

MDA

-

The Real Value

Oliver Sims

Sims Associates

oliver.sims@simsassociates.co.uk



Background

- Consultant
 - Enterprise systems, component-based development, distributed system software architectures,
 - Transitioning to affective CBD
- Chief Architect
 - Component (app server) middleware, 1990s
 - OMG Architecture Board member
- Author
 - *Business Component Factory* (2000, with Peter Herzum)
 - *Building Business Objects* (1998, with Peter Eeles)
 - *Business Objects* (1994)
- Systems Engineer (IBM 1969-1993)
 - UK large complex systems
 - Specialist, distributed systems, programming technologies, communications, system design, manufacturing industry, etc.



Agenda

- MDA positioning
- MDA essentials
- MDA – the envelope
 - Product Line
 - Architecture
- MDA – the real value
- Next Steps

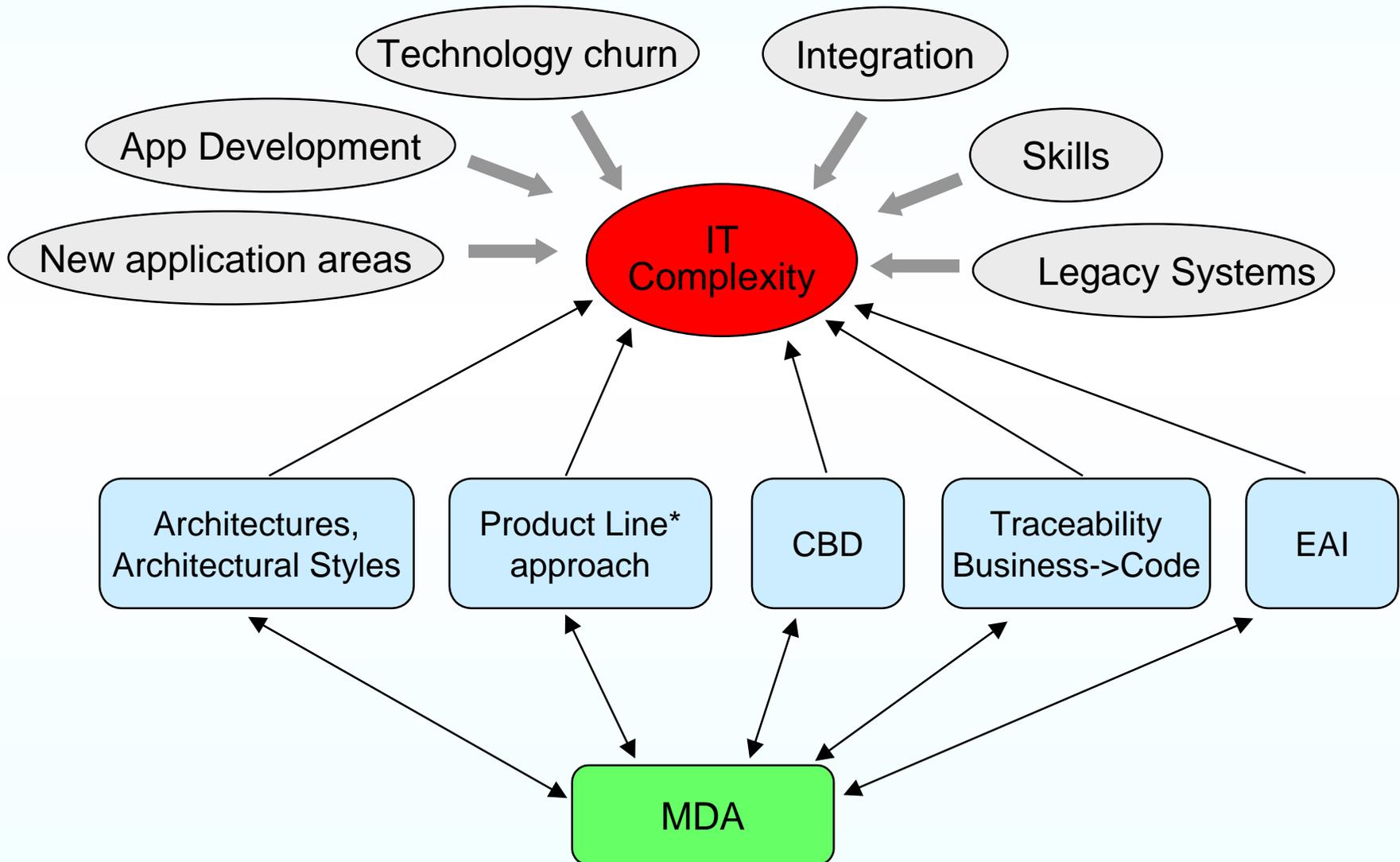


MDA Positioning (1)

- Application development is lengthy and complex
 - Software technology considerations
 - Business requirements become lost
 - Structural design re-invented
 - Architectural concerns often not separated
 - Integration demands often hugely complicated
- Many people have “done MDA” in the past
- MDA is the focus for resolution of the development crisis
 - A major strategy ... from a major standards organization
 - Addresses a key architectural separation of concerns
 - Provides single conceptual framework and vocabulary
 - The focus for synergy between key resolution enablers...



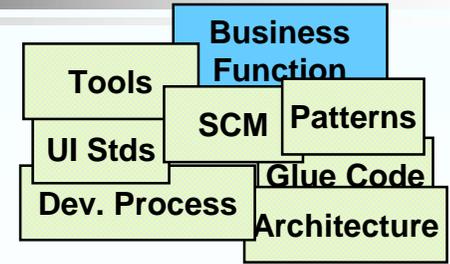
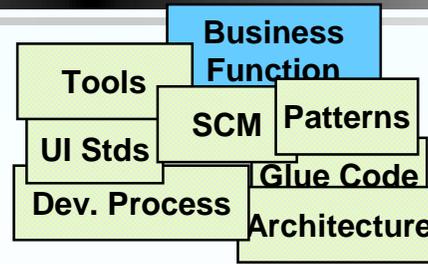
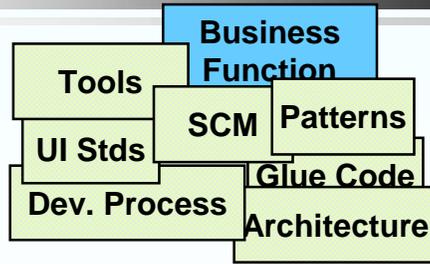
MDA Positioning (2)





Traditional Development

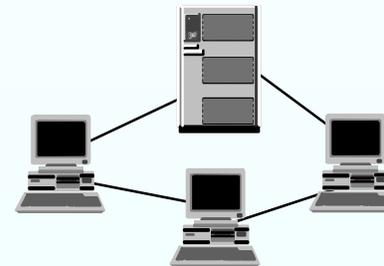
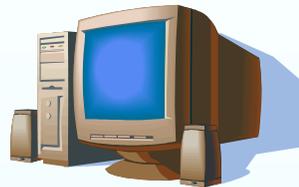
Traditional Development Projects



Traditional IT Infrastructure



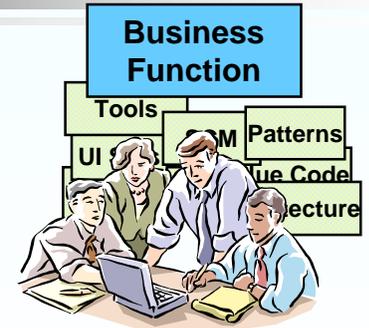
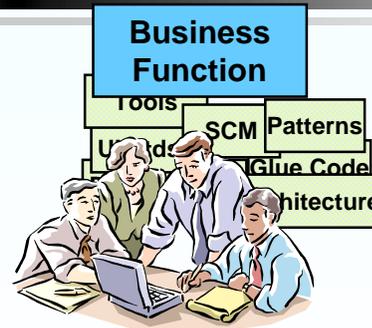
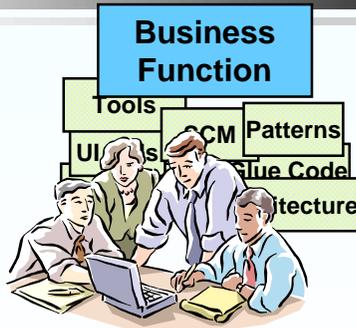
Traditional Office Infrastructure



