



“Toward Completing A Vision”

***A Preview of Version 7 of the IBM Rational
Software Model Driven Development Product
Family***

William T. Smith (smithtw@us.ibm.com)

***Product Manager, Model Driven Development, IBM
Rational***

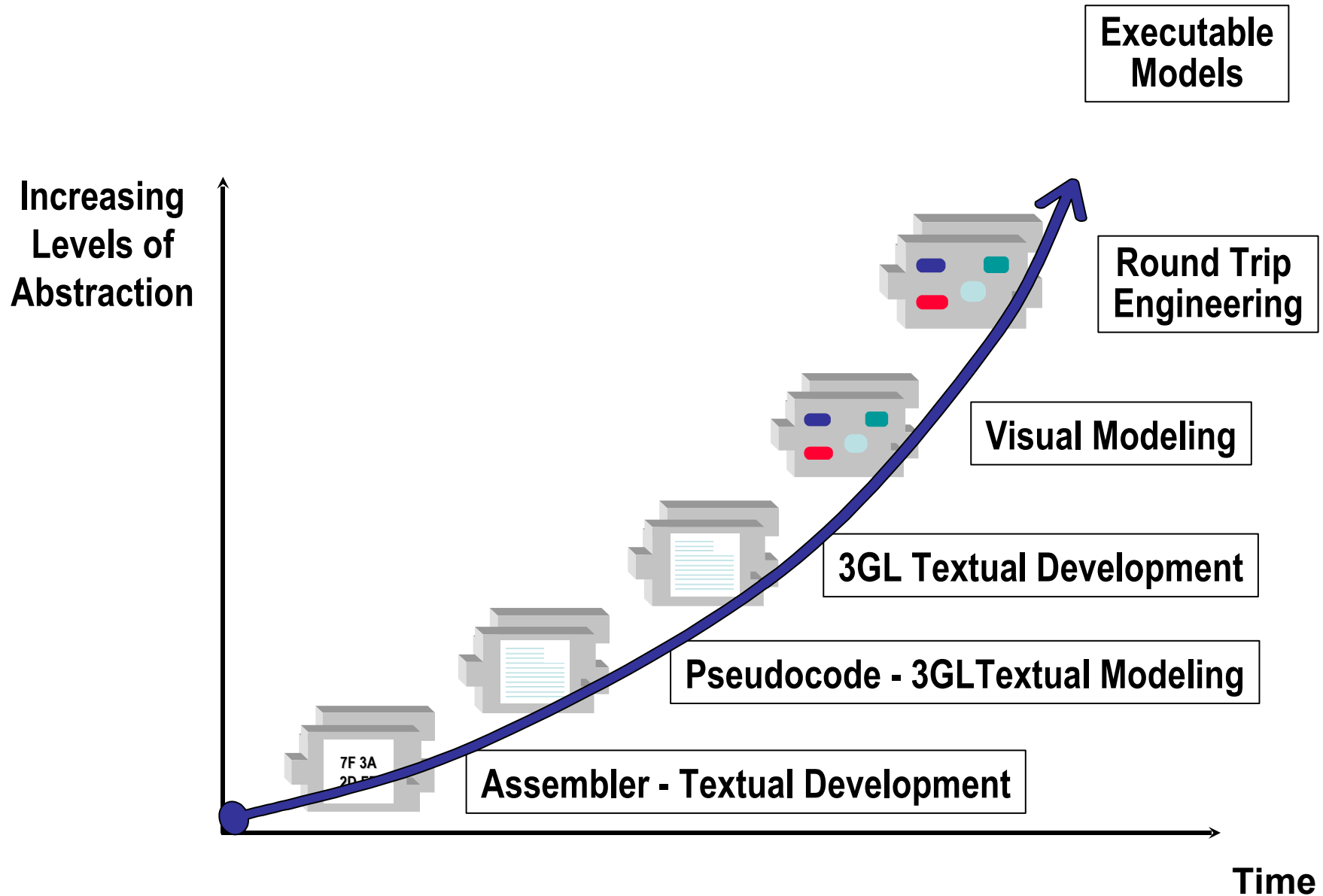
Agenda



- *The Rational MDD Heritage*
- *The New MDD Product Family: Vision and Current State (v6)*
- *The New MDD Product Family: Toward Completing The Vision (v7)*



The Abstraction Curve



- ***UML was new***
- ***C/C++ were dominant languages, Java just emerging***
- ***IDEs were not common***
- ***Frameworks and runtimes we take for granted today, did not exist***

Rational Rose emerged as...

