

# **Model Driven Legacy Integration™**

**OMG Web Services Workshop 2003**

**Rüdiger Schilling, CEO**  
**Delta Software Technology GmbH**  
**[rschilling@d-s-t-g.com](mailto:rschilling@d-s-t-g.com)**

# Model Driven Legacy Integration

**Overview**

**The Task**

**From MDA to MDLI**

**Generated PSM and Code**

**Tool Support**

**Summary**

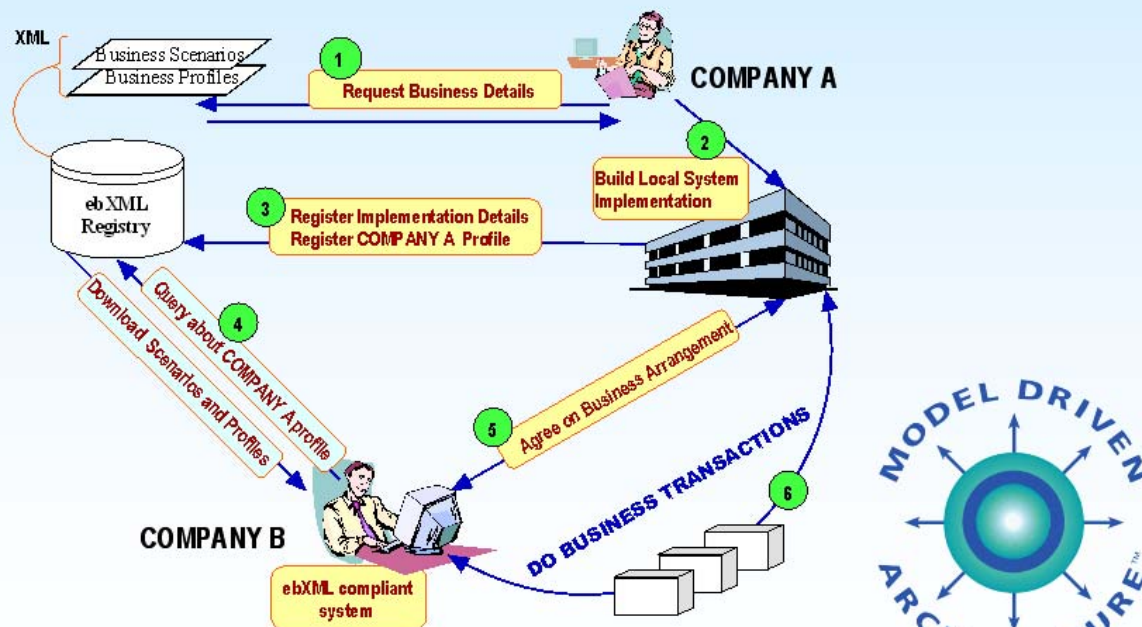
# Model Driven Legacy Integration



## The Task



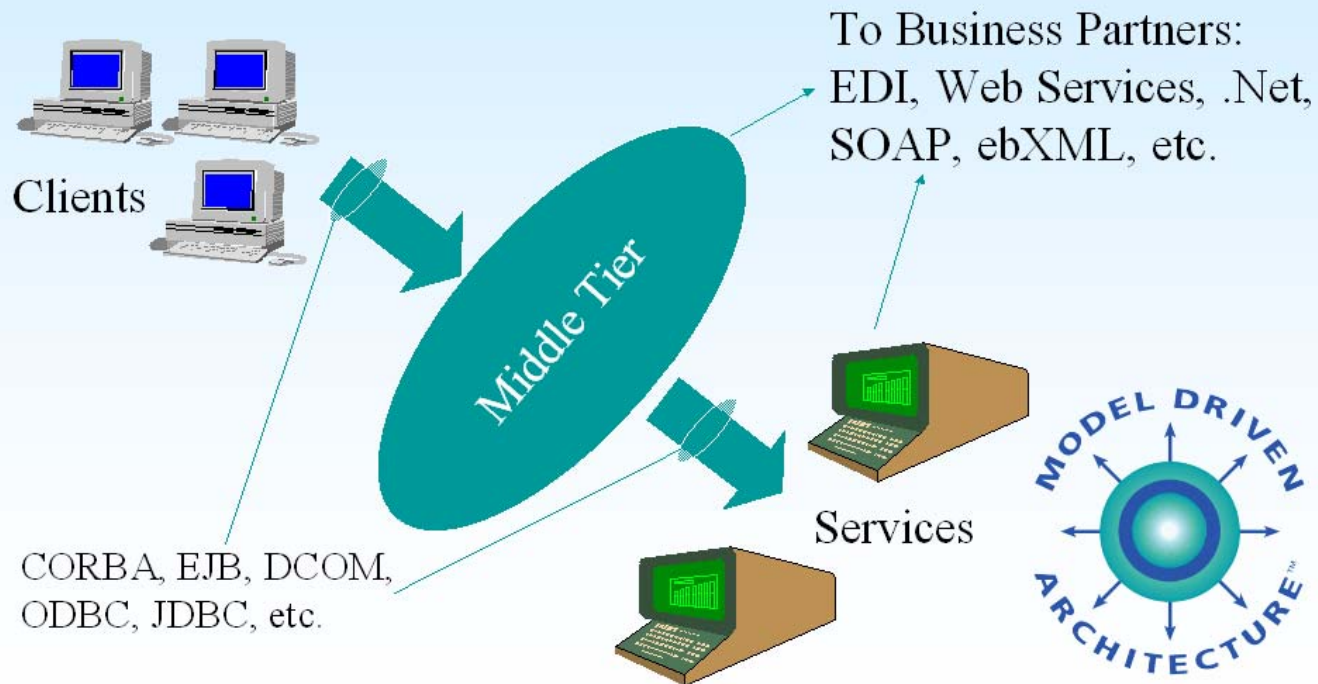
# The Dream: Web Services



(Clipped from ebXML Technical Architecture)



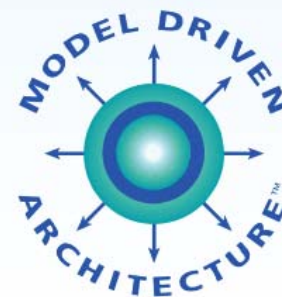
## The Reality: Integration


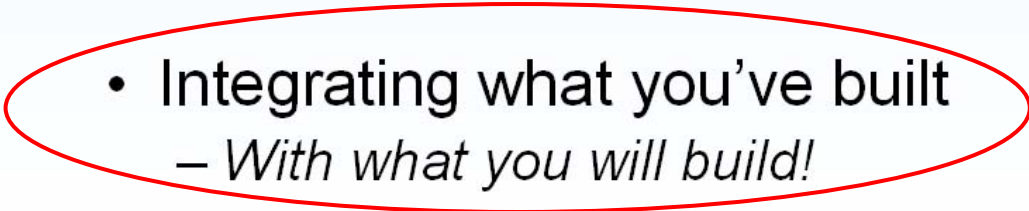




## How Can We Protect Software Investment?

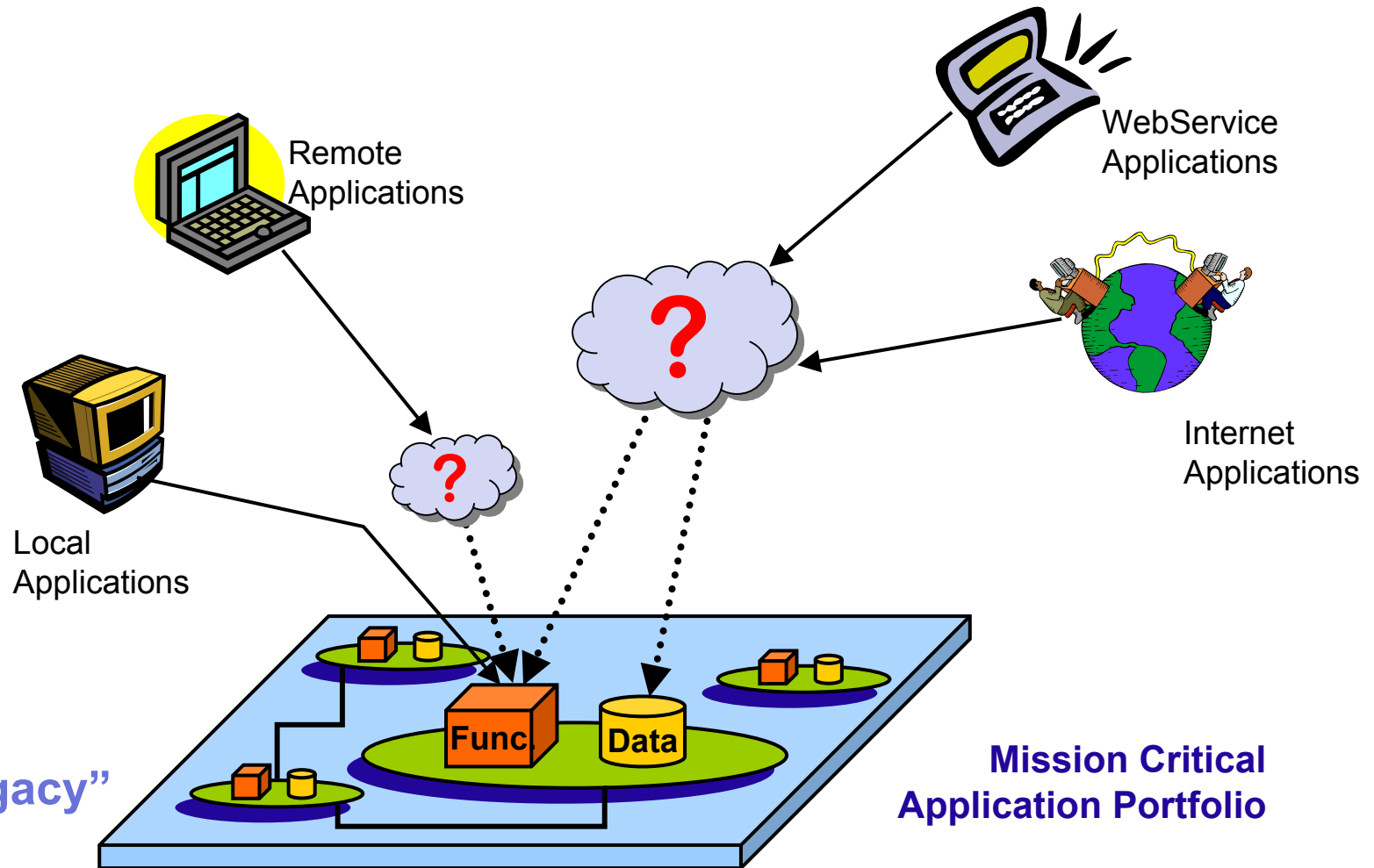
- The problem remains
  - Tracking the *next best thing*
  - Protecting your investment in existing software base
  - Retaining qualified staff
  - Maintaining existing code base