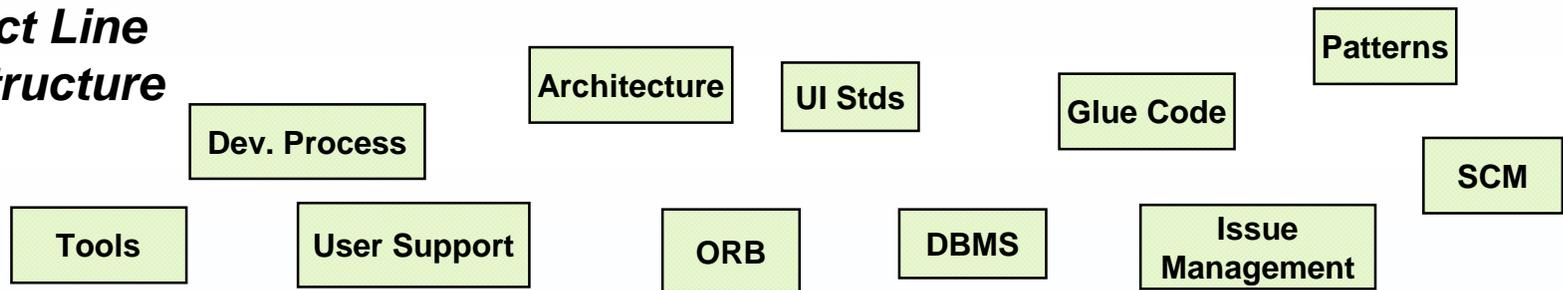


“Product Line” Development

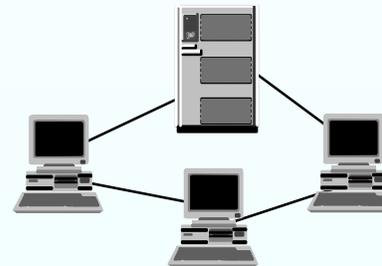
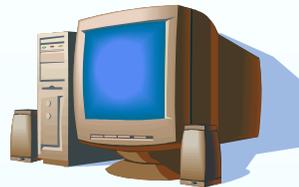
**Business
Function
Projects**



**Product Line
Infrastructure**



Traditional Office Infrastructure





Agenda



- MDA positioning
- MDA essentials
- MDA – the envelope
 - Product Line
 - Architecture
- MDA – the real value
- Next Steps

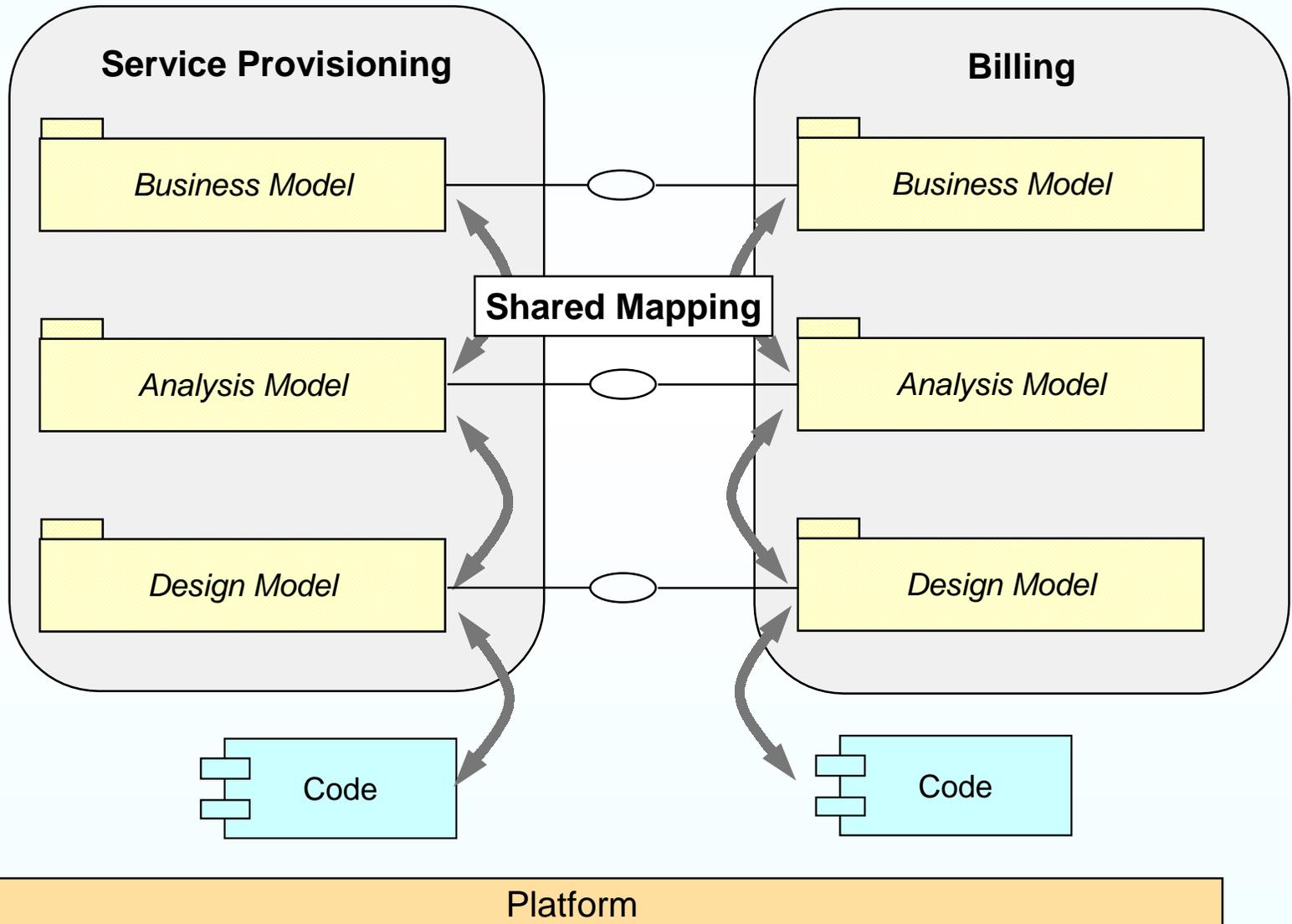


MDA Essentials

Computation-
& Platform-
Independent
Model
(CIM/PIM)

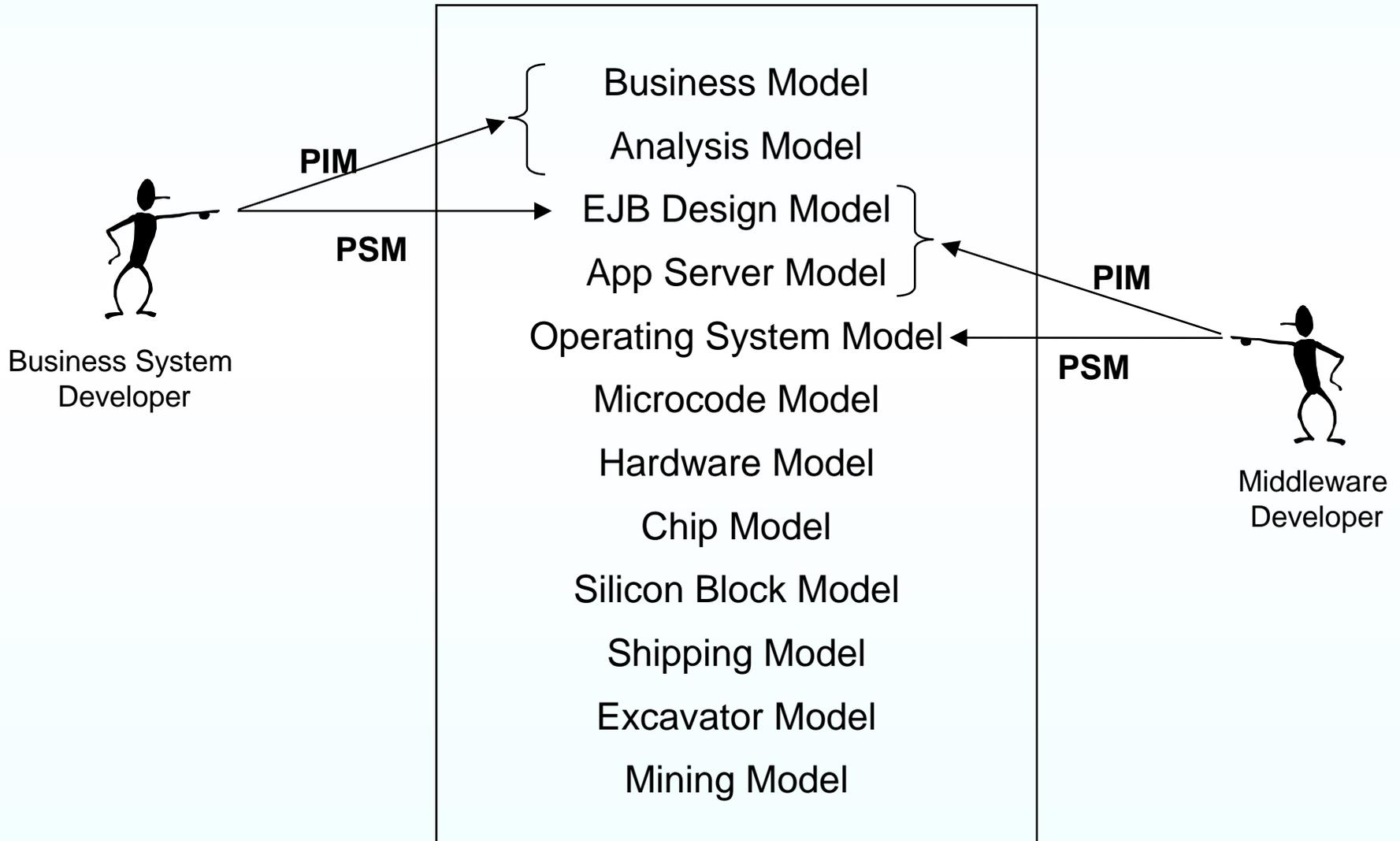
Platform-
Independent
Model
(PIM)
(Computation-
Dependent)

Platform-
Specific
Model
(PSM)