SEE DIAGRAMS:
OrganizationalEnvironment
HealthCareFinancing

```

classDiagram
    class Contract["Contract (from Organizational Classes)"]
    class ProviderEngagement["ProviderEngagement (from Organizational Classes)"]
    class HealthcareCoverageContract["HealthcareCoverageContract (from Organizational Classes)"]
    class ProviderContract["ProviderContract (from Organizational Classes)"]
    class ContractBase["Contract (Classes)"]

    Contract <|-- ProviderEngagement
    Contract <|-- HealthcareCoverageContract
    Contract <|-- ProviderContract
    ContractBase <|-- Contract

    Contract "1..n" -- "1..1" : +is incorporated in
    Contract "1..n" -- "1..1" : +may specify
    HealthcareCoverageContract "0..n" -- "0..n" : +may incorporate
    ProviderContract "1..n" -- "1..1" : +may specify
  
```

- *Language- and platform-agnostic*
- *Stand-alone (not in an IDE shell)*
- *Based on a proprietary UML 1.x metamodel*
- *Later introduced model-code synchronization ("RTE") to support construction activity*

A ProviderEngagement could follow the following:
 1) A Provider being employed by a ProviderOrganization
 2) A Provider entering into a service to a ProviderOrganization basis (e.g. locum tenens).

Log

For Help, press F1

Market Landscape, 1999-2001

- ***Broad and enthusiastic adoption of Java***
- ***Emergence of J2EE framework and runtimes***
- ***Emergence of .NET***
- ***Introduction of Eclipse***
- ***IDEs became ubiquitous in the Java and .NET worlds***
- ***Rose began to face competition from code-centric, in-IDE modeling products***

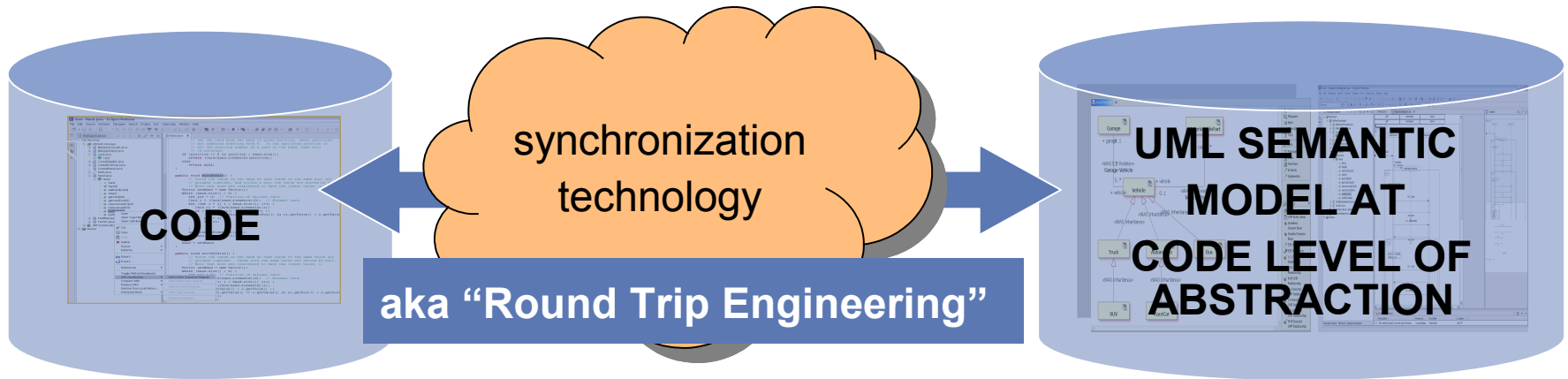
Rational XDE evolves as...

The screenshot shows the Rational XDE Modeling environment. The main workspace displays two UML class diagrams: **SimpleMain** and **MortgageInfo**. The **MortgageInfo** class has the following attributes: + Principal : double, + Rate : double, + Term : double, + TotalAmount : double, + TotalInterest : double, and a method # finalize (). The **Model Explorer** on the right shows a project structure with files like Calculator.java, MortgageInfo.java, SimpleMain.java, and Mortgage.java. The **Properties** window at the bottom right shows properties for the 'Mortgage Main Diagram' such as GridSize (250), SnapT... (True), and Type (1 - CLASS).