- 
- 
- Integrating what you've built
    - *With what you will build!*

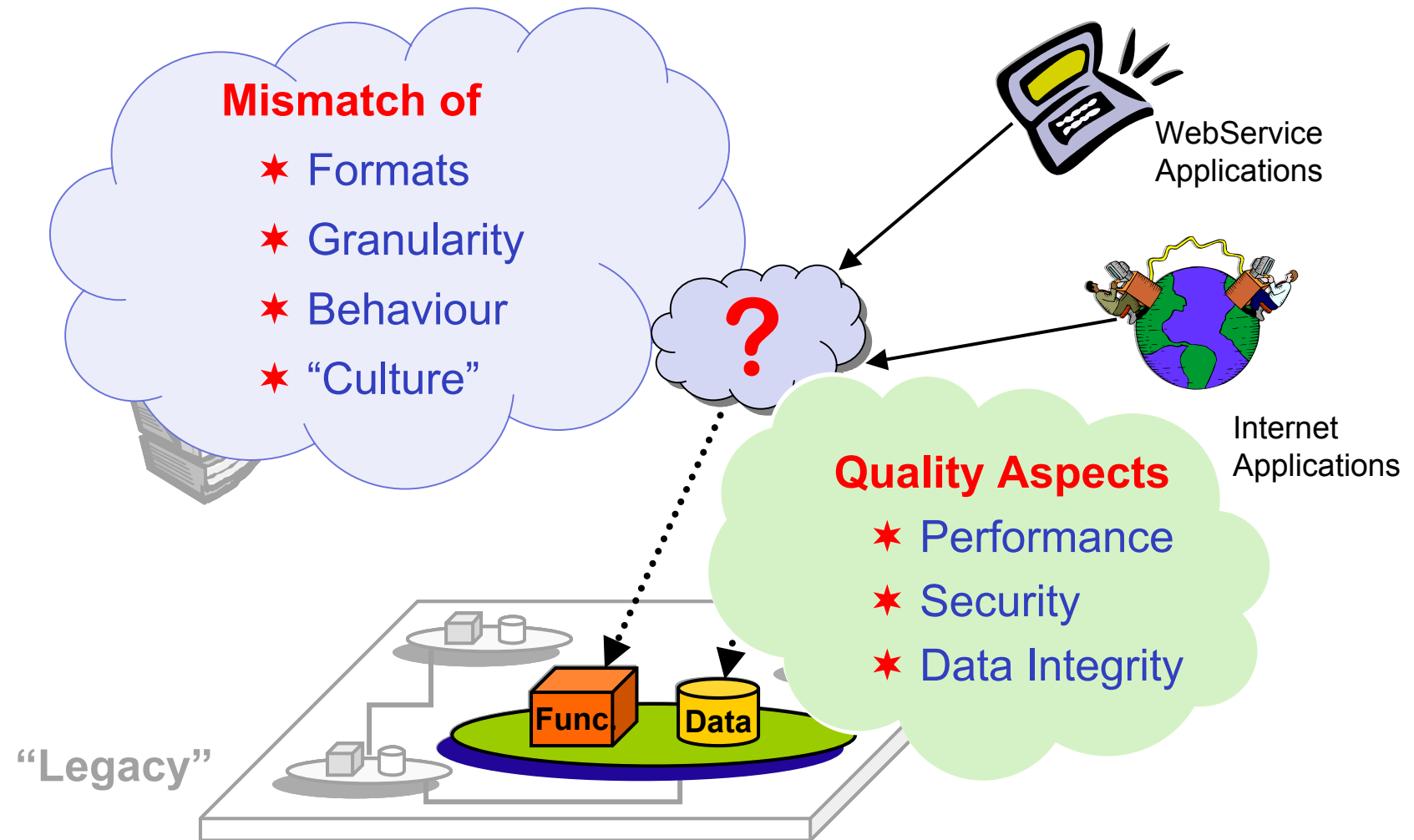
# Model Driven Legacy Integration

## The Task



# Model Driven Legacy Integration

## The Task





# Mismatches to Overcome

## ■ Formats

- Mapping of data types and data formats
- Flattening of groups, overlay structures and arrays
- Various message formats
- Various middleware APIs

## ■ Granularity

- Conventional function and call interfaces with coarse grain interfaces → sometimes more than 100 KB per call
- Fine grain OO interfaces → set/get methods
- (R)DBMS interfaces with insufficient encapsulation
- No single perfect interface
- "Multi-grain" interfaces for Web Services

# Mismatches to Overcome

## ■ Behaviour

- Transactions vs. sessions
- Stateful servers vs. stateless clients and middle tier
- Tightly vs. (very) loosely coupled systems

## ■ “Culture”

- Client developers and server developers (on legacy platforms) think in different concepts and languages
- A Java programmer hardly accepts a bean that looks like COBOL ...
- ... and vice versa

# Quality Aspects to Observe

## ■ Performance

- Mission critical, enterprise class applications with high transaction volumes
- No impact on existing users of systems allowed