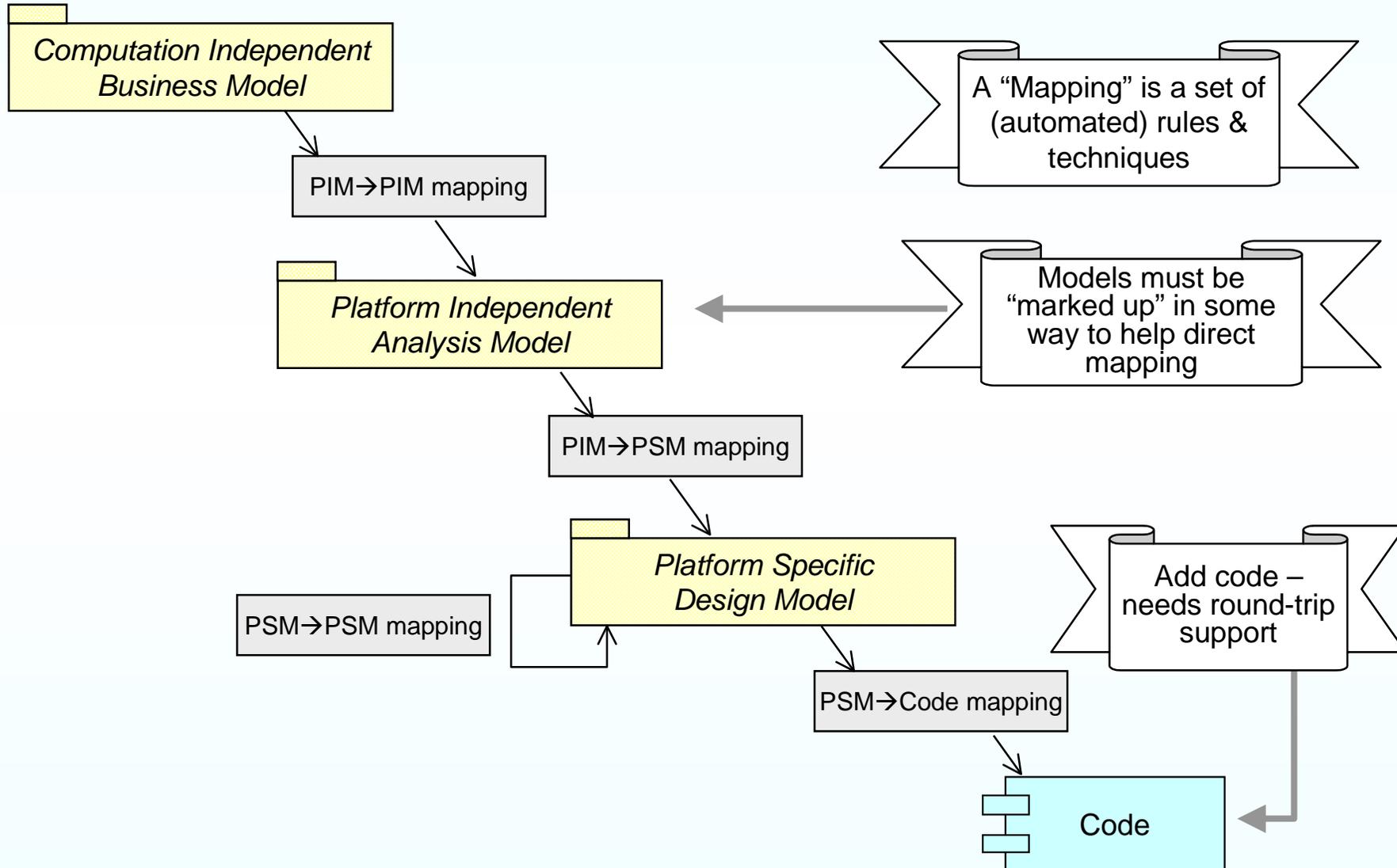


The Importance of Context



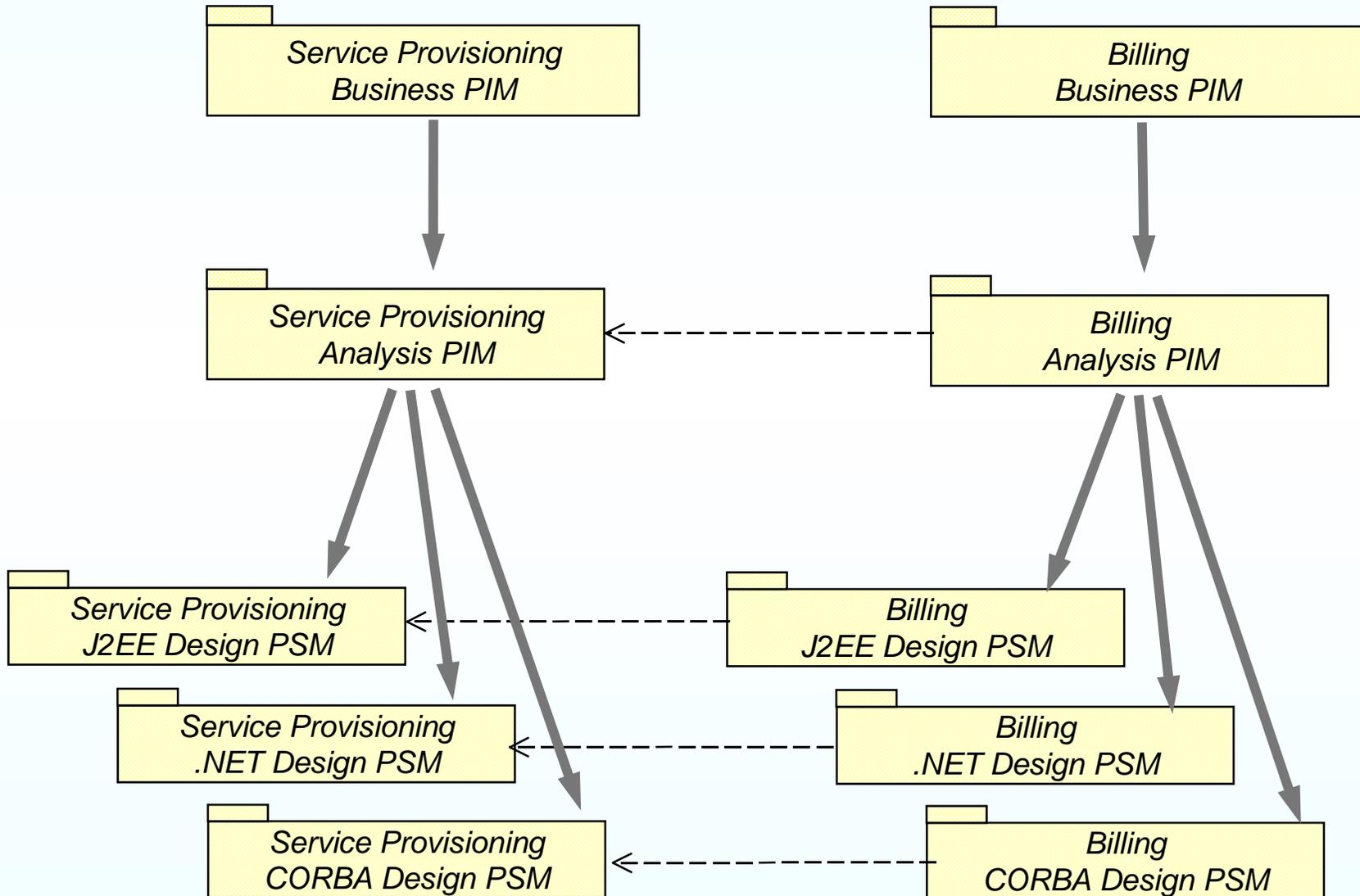


Mapping





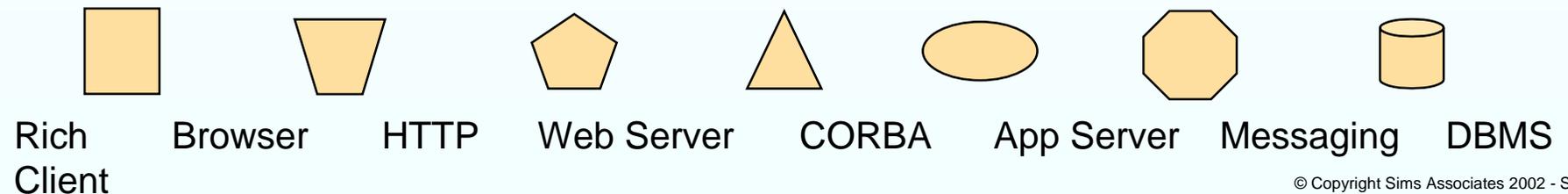
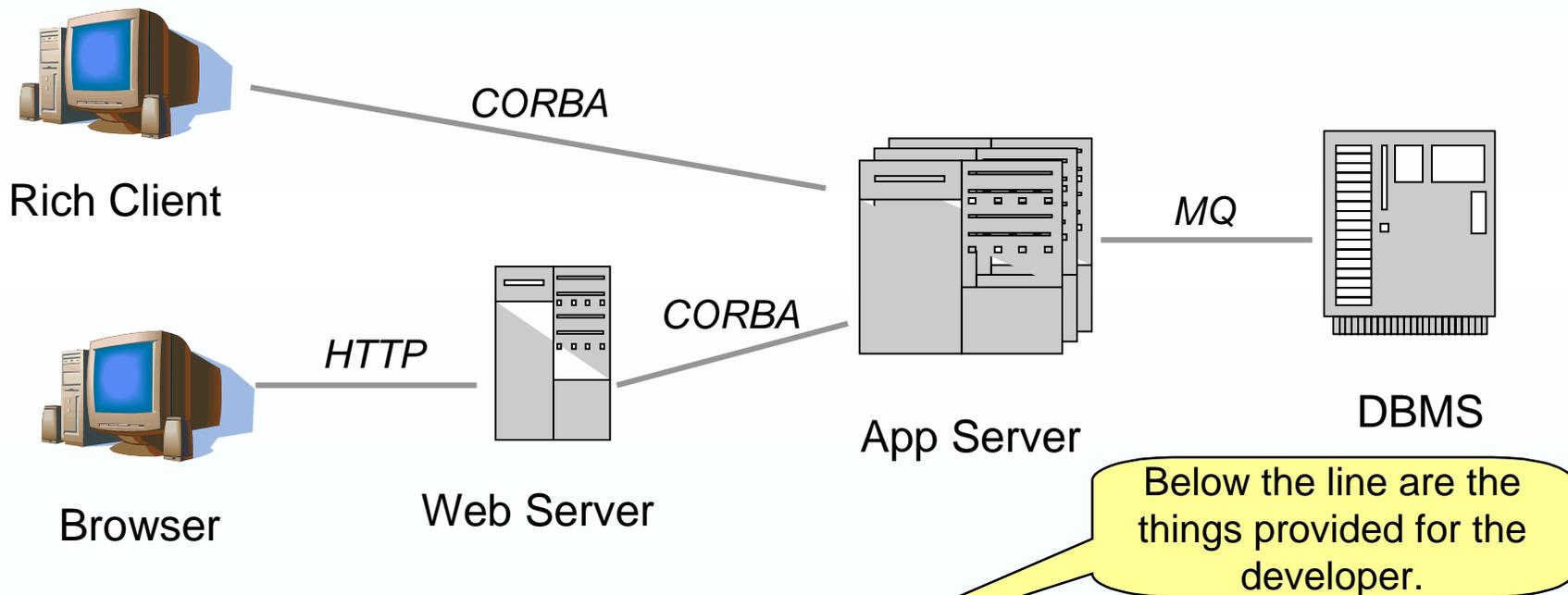
MDA Example (1)





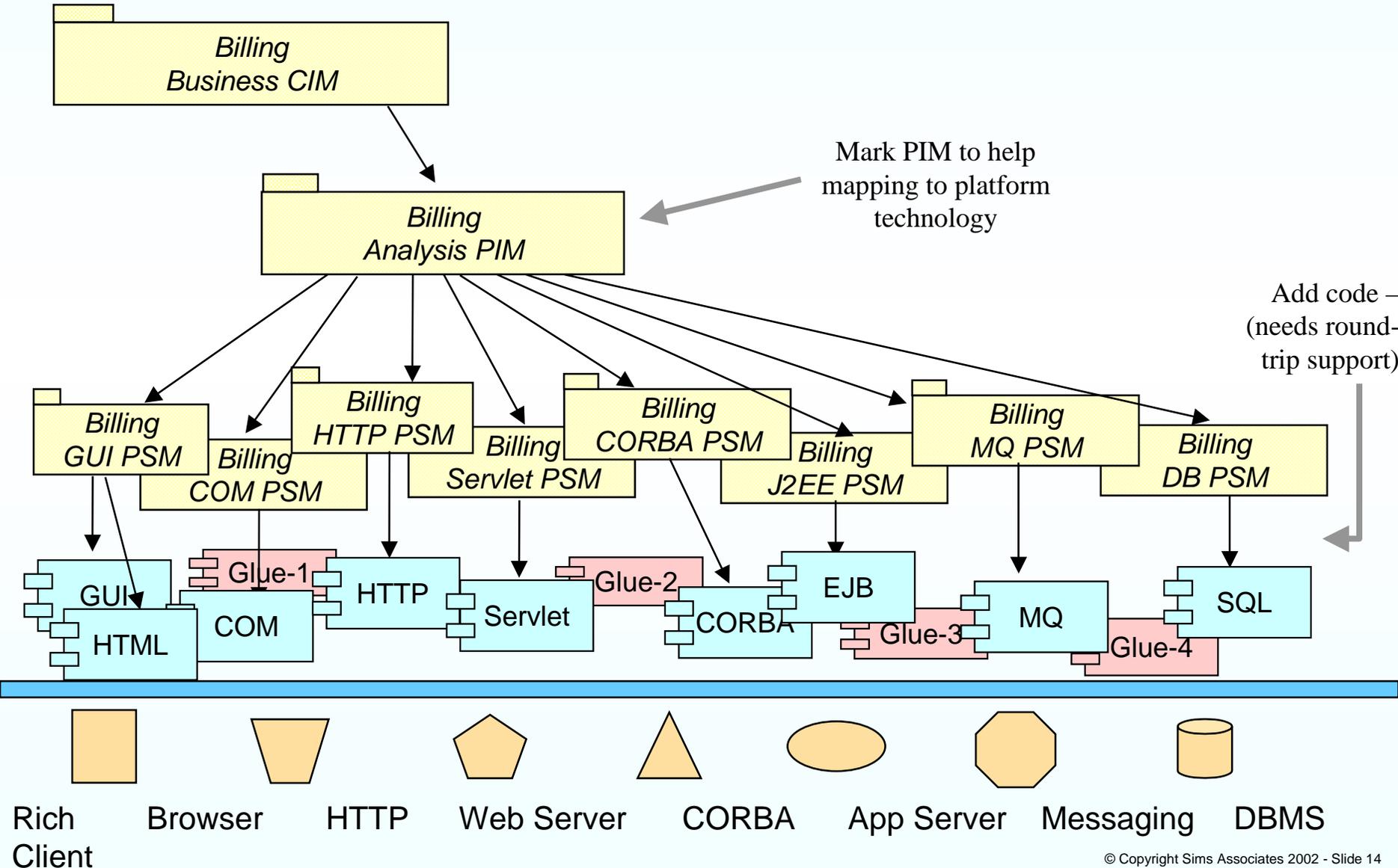
MDA Example (2)

- Target run-time environment:





MDA Example (3)





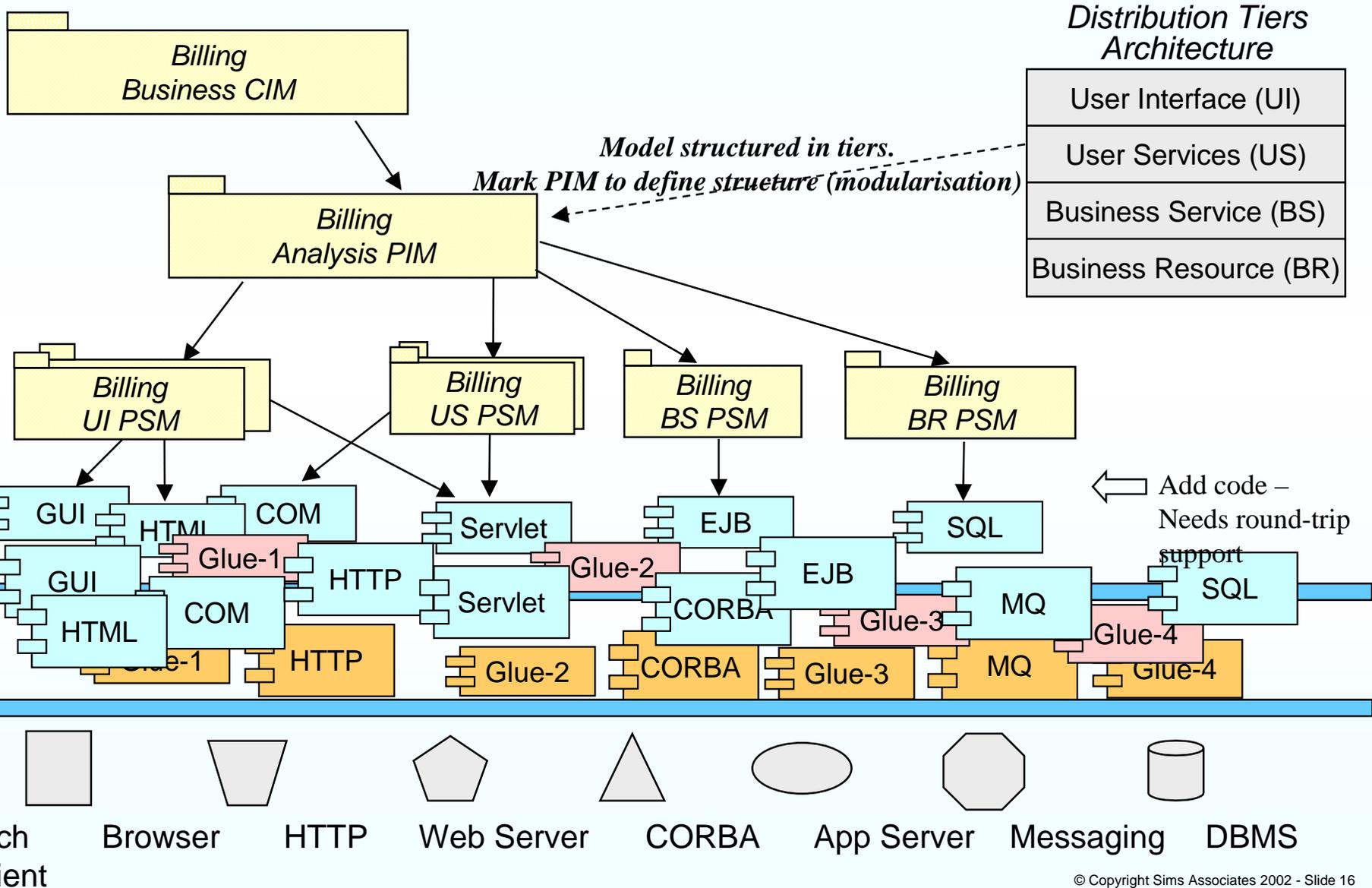
Agenda

- MDA positioning
- MDA essentials
- MDA – the envelope
 - Product Line
 - Architecture
- MDA – the real value
- Next Steps





MDA + Product Line





Process Improvement with Product Line

“When an organization with a single project/product focus moves from CMM level 3 to level 4, the productivity gain is minimal because most of the improvements that can affect a single product or project will already have happened at level 3. ...