- *Biased toward Java/J2EE and .NET*
- *Hosted within IDE environment*
- *Based on another proprietary UML 1.x metamodel*
- *Still using model-code synchronization to support construction activity*

- *Limitations of model-code synchronization were showing*
- ...
- ...

Model-Code Synchronization in Rose and XDE

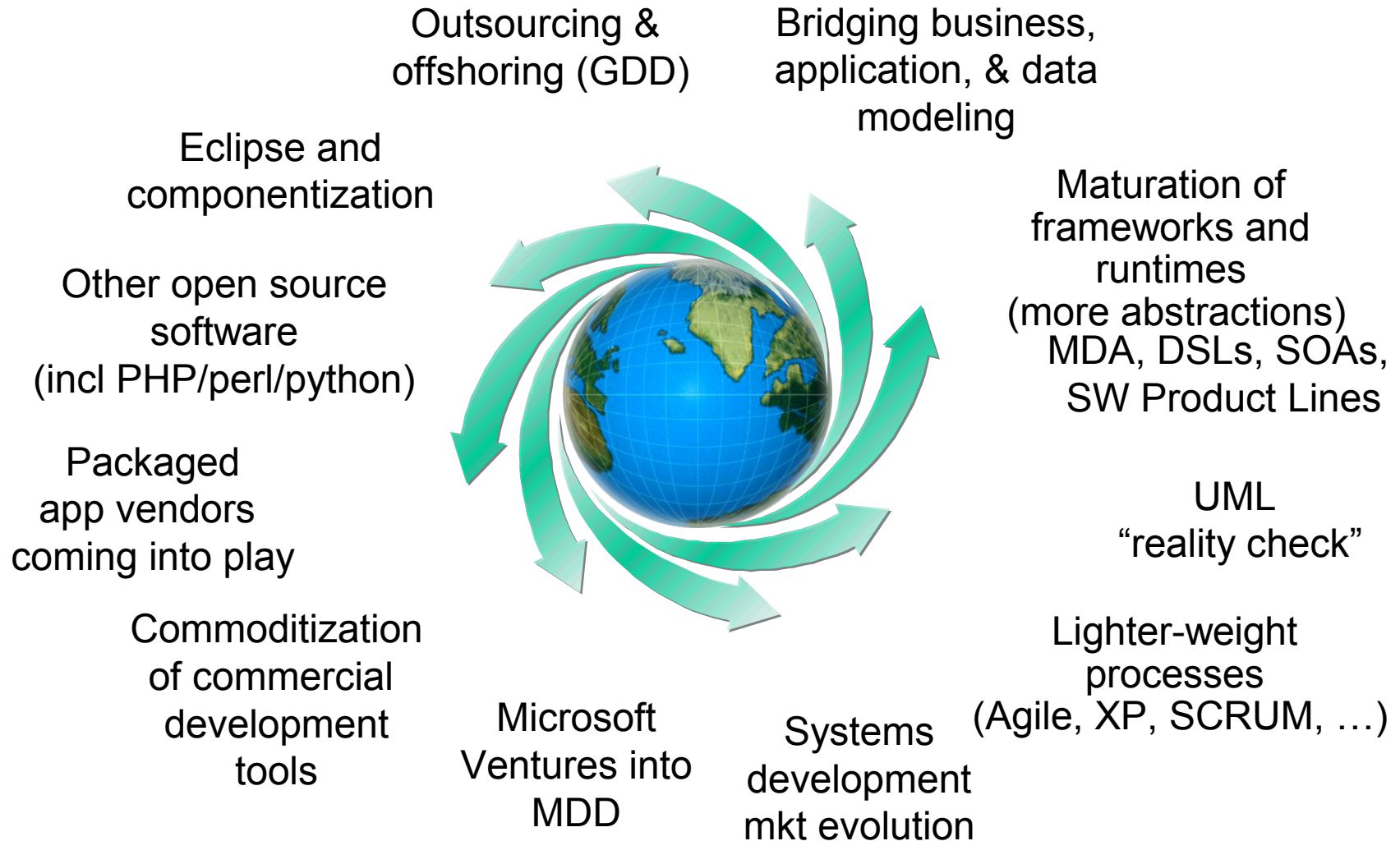


- **Performance overhead**
- **Anomalous sync behaviors**
- **Tendency to drag design down to the implementation level of abstraction**
- **Code refactoring must process two sets of semantics**
- **Challenging team workflows**

A UML model persists code semantics redundantly.