## ■ Security

- Integration into existing security architectures
- Auditing, Logging

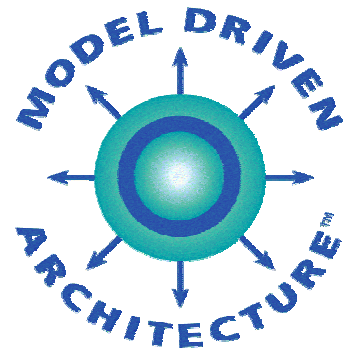
## ■ Data Integrity

- Managed access to data
- No J/ODBC to production database

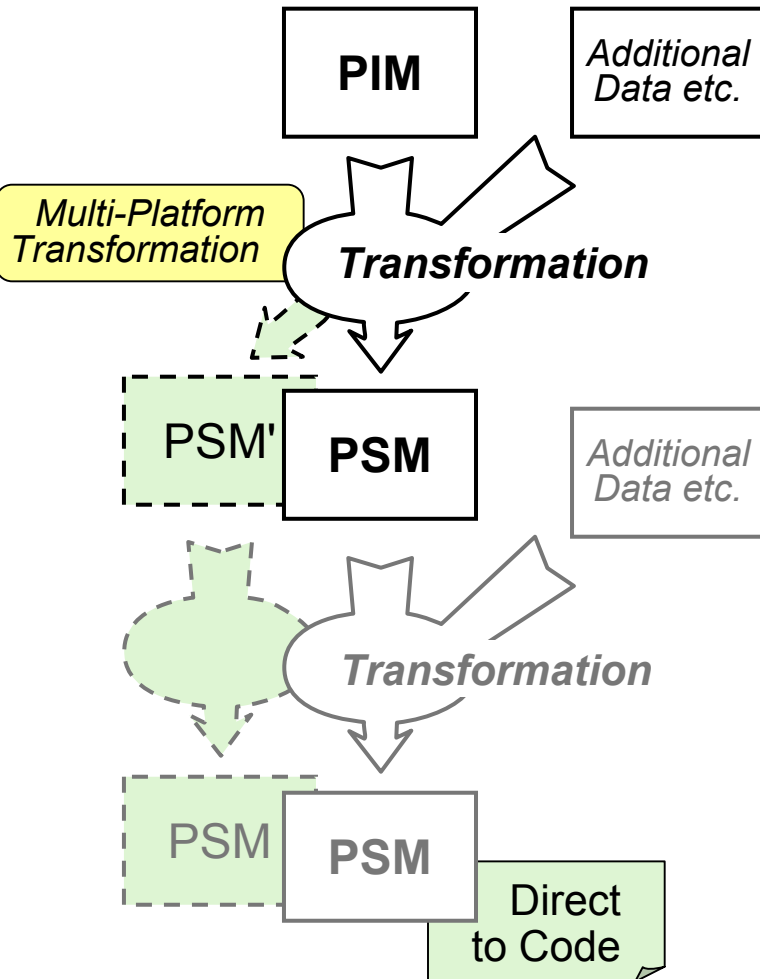
# Model Driven Legacy Integration



**From MDA to MDLI**



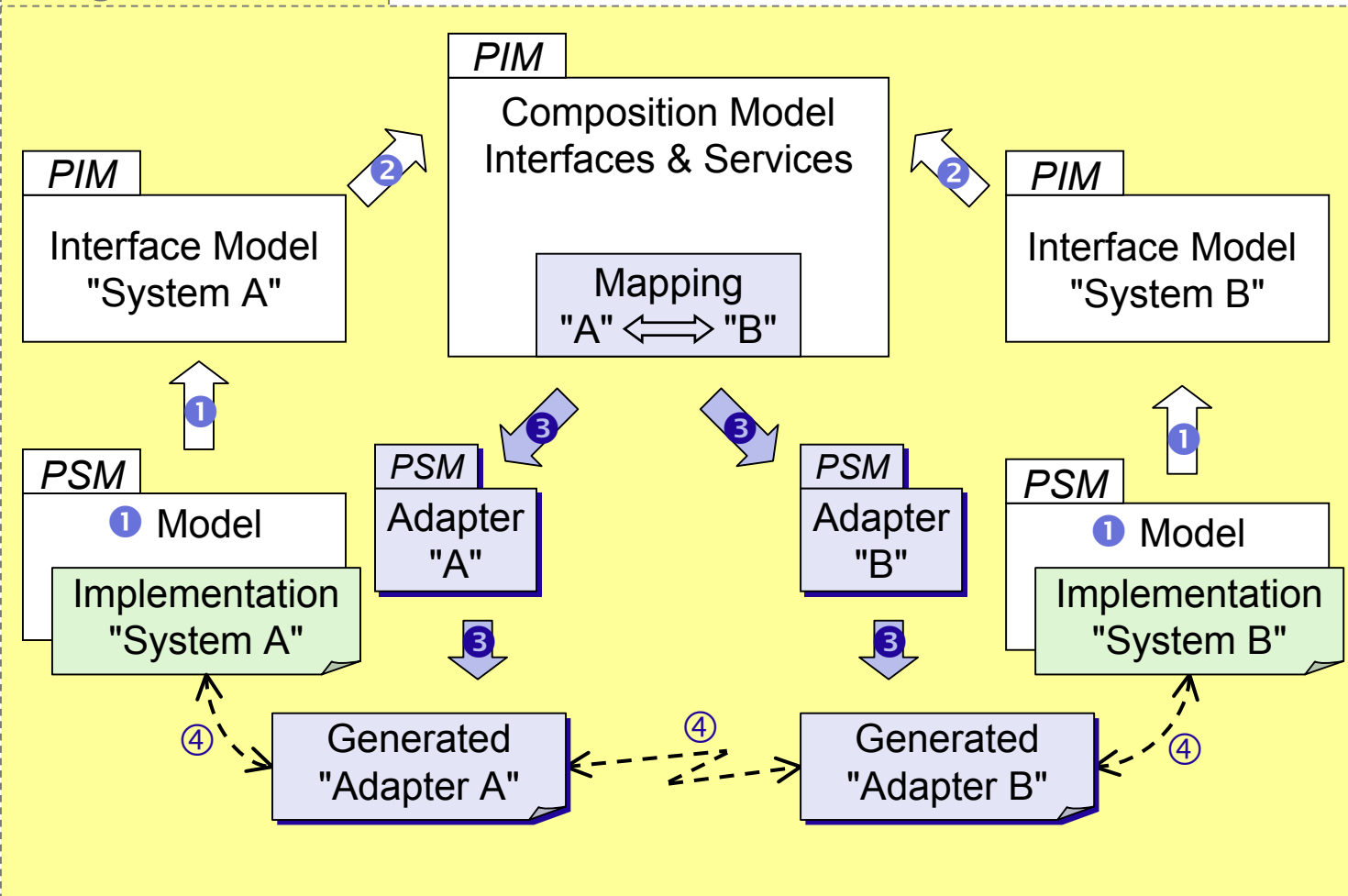
# MDA Transformation Pattern



- **Transformation (Mapping)**
  - Automatic, using generators
  - Optionally controlled by additional data (marks, parameters etc.)
- **Different transformation concepts**
  - Templates, patterns ...
  - Generator technology itself is not part of the standards process*
- **Multi-platform transformation**
- **Direct transformation**
  - PIM/PSM to Code