“However, when the process improvement from level 3 to level 4 includes a shift to product line focus, the productivity increase is very significant. Vu’s data indicate as much as a 70% productivity improvement, as well as highly satisfied employees.”

Quote by John D. Vu, Technical Fellow and Chief Engineer at Boeing, from *Software Product Lines*, Paul Clemens and Linda Northrop, Addison-Wesley 2002.



Computational Completeness

- A model that can be executed - via code generation or interpretation - is said to be “computationally complete”
- Requires:
 - Action Language for algorithmic logic
 - Computational structure
- The aim is to build computationally complete PIMs
 - All development at the model level
 - Execute the model to test
 - Generate code where necessary
- Component architecture provides an excellent structure



What is a “Component”?

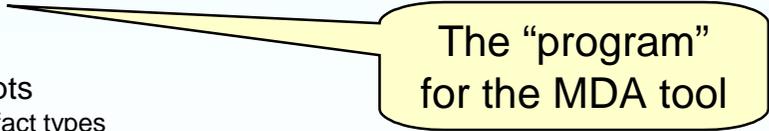
- A managed executable – runs in a middleware “container”
- A pluggable artifact
 - Through the development lifecycle
 - A managed module/package
 - Built with non-component artifacts
- Has programmatic “interface”
 - Local/Remote transparency
- Designed to represent a single “business” concept
 - Throughout the development life cycle
- Composable with other components
 - “Autonomous” not isolated – composed by reference
 - Defined granularities
 - Defined ownership
- Conforms with OO concepts
 - Encapsulation, instantiation, state + behavior
 - Inheritance
 - (where technically possible)
- Can address distributed systems end-to-end



Product Line + CBD Architecture

- Architecture (for a given architectural style):
 - Conceptual models that define:
 - Component-oriented application structure concepts
 - Including component granularities and re-usable artifact types
 - Superior modularisation strategy
 - scalability patterns
 - Separation of distributed system (and other) concerns
 - The product line for this architectural style, including design of technology/business separation
 - Development process
 - Skeleton structure for models
 - Mappings (and traceability) between models
 - Plus UML profile for models
 - Design for glue code
 - etc.

(there are a number of published examples)
- Result:
 - Design for component-oriented application structure
 - Design for a product line
- Evolve the new product line
 - A new level of productivity
- **Re-use the product line** for all applications of the same architectural style
- Repeat for each architectural style
 - Re-using common architectural elements

