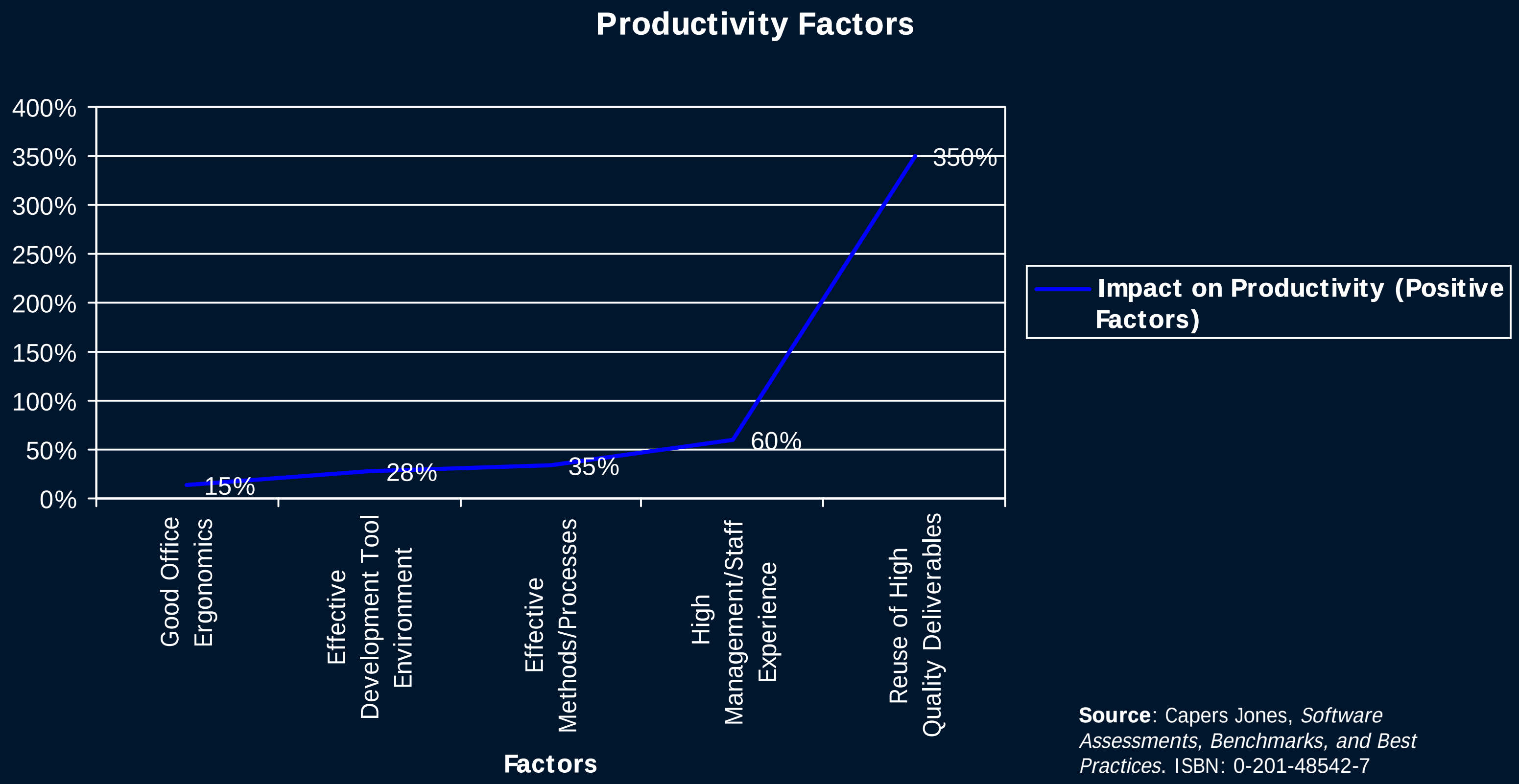
More Lessons Learned

- Appropriate Development Environment.
    - Tools! Tools! Tools!
    - Well-integrated tools suite that supports the full development lifecycle;
    - Automation and optimization of software development tasks upfront;
- A “natural” tool environment for MDA.

Closing Point

- To MDA or not to MDA?
  - Does it work?
  - Measurable improvements?
  - …
- Reusable enterprise infrastructure across projects

Impact on SD Productivity



Source: Capers Jones, *Software Assessments, Benchmarks, and Best Practices*. ISBN: 0-201-48542-7

Questions?