# Integration PIMs and PSMs (1)

## Integration Model



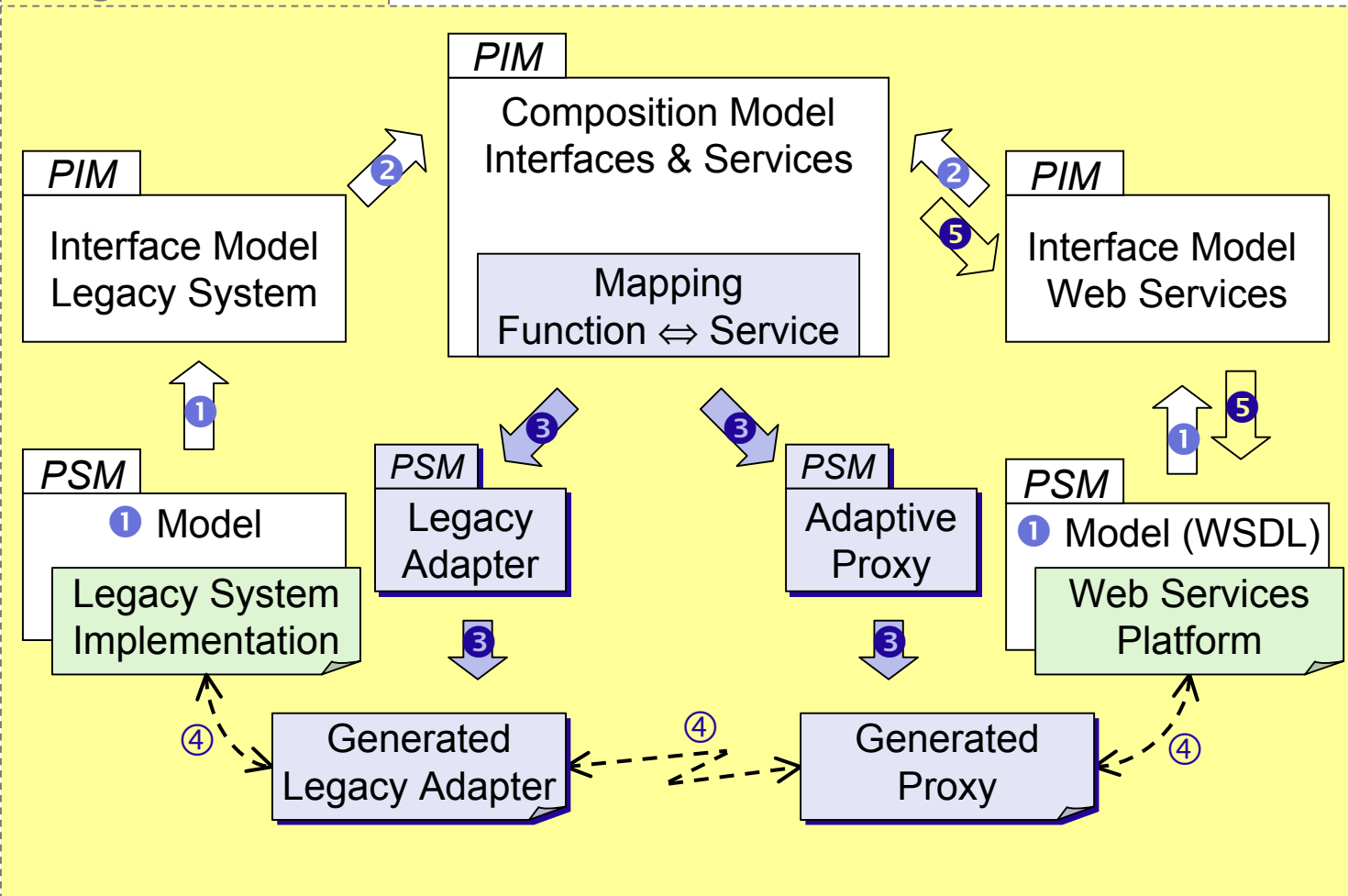
*"Horizontal"*  
integration

*e.g. of two  
application  
systems*

- ① *Discovery*
- ② *Composition*
- ③ *Production  
via Generator*
- ④ *Runtime*

# Integration PIMs and PSMs (2)

## Integration Model



*“Vertical”  
integration*

*e.g. of a legacy  
system into  
Web Services  
Platform*

① *Discovery*

② *Composition*

③ *Production  
via Generator*

④ *Runtime*

⑤ *Alternative  
approach*

# Platform Independent Integration Model

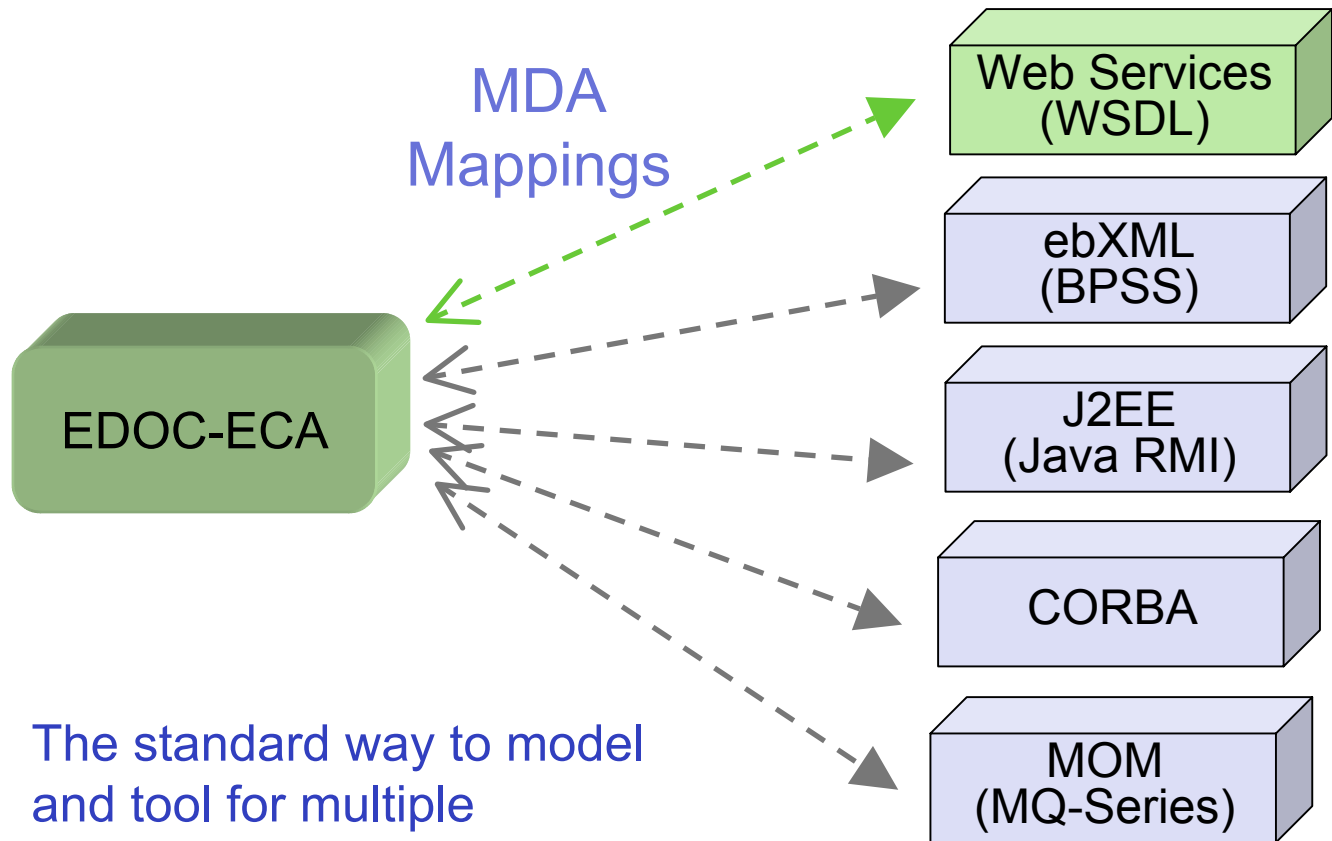
- Based on
  - **OMG Standard EDOC-ECA/CCA**
    - Enterprise Distributed Object Computing  
Enterprise/Component Collaboration Architecture
  - **That defines how**
    - “... classes, collaborations and activity graphs are used to model at varying and mixed levels of granularity the structure and behaviour of the components that comprise a (distributed) system ... ”
- **Well suited for**
  - Integration PIMs
  - Service Oriented Architectures (SOAs),  
e.g. Web Services



# From MDA to MDLI

## OMG-Standard EDOC-ECA/CCA

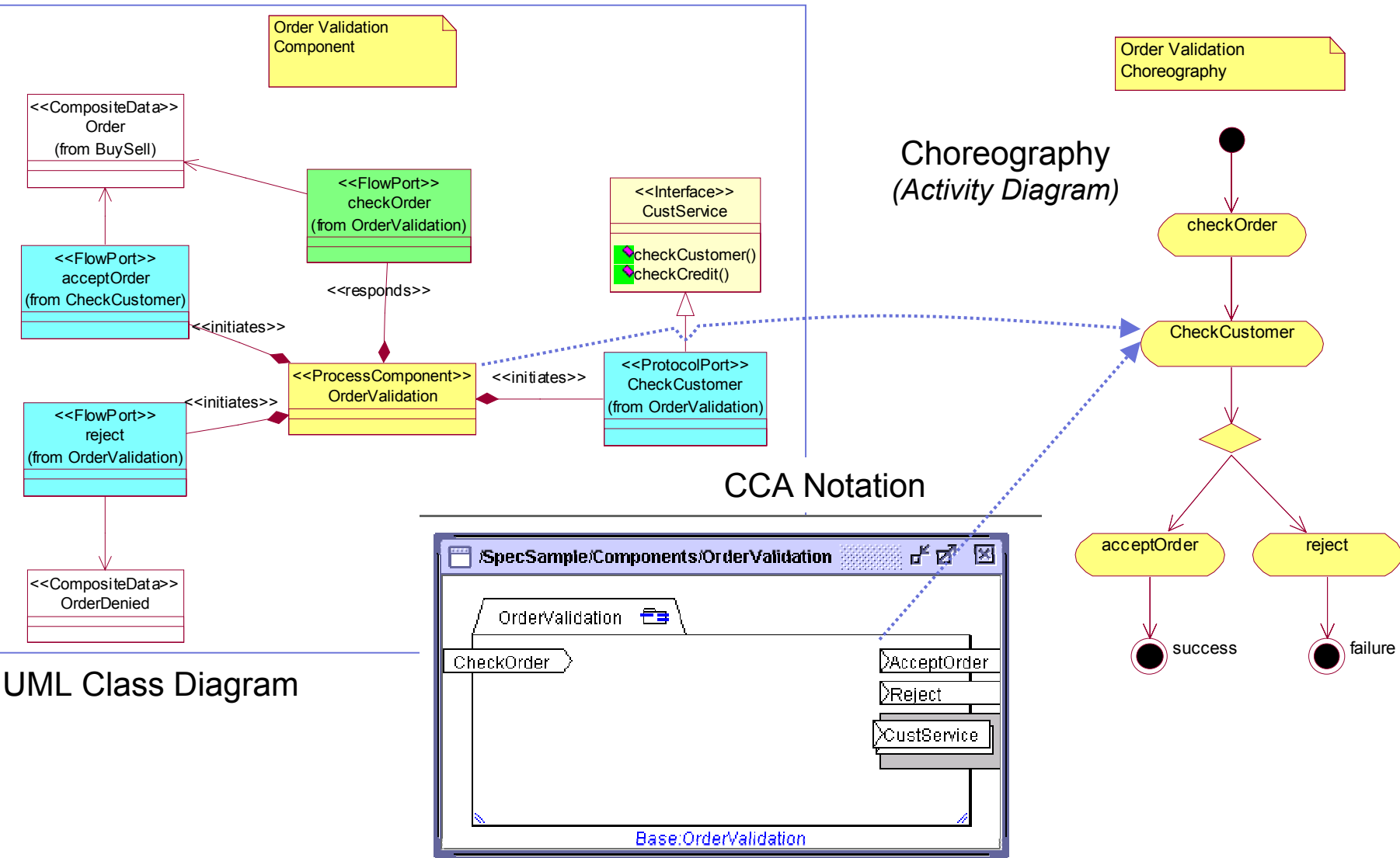
### ECA as the normal form



Source: Data Access Technologies, Inc.