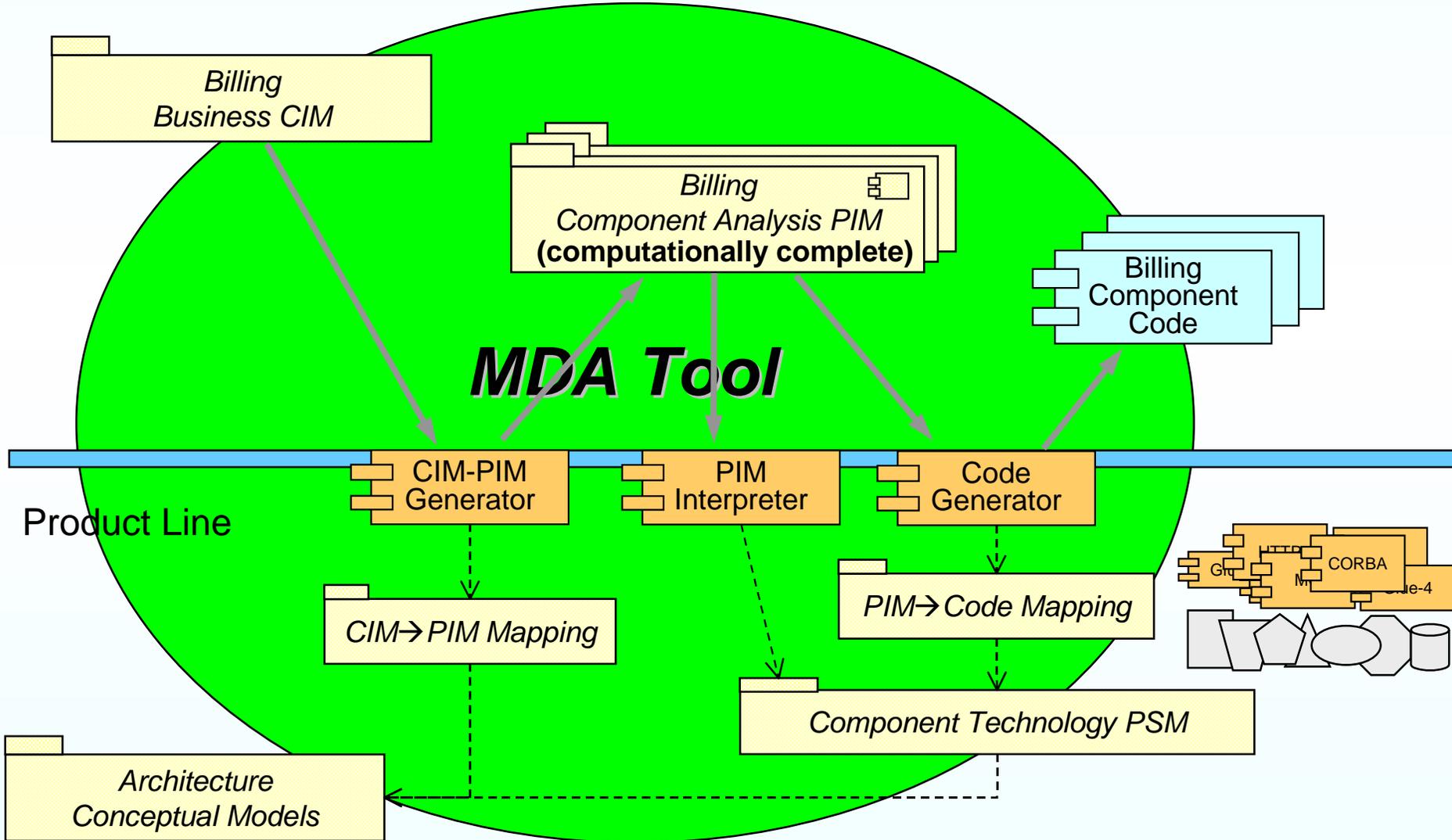


The “program”
for the MDA tool



MDA + Product Line + Architecture

Application Development



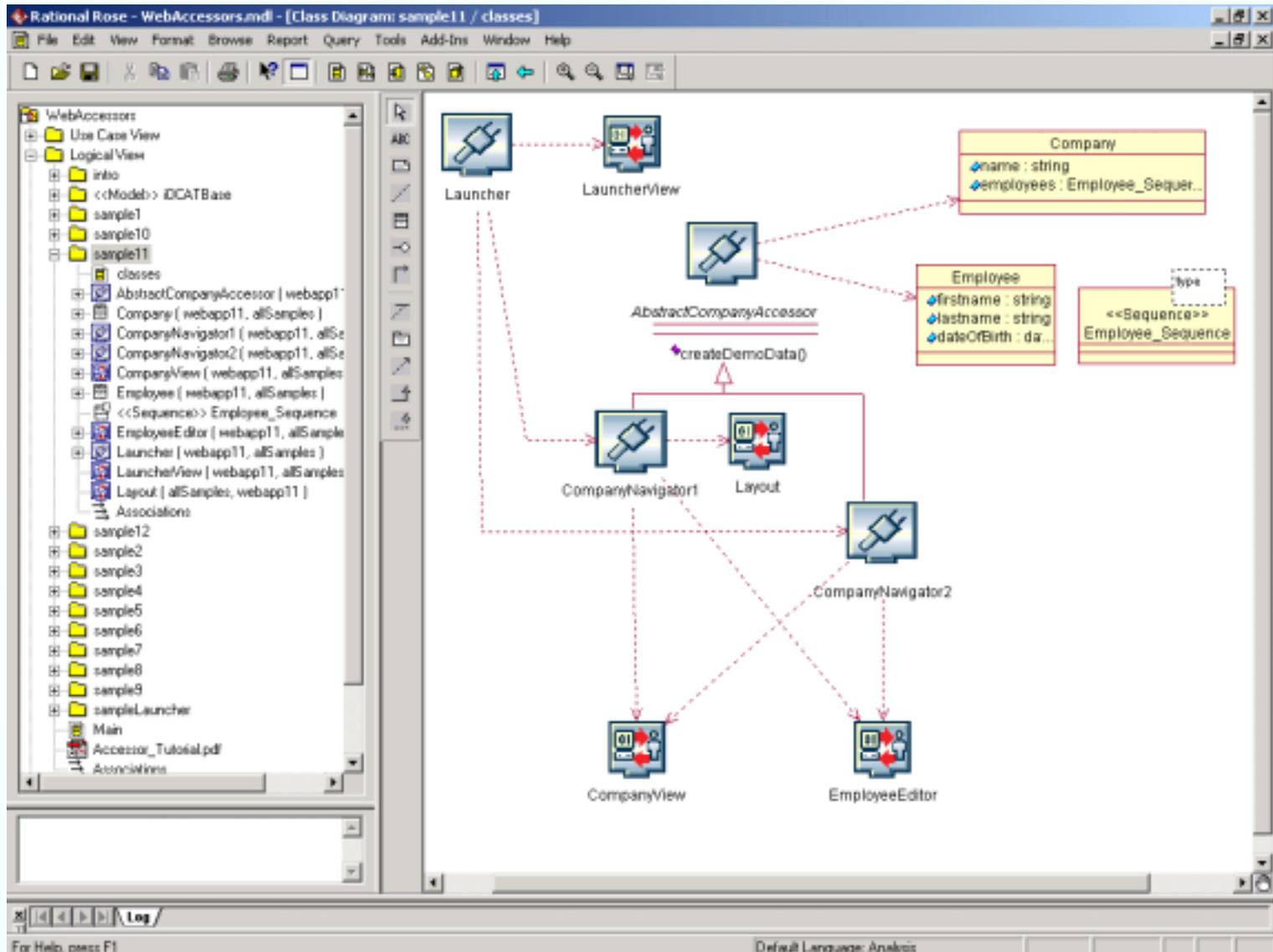


The Perfect MDA Tool (2002)

- Current “standard” capabilities
 - UML modeling
 - Models (or parts of models) can be web-published
 - Code generators for major (and some other) platforms
 - XMI model interchange
- Action Language and/or OCL
- Integrated IDE
- Integration of modules at the PIM level
 - Re-use previously-built PIM modules – e.g. in CBD
- Integration with EIA tools
- CIM/PIM/PSM differentiation available
- Reverse engineering
 - Round-trip engineering
- Executable models
 - in run-time as well as in development
- Architecture support:
 - Metamodel approach, so new UML and other metamodels can be developed and “plugged in”
 - Using MDA of course!
 - Pluggable PIM architectures
 - E.g. CBD architecture
 - Support for GUI and Data specification consistent with architecture
 - Pluggable generators and mappings if necessary
 - Pluggable “glue” code consistent with architectures supported
- Tool designed using MDA for fast evolution
- Scalable to large development teams
 - Repository support, versioning, sharing, revision-marking, etc.
- Excellent user interaction design
 - E.g. following Alan Cooper’s guidelines (see “The Inmates are Running the Asylum”)
- ???



CBD Architecture support





Configuration and Code Generation

ArcStyler Configuration

- Global
- C-GEN
 - Generate
 - Projections**
 - Build
 - Database
- C-GEN-IDE
 - Coloring
 - Editor Options
- Tools

C-GEN - Projections

Chosen Technology Projections:

- S:\carat.carat\gen\cartridges\carat.tpr
- S:\carat.java2\gen\cartridges\Java2.tpr**

Add
Remove
Reload

OC | Webservices | Test | Server Config | CIX | Default
Generator | Client Config | FSM Support | RMI

Use precompiled templates Yes No

Cartridge source code directory :

Cartridge components working directory :

Generate depending physical components : Yes No

Project Configuration :

Domain Configuration :

OK Apply Reset Cancel Help



Reverse Engineering

The screenshot shows the Rational Rose IDE with several windows and annotations:

- Class Specification for Trip**: A window showing the properties of the Trip class, including Bean Home Name (TripHome), Bean Type (Entity), Persistence Management (Container), and isReentrant (True).
- ArcStyler Configuration**: A window showing the configuration for the Trip class, including the Source (C:\Programme\ArcStyler\samples\extended\TripPlanning\src) and Category (Elements to Explore).
- Model Properties**: A table showing the properties of the Trip class.
- Model Browser**: A tree view showing the model structure, including the Trip class and its associations.
- Log**: A window showing the Hierarchical Action Log, listing various actions performed during the reverse engineering process.

Annotations in yellow boxes point to specific features:

- Detailed Exploration**: Points to the Class Specification for Trip window.
- Referenced Java Framework**: Points to the Associations section in the Model Browser.
- "Compact" EJB Components**: Points to the Trip class in the Model Browser.
- Configuration**: Points to the ArcStyler Configuration window.
- Hierarchical Action Log**: Points to the Log window.



IDE Integration

The screenshot displays the Objecteering IDE interface. On the left, a project browser shows a package structure for 'LiftSimulator'. The central pane shows the 'LiftSimulator' class with various properties and methods. The right pane displays a class diagram showing a dependency between 'LiftSimulator' and 'Administrator' with a multiplicity of '0..1' and the role 'administrator'. The bottom pane shows the source code for 'LiftSimulator.java' and 'Administrator.class'. The console window at the bottom right shows the execution output.

```
objecteering-startJavadoc ..... H-DIY/0V4EE AV4
objecteering-endJavadoc ..... E-DIY/0V4EE AV4
public class JLiftSimulator
{
  objecteering-startJavadoc ..... H-FIY/0V4EE PV4
  objecteering-endJavadoc ..... E-FIY/0V4EE PV4
  private Lift PkgAdministrator Administrator administrator;
  private Lift PkgAdministrator Administrator getAdministrator () {
    return this.administrator;
  }
  private int cardAdministrator () {
    if ( this.administrator == null ) return 0;
    else return 1;
  }
}
objecteering-startJavadoc ..... H-FIY/0V4EE TV4
objecteering-endJavadoc ..... E-FIY/0V4EE TV4
```



Action Language

IUML - C:\HRIUML_28Jun\HRIUML\HRI.uml [Base Variant] - [3: Class Diagram for HRI Access Management::HRI Version 2]

HRI Access Management v2

IUML - C:\HRIUML_28Jun\HRIUML\HRI.uml [Base Variant] - [4: Class Operation Details for HRI Version 2:linkRoleToTypeOfAccessor]

Database : NHSIA
 Domain : HRI Access Management, HAM
 Version : 2: HRI Version 2
 Class : 22 Role (RL)
 Operation : 2, linkRoleToTypeOfAccessor

External Visibility : FALSE
Scope : Class
Deferred : No

Description
 <Description>

Contract Type : Closed Blocking
Contract Description
 <Contract Description>

Input Parameters

Parameter	Type
theRoleName	Linked To : Role.roleName
theAccessorTypeName	Linked To : Type Of Accessor.accessorTypeName

Return Parameters

Parameter	Type
roleFound	Boolean

Method
 myRole = find-one Role where roleName = theRoleName
 if myRole = UNDEFINED then
 roleFound = FALSE
 else
 roleFound = TRUE
 myTypeOfAccessor = find-one Type_Of_Accessor where \
 accessorTypeName = theAccessorTypeName
 if myTypeOfAccessor = UNDEFINED then
 [myTypeOfAccessor] = TOA1:createTypeOfAccessor[theAccessorTypeName]
 endif
 link myRole R3 myTypeOfAccessor
 endif

Exception Handling Code
 <Exception Code>

Attached Tags

Name	Value

Code Block
 <None>

IUML - C:\HRIUML_28Jun\HRIUML\HRI.uml [Base Variant] - [3: State Machine for HRI Version 2:LoggedIn User]

```

stateDiagram-v2
    [*] --> SelectingApplication
    state "Selecting Application" as SelectingApplication {
        entry /
        (myApplications) = this -> R4 -> R3 -> R5.Application
        for myApplication in {myApplications} do
            [] = US2:displayApplicationName[myApplication.applicationName]
        endfor
    }
    SelectingApplication --> ViewingPatientData : viewPatientRequest(userCode)
    state "Viewing Patient Data" as ViewingPatientData {
        entry /
        [newPatientBeingIdentified] = PB11:createPatientBeingIdentified[this]
    }
    ViewingPatientData --> DeletingUser : userLoggedOff(userCode)
    ViewingPatientData --> PatientDataViewed : transactionComplete(user)
    state "Deleting User" as DeletingUser {
        entry /
        myPatient = this -> R9
        if myPatient != UNDEFINED then
            [] = POI2:deletePatientOfInterest[] on myPatient
        endif
        delete this
    }
    state "Patient Data Viewed" as PatientDataViewed {
        entry /
        myPatient = this -> R9
        if myPatient != UNDEFINED then
            [] = POI2:deletePatientOfInterest[]
        endif
        delete this
        generate U4:displayApplications()
    }
  