The model is at the same level of abstraction as the code

Industry trends for Analysis Design and Construction



Agenda

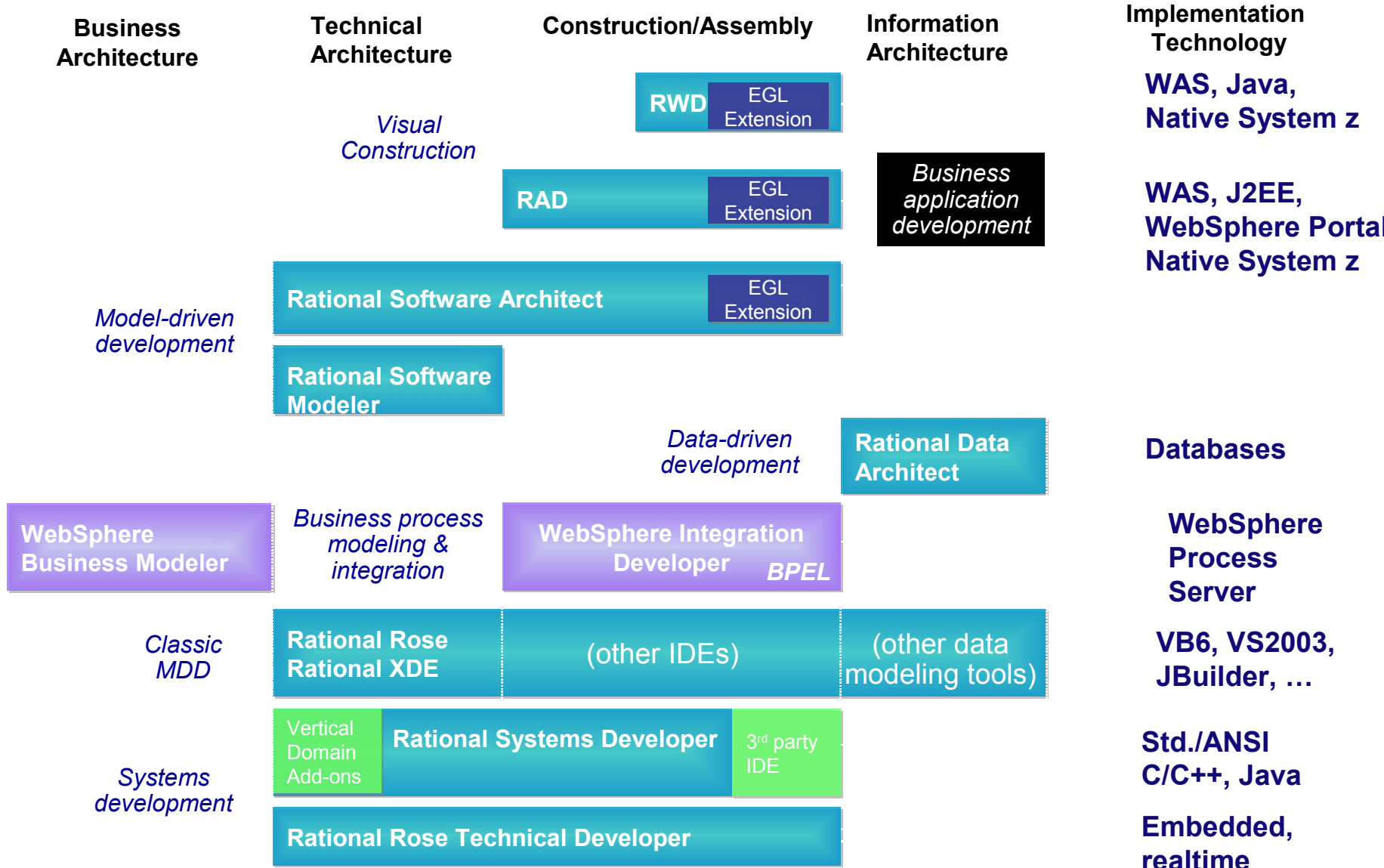


- *The Rational MDD Heritage*
- *The New MDD Product Family: Vision and Current State (v6)*
- *The New MDD Product Family: Toward Completing The