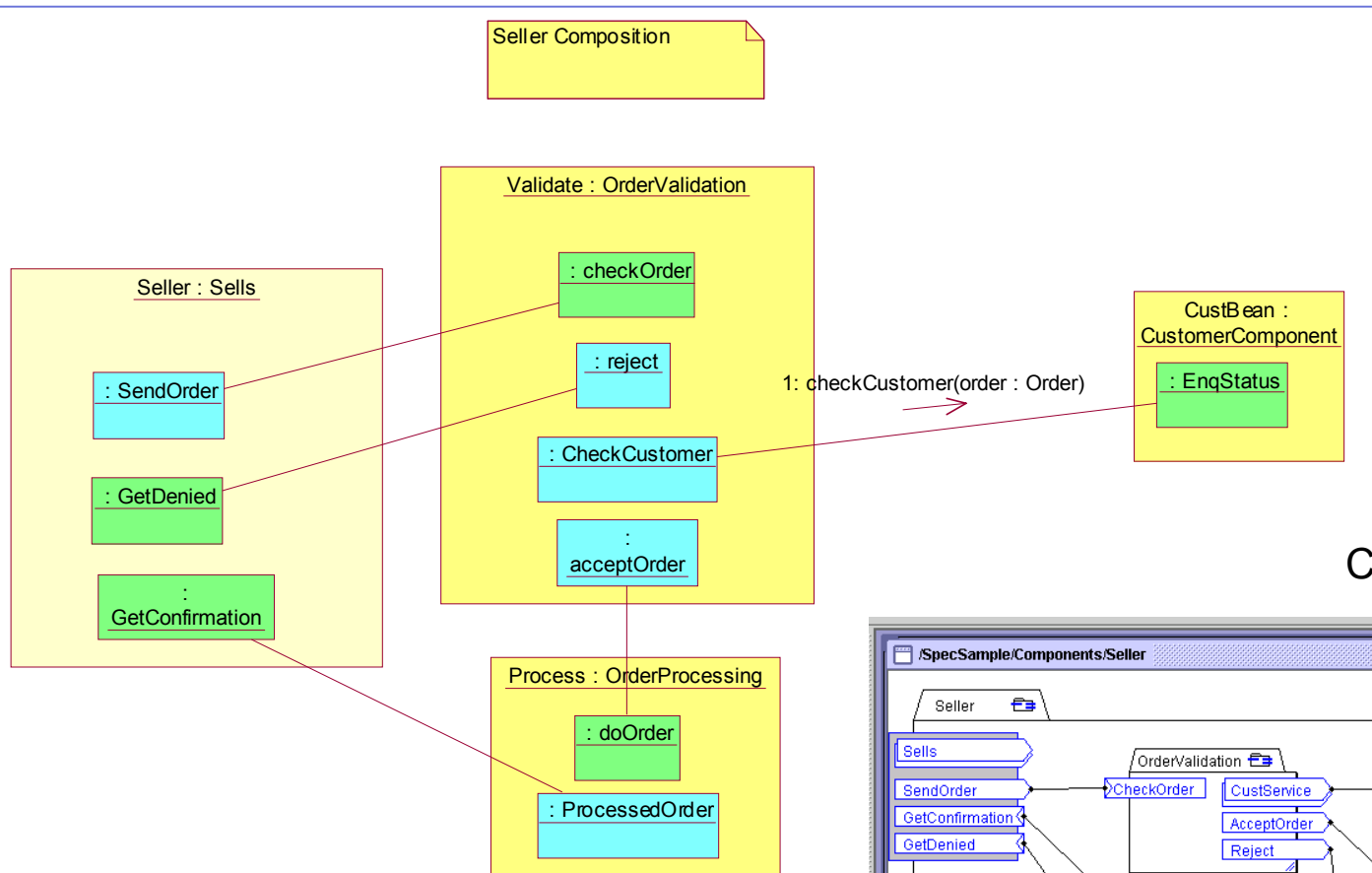
# From MDA to MDLI

## EDOC-CCA Modelling of Components

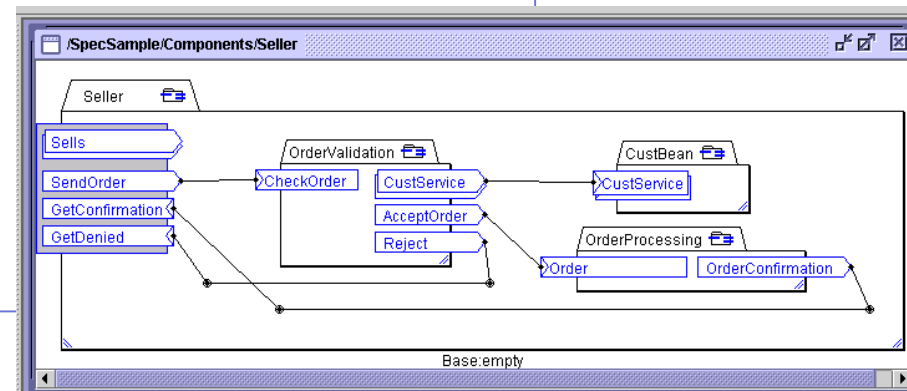


# From MDA to MDLI

## EDOC-CCA Modelling of Collaborations



CCA Notation

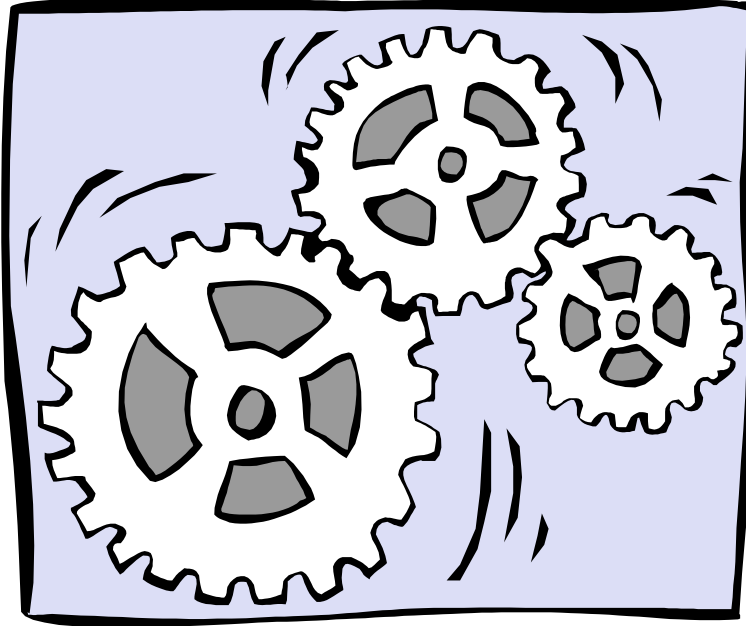


UML Collaboration  
Diagram

# Integration PIM – Contents

- Interfaces of involved (legacy) components
  - Methods or functions, parameters, data elements
  - Data base structures – object based view
- Composition of services to be provided
  - Ports, protocols and messages
  - Components, compositions and (external) interfaces
  - Choreography:  
“... specifies the intended external behavior of a component ...”
- Mapping between encapsulated functions and services
  - Data elements and parameters
  - Internal choreography: function calls, sequences, conditions
  - Managed access to data bases

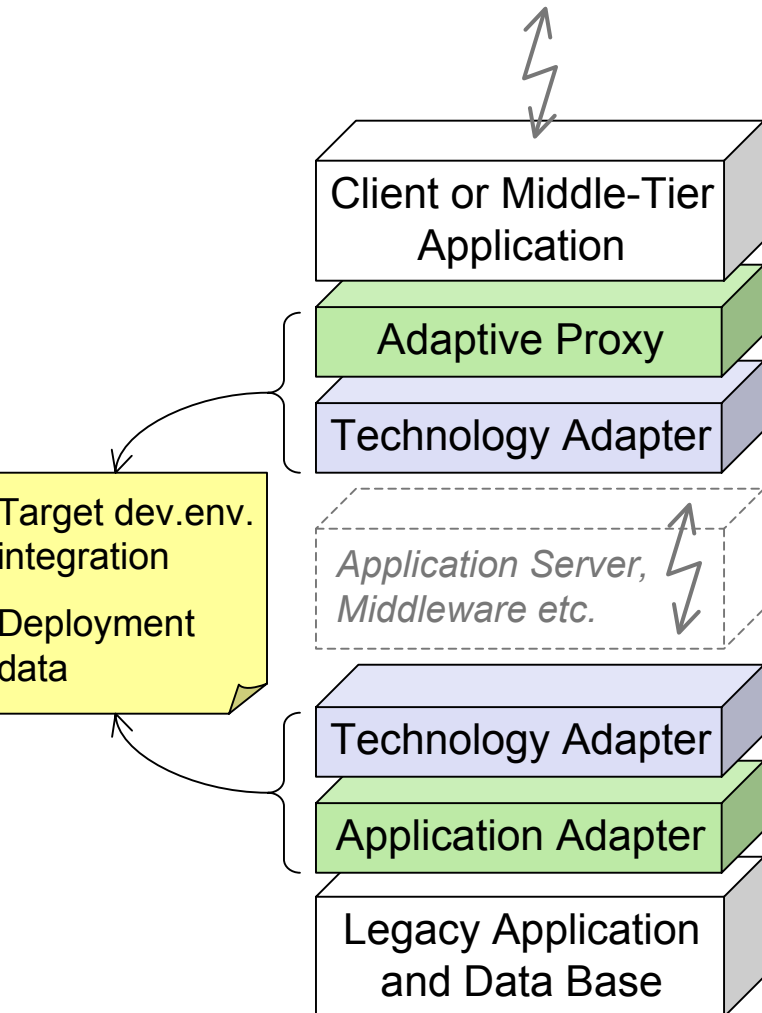
# Model Driven Legacy Integration



**Generated  
PSM and Code**

# Generated PSM and Code

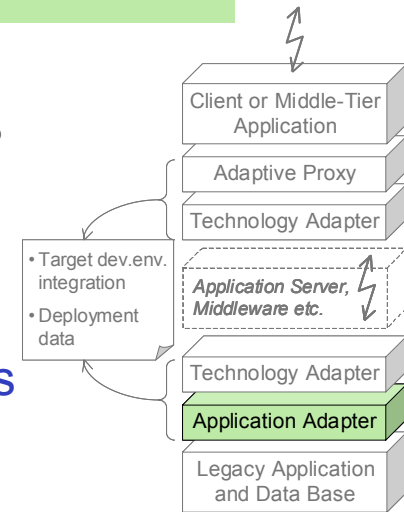
## Multiple Layers



- “Separation of Concerns” by multiple layers
- **Adaptive Proxy**  
Client representation of service
- **Technology Adapter**  
Technical transformations  
Connection to middleware
- **Application Adapter**  
Transformation from legacy architecture to service architecture  
Paradigm mapping
- **Target Dev.Environment integration**
- **Deployment data**

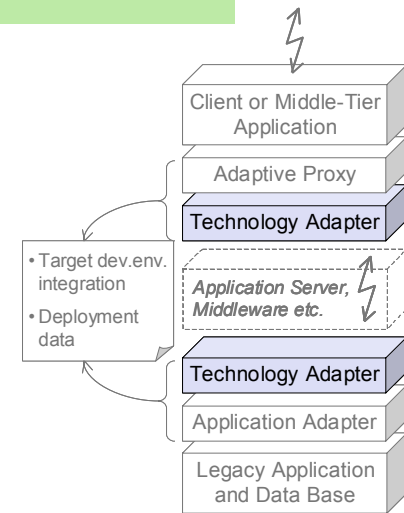
# Generated PSM and Code Application Adapter

- Mapping of service interfaces to legacy functions
  - Move data, activate functions, evaluate results
- Object-relational mapping
  - Encapsulated and managed access to data bases
- Implementation of choreographies
  - Following application logic
  - Enforce operation completeness and data integrity
  - Session logic vs. transactions
- State management
  - Depending on server transaction logic and correlation of choreographies (external vs. internal)
- Support different message versions at runtime



# Generated PSM and Code Technology Adapter

- Transformation of formats
  - Data types and structures
  - Message formats and encoding
- Pack and unpack compound messages
  - Pre-fetch and cache management
- Connection to Middleware
  - Call and respond
  - Handle events and exceptions
- Connection to local operating system
  - Physical transactions
  - Work space implementation for state management

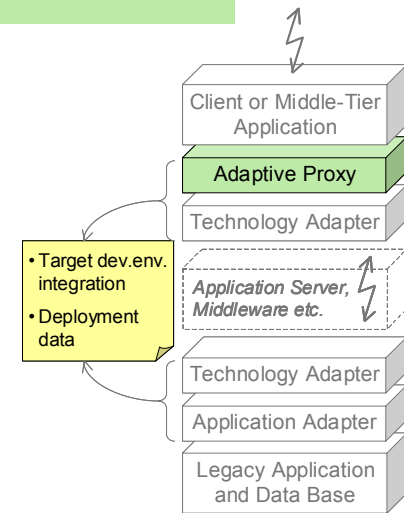




# Generated PSM and Code

## Adaptive Proxy

- Interfaces, Classes and Methods
  - Generated as defined PIM
  - Java, EJB, C#, C++ etc.
- Opaque representation of service
  - Hiding legacy server and middleware
- Acts like a class locally implemented and instantiated
  - No specific data types, objects or operations
  - No “culture clash” for client or middle-tier programmer
- Seamless integration into target development environment
- Deployment data as necessary