```



Model execution

Domain: PDAR Class: Patient_With_Consents State: 7 Line: 20

File View Execution Data Breakpoints Stimulus Tools Help

```
#5 (allCandidateAccessRules) = union-of {patientSpecificAccessRules
#6
#7 (enablingAccessRulesForTheseAccessors) = this -> R16 -> R1.Access
#8 (disablingAccessRulesForTheseAccessors) = this -> R16 -> R7.Acce
#9 (allAccessRulesForTheseAccessors) = union-of {enablingAccessRule
#10
#11 (allAccessRulesToApply) = intersection-of {allCandidateAccessRul
#12
#13 if countof {allAccessRulesToApply} = 0 then
#14 # No access rules found - set to "undefined" access level
#15 currentLowestAccessLevel = 1
#16 else
#17 # Start access level at "enable"
#18 currentLowestAccessLevel = 2
#19
#20 --> for accessRule in {allAccessRulesToApply} do
#21 [accessLevel] = AR1.applyRule[this] on accessRule
#22 if accessLevel < currentLowestAccessLevel then
#23 currentLowestAccessLevel = accessLevel
#24 endif
#25 endfor
#26 endif
#27
#28 [] = PAT3:deletePatientWithConsents[] on this
#29
#30 [] = AU1:informOfAccessLevelForPatient[this.NHSNumber, currentLo
```

Variable Name	Type	Value
this	Instance	DEFINED
patientSpecificAccessRi	Settype	DEFINED
masterAccessRules	Settype	DEFINED
masterAccessRuleSup	Settype	DEFINED
allCandidateAccessRule	Settype	DEFINED
enablingAccessRulesFo	Settype	DEFINED
disablingAccessRulesFr	Settype	DEFINED
allAccessRulesForThe	Settype	DEFINED
allAccessRulesToApply	Settype	DEFINED
currentLowestAccessLev	Integer	2
accessRule	Instance	UNDEFINED
accessLevel	Integer	0

Cancel

Application Simulation

```
DEBUG: **** Running Scenario 2 - 'Initialise
DEBUG: **** Scenario setup complete 2 - 'Init
Access Level : 2
Access Level : 0
Access Level : 2
```

NHSNumber	patientLocationName	itemTypeName	Current_State	R16 Accessor	R17 Type Of Accessible Item	R3 Patient	R30 Signed
1111111111	TEES	EHR	Getting_Current_Consents_f5	1			
1234567890	TEES	EHR	Checking_Current_Consent	5	1		

Cancel

Signal Trace

Src State	Src Id.	Signal	Signal Name	Dest Domain/Class	Dest Id.	Dest Old St.	Dest New St.
6	0	PAT1	allConsentsRetrieve	PDAR.Patient_With_Consents	6	6	7
7	0	PAT1	allConsentsRetrieve	PDAR.Patient_With_Consents	7	6	7

Cancel



Agenda

- MDA positioning
- MDA essentials
- MDA – the envelope
 - Product Line
 - Architecture
- MDA – the real value
- Next Steps



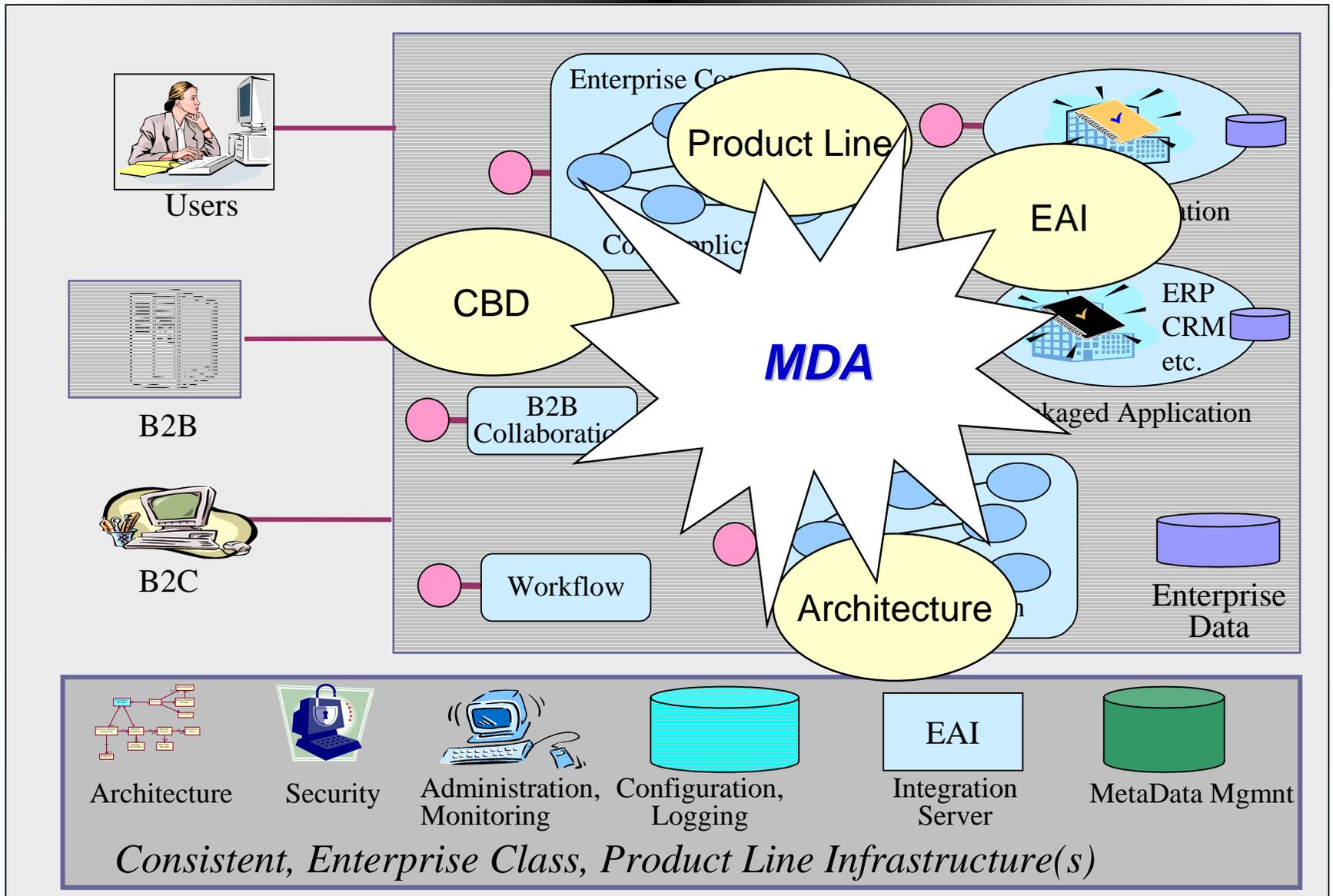


MDA – The Real Value

- MDA possibilities will force middleware upwards (in simplicity and capability)
 - IT organisations will demand higher-level middleware
 - Because IP can be moved up from code to PIMs
 - Computationally-complete PIMs
 - Easier integration at the PIM level
 - Better management of technical churn
 - Recover PSMs and PIMs from legacy
 - Suppliers will start delivering standard “glue” code
 - E.g. concurrency, transactions, client proxy mechanism, client-side frameworks, etc.
 - The “platform” will become more productive, more developer-friendly
- MDA possibilities will drive tools upwards
 - IT organisations will demand higher-function tools
 - Along lines of “perfect MDA tools”
 - Suppliers will provide standard architectures (probably component-based)
 - CBD architecture a priority
 - Architectures will grow in scope and applicability
- Result ...



Managing Complexity





“Naked Objects”

- “Naked objects’ are core business objects, such as Customer, Product, and Order, that show directly through to the user, rather than being hidden behind the menus, forms, process-scripts and dialogue boxes that make up most user interfaces.” (www.nakedobjects.org)
- Used to build working prototypes while business modelling...
- ... and then used to implement the system!
- GUI is automatically-generated



Naked Objects Example (1)

The screenshot displays a software interface for a 'Naked Objects' example. It features a main window titled 'NakedCollection instance' which contains two primary folders: '10 TradingPartys' and '6 Contracts'. The 'TradingPartys' folder lists ten entities, including 'test', 'Hydro Agri Sliskil BV', 'Verinigte Energiewerk AG', 'Hydro Kraft AB', 'RWE Netz', 'E.ON Netz', 'Power-Next Exchange', 'LPX', 'APX', and 'Nordpool Exchange'. The '6 Contracts' folder lists six specific contracts, such as 'E.ON Netz RWE- EON Buy 4000', 'E.ON Netz Tennet- EON Buy 50', and 'Hydro Kraft AB Eltra Buy 120'. On the left side, a vertical sidebar provides navigation options with icons for 'Trading Parties', 'Contracts', 'Locations', 'Links', and 'Networks'. At the bottom of the interface, three smaller windows are open, showing 'European Network', 'Tennet- EON', and 'Tennet'.



Naked Objects Example (2)

The screenshot displays a software interface with four main windows and a sidebar. The sidebar on the left contains icons for Trading Parties, Contracts, Locations, Links, and Networks. The windows are as follows:

- European Network**: Shows a list of locations (Eltra #10, Tennet #11, EON #12, RWE #10, RTE #12) and links (Eltra-EON #, Tennet-EON, RWE-EON #, Tennet-RWE, RTE-RWE #).
- European Network #00**: Shows details for a specific network instance on 16-Aug-2002, including link positions (e.g., 0 out of 100 #105) and location positions (e.g., Eltra 120 / 120 Long, Tennet 60 / 60 Long).
- E.ON Netz RWE- EON Bu**: Shows details for a long-term contract, including partner (E.ON Netz #), link (RWE-EON #), location, volume (Buy 4000 #70), and price (£0.00).
- Location instance**: Shows details for the Tennet #11 location, including exchange (APX #17) and network (European Network).



Naked Objects Example (3)

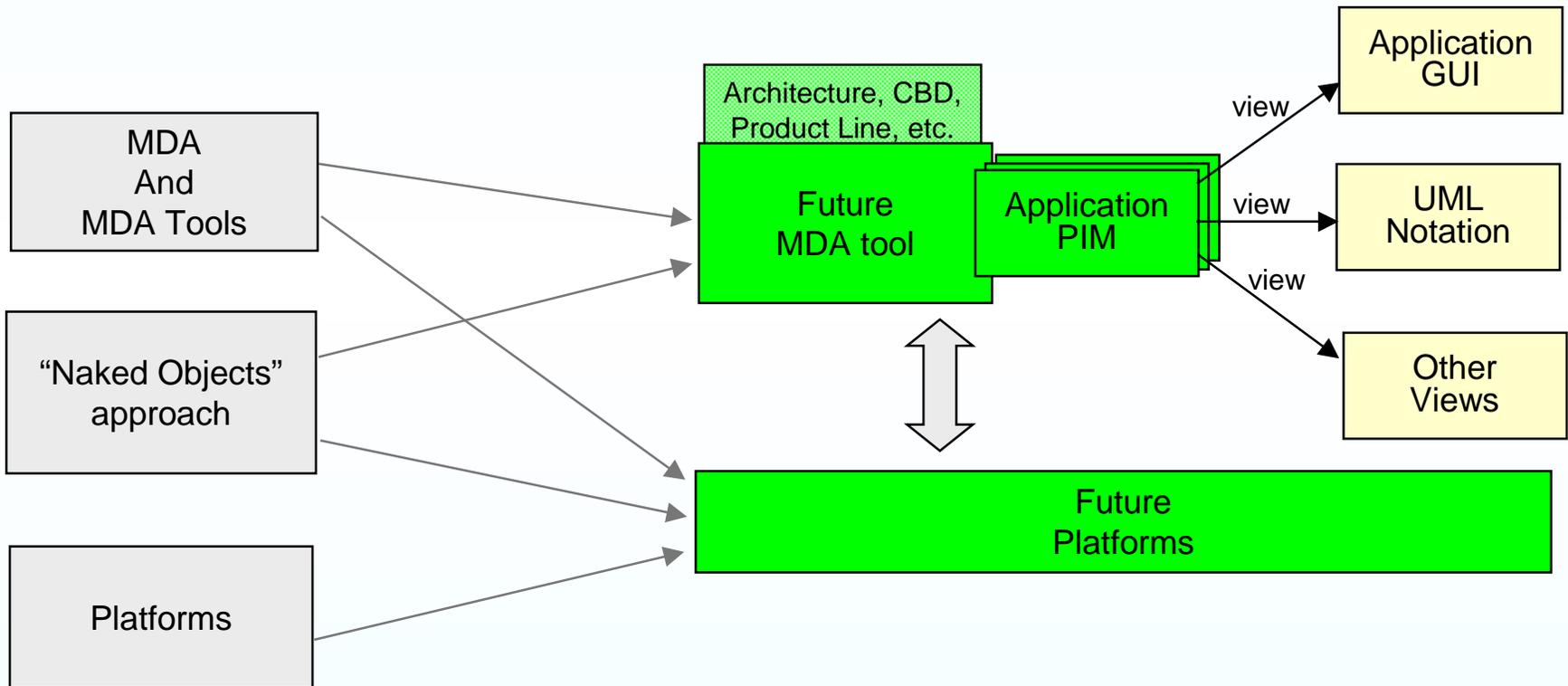
The screenshot displays a software interface with a sidebar on the left and a main map area on the right. The sidebar contains icons for Trading Parties (factory), Contracts (handshake), Locations (target), Links (two red dots), and Networks (triangle with lines). The main map area, titled "SpatialCollection instance", shows a map of Europe with several trading parties and contracts. A context menu is open over a green triangle icon, listing options: Inspect..., As Form, As Text, Destroy object, Remove reference, Buy At Exchange..., Create New Contract..., Sell At Exchange... (highlighted), and Clone... The map also shows labels for trading parties: "Eura 120 / 120 Long", "Tennet 60 / 60 Long", and "RTE 0 / 0 Balanced". Contracts are labeled with "0 out of 100", "0 out of 50", and "0 out of 0".



Possible MDA Evolution

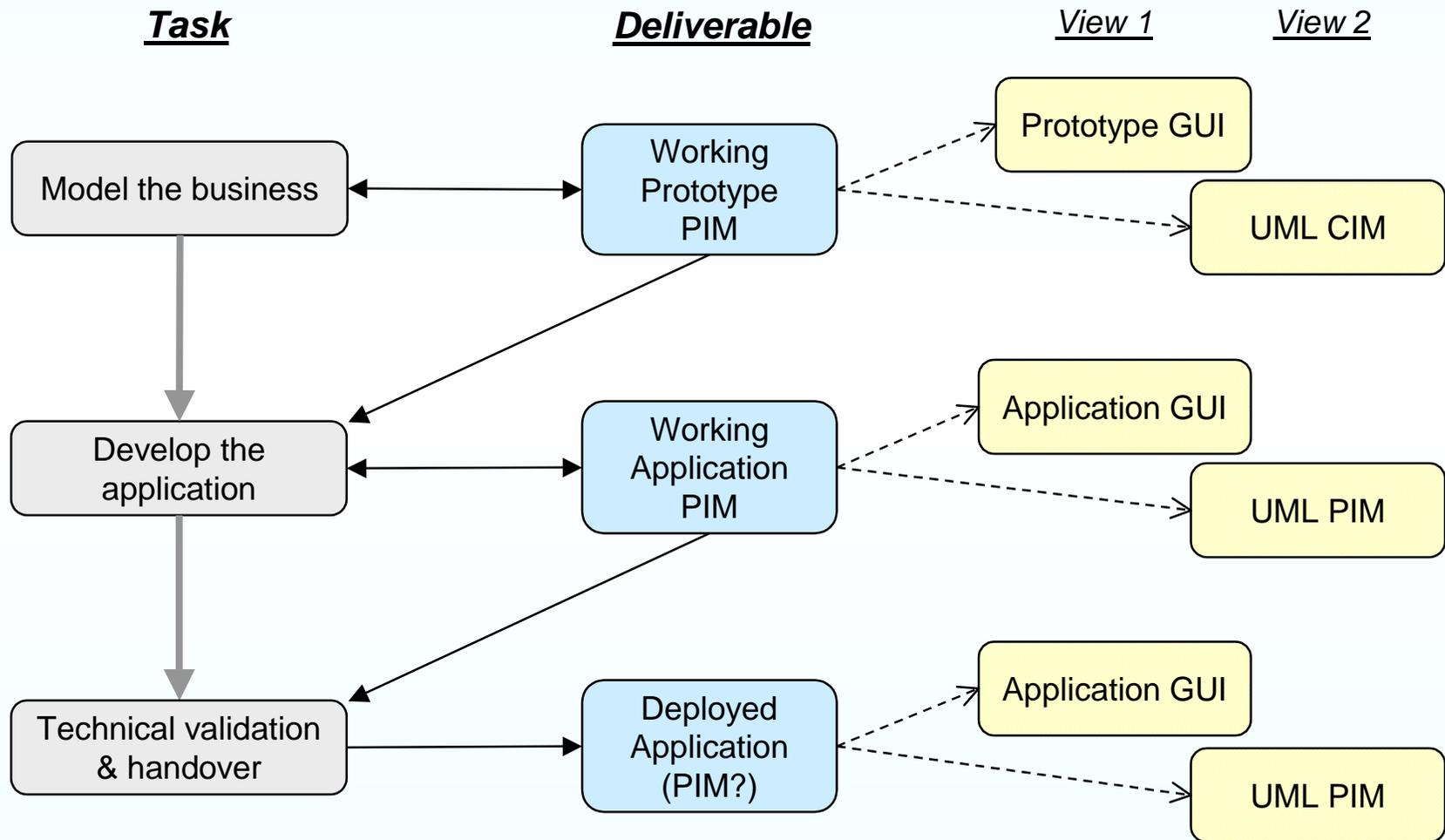
Today

Future





Future MDA Development Process?





Agenda

- MDA positioning
- MDA essentials
- MDA – the envelope
 - Product Line
 - Architecture
- MDA – the real value
- Next Steps





Next Steps

- Tomorrow:
 - Schedule an MDA planning meeting
- Next week:
 - Review the current development situation
 - Define the goal
 - Plan the first step
 - Define process for evolution to the goal
while continuing with development schedule
- Next month – Start!
 - Launch the first MDA project, using an MDA tool
 - Real application
 - Architecture capture
 - Product Line initiation
 - Manage the evolution to the goal
- 😊 ... call us if you need help!

MDA

-

The Real Value

Oliver Sims

Sims Associates

oliver.sims@simsassociates.co.uk