# Model Driven Legacy Integration

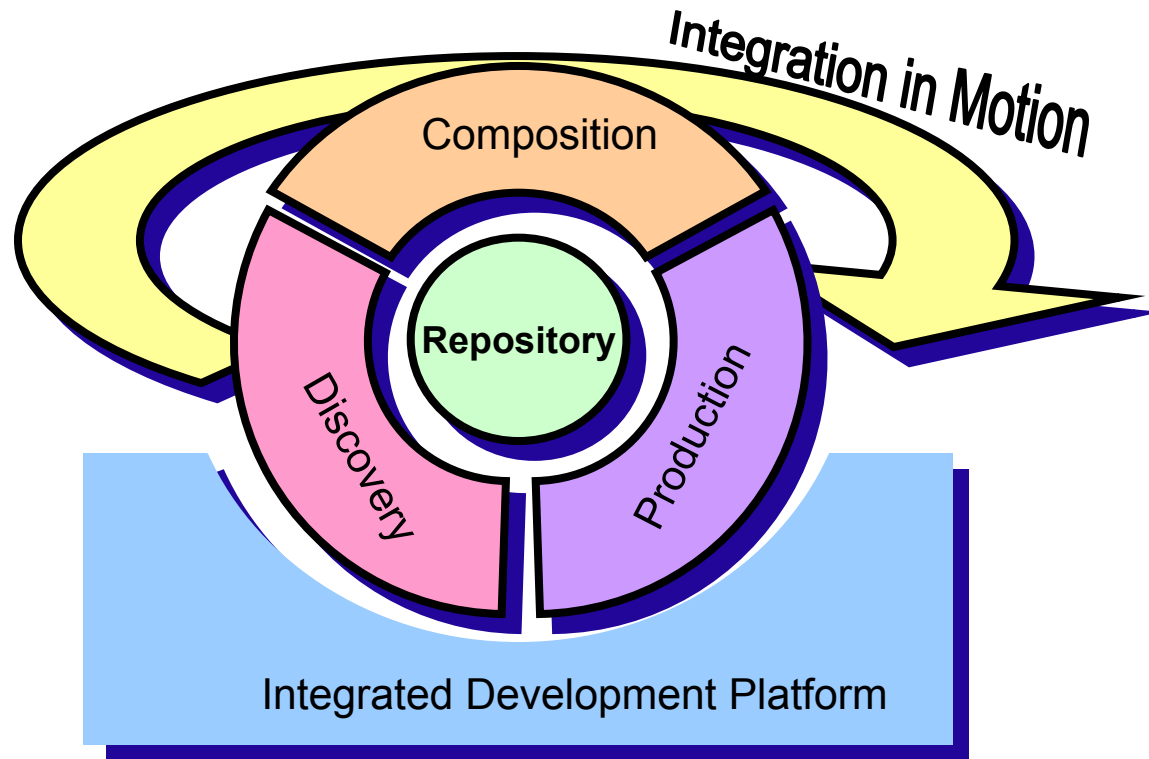


**Tool Support**

**SCORE®**  
**Integration Suite**

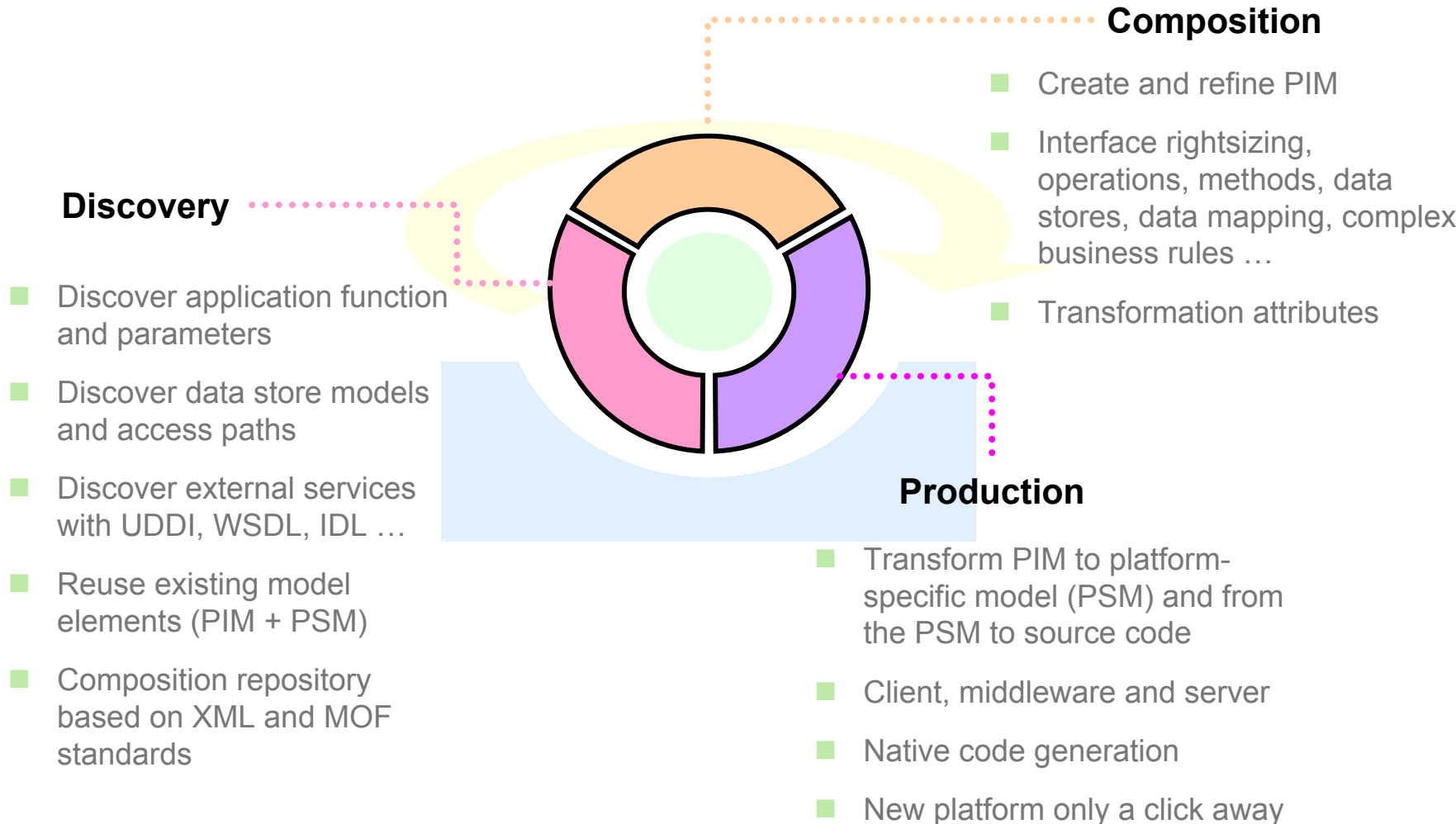
# Tool Support

## SCORE Integration Suite



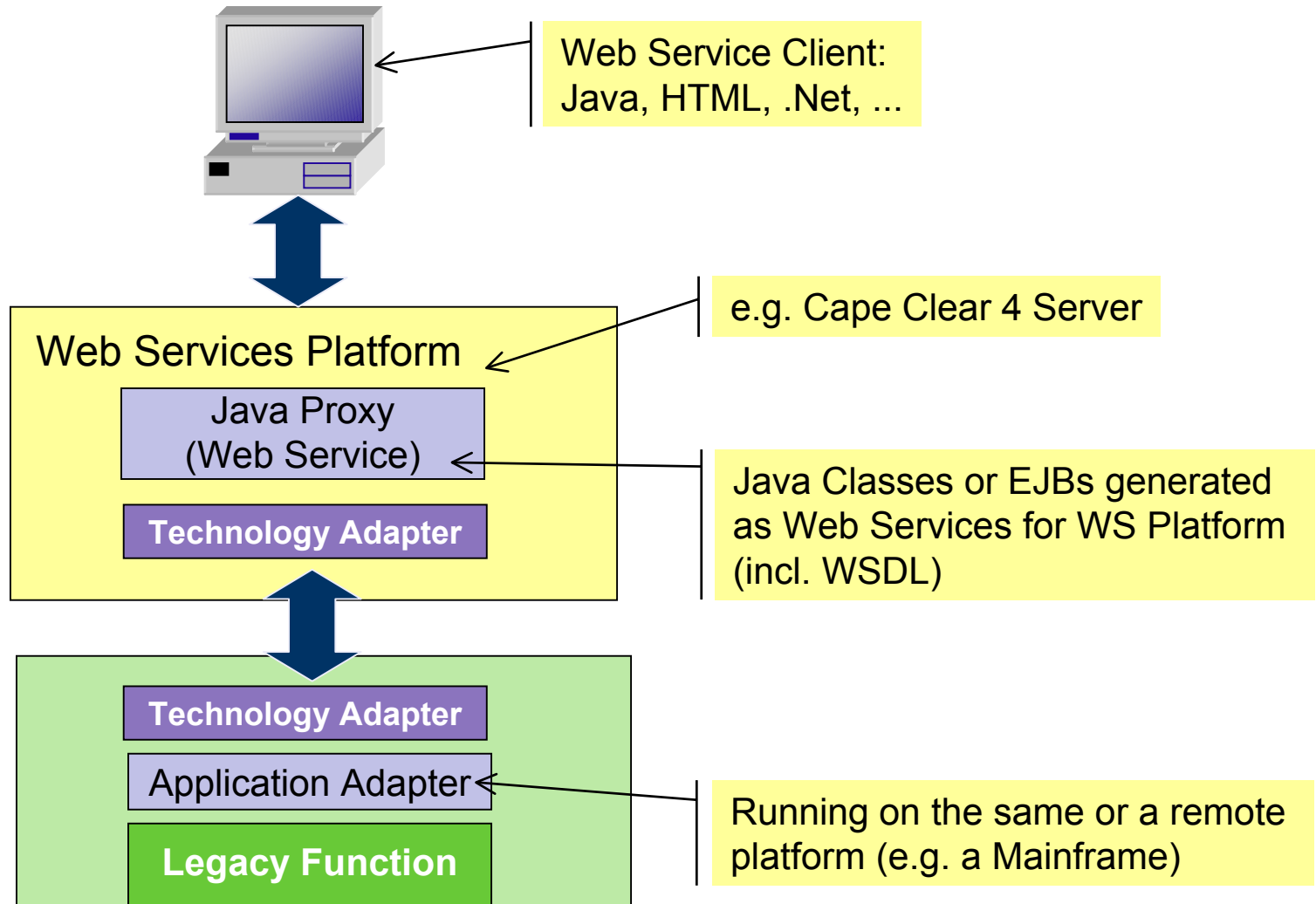
# Tool Support

## SCORE Integration Suite



# Tool Support

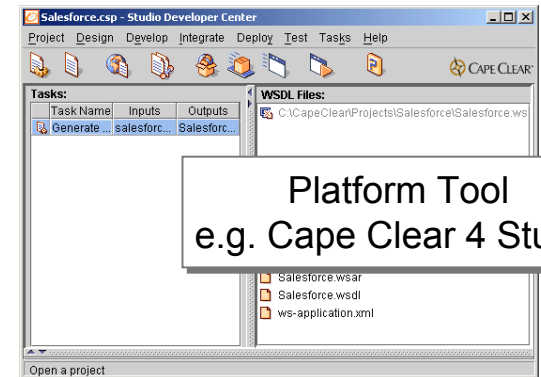
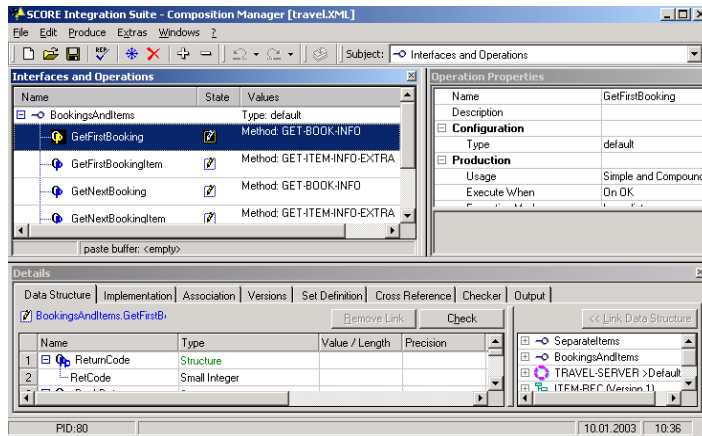
## Production of Web Services



# Tool Support

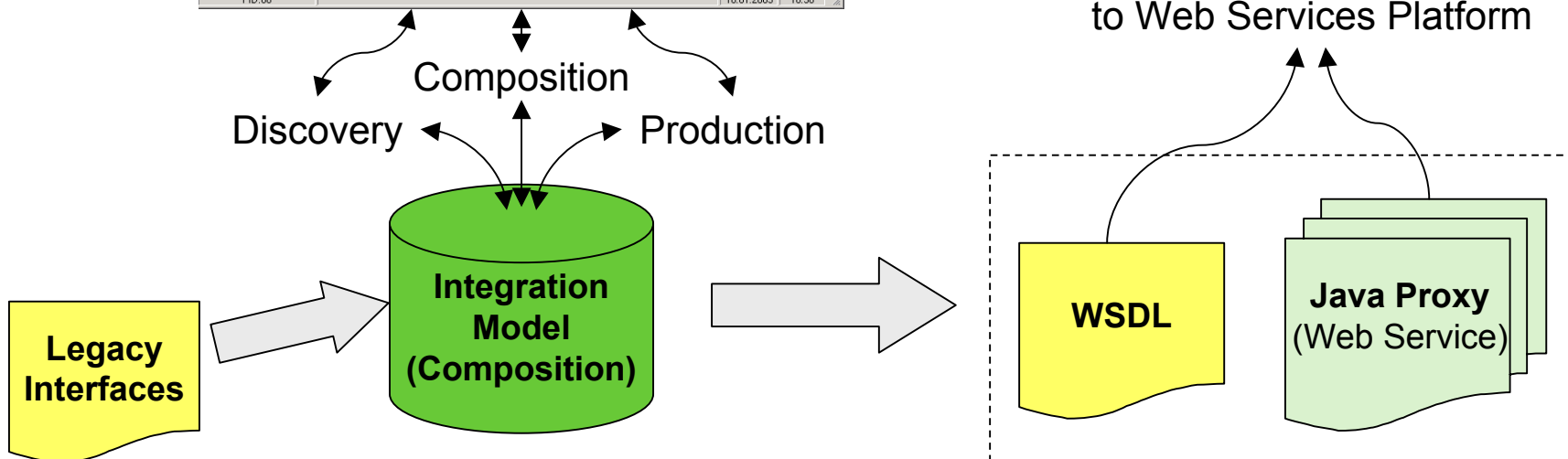
## From Legacy to Web Services

### SCORE Integration Suite Composition Manager

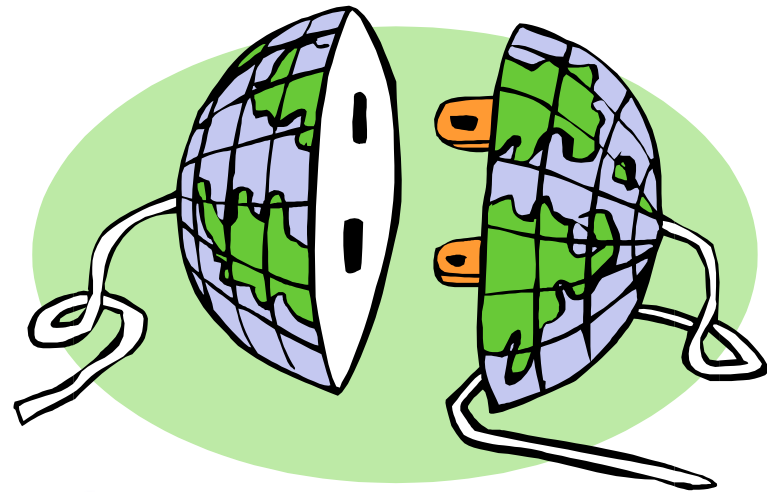


Platform Tool  
e.g. Cape Clear 4 Studio

Package and Deploy  
to Web Services Platform



# Model Driven Legacy Integration

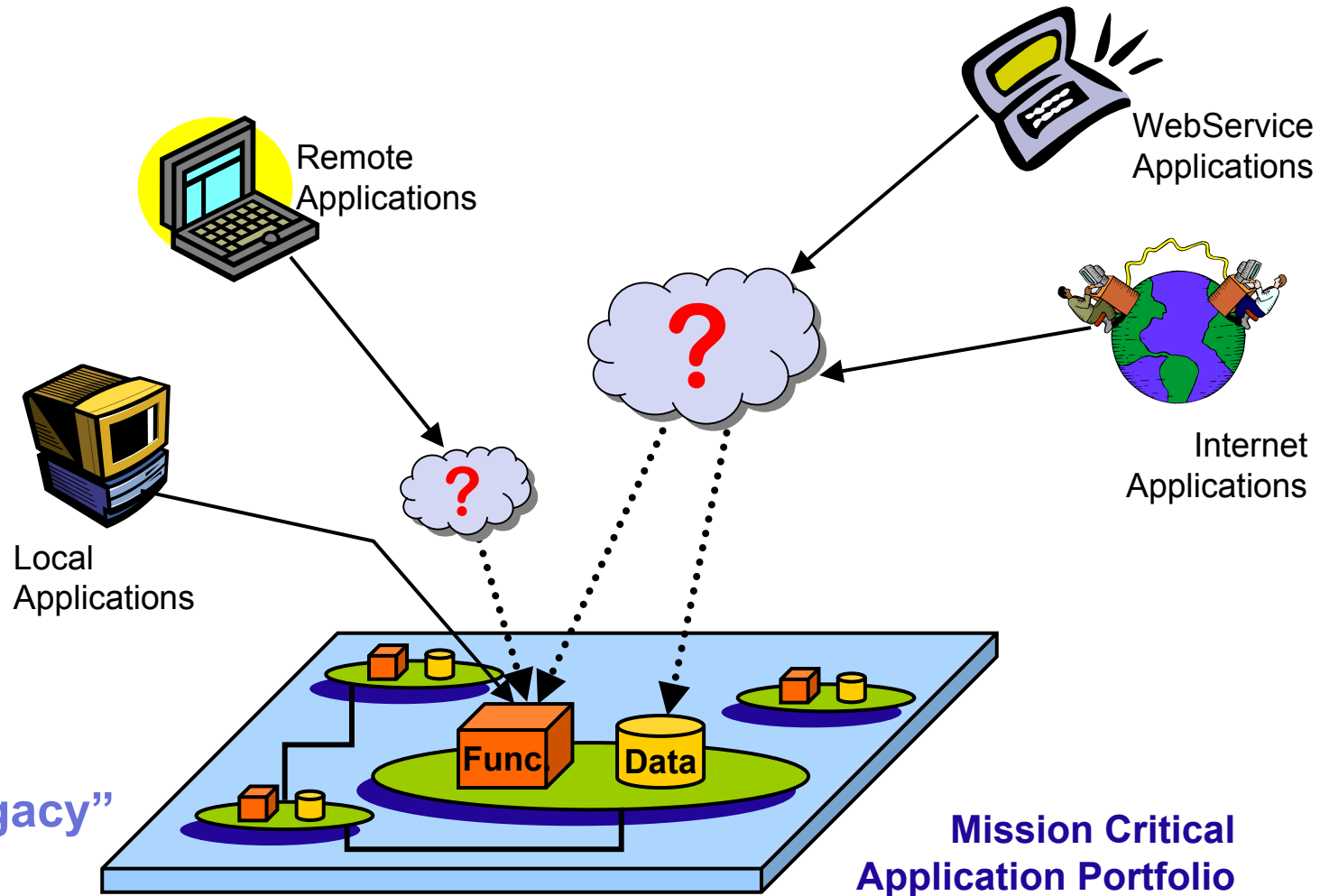


## Summary

## Integration in Motion™

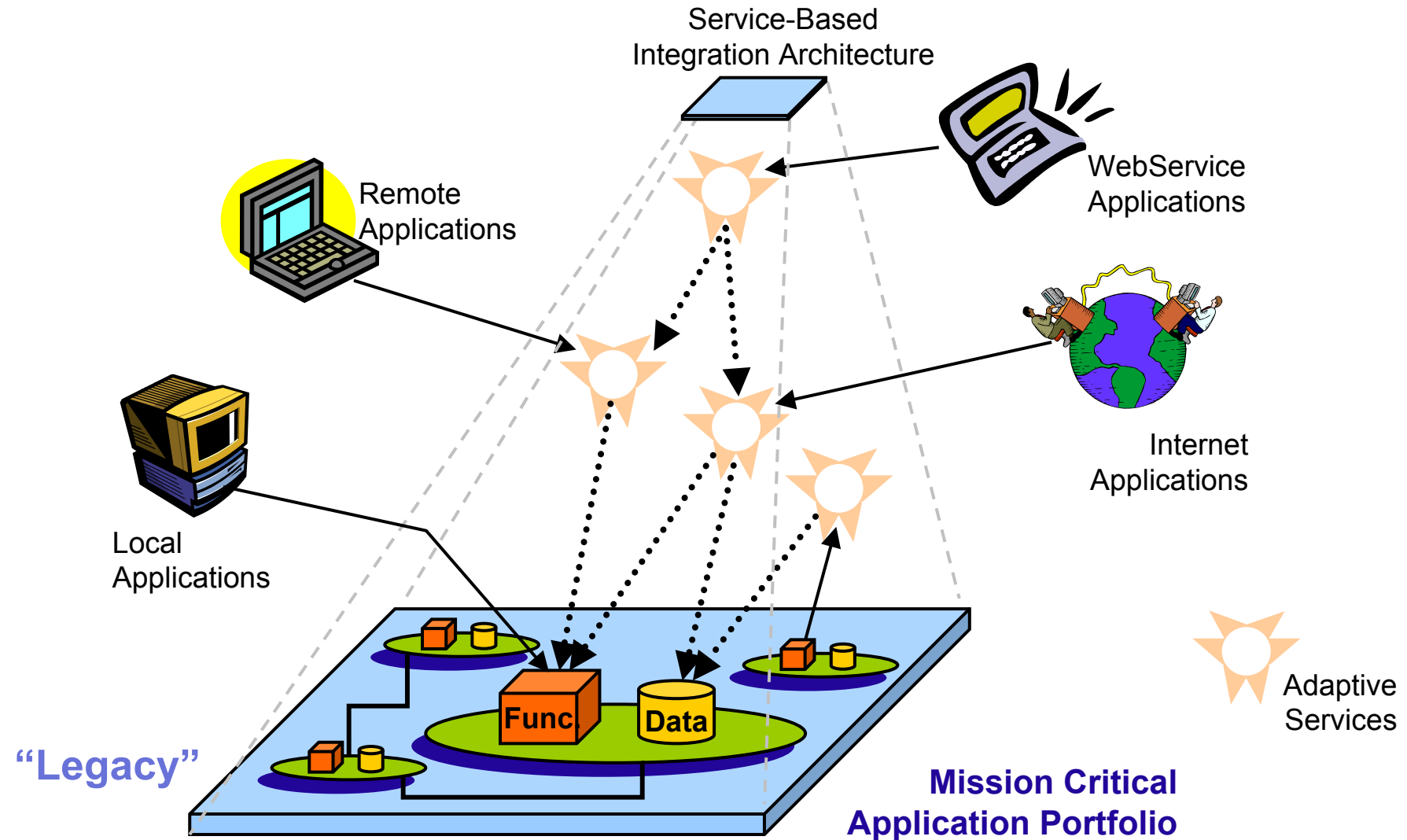
# Integration in Motion

## The Task - Revisited

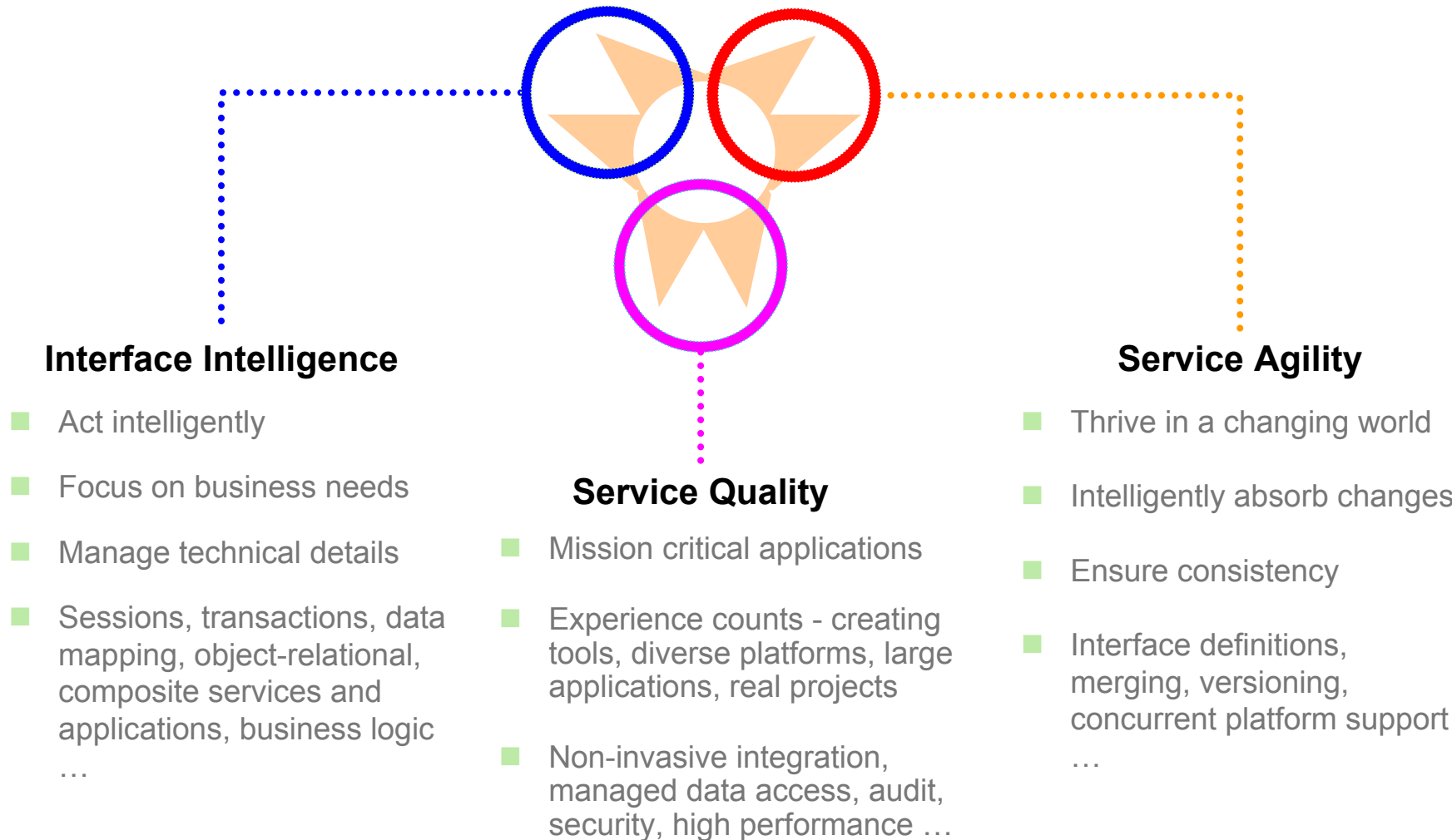




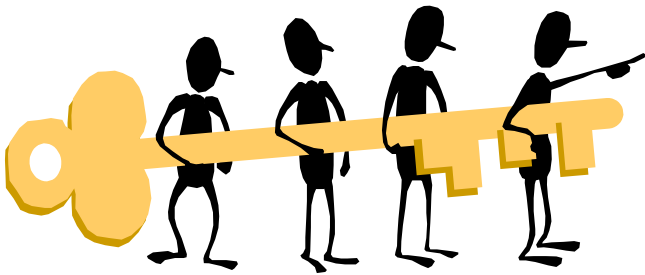
# Adaptive Services for Integration



## Three Aspects of Adaptive Services



# Model Driven Legacy Integration



More about Cape Clear 4

[www.capeclear.com](http://www.capeclear.com)

More about EDOC-CCA and tools

[www.enterprise-component.com](http://www.enterprise-component.com)

All about  
**Model Driven Legacy Integration**  
and  
**SCORE Integration Suite**

[www.d-s-t-g.com](http://www.d-s-t-g.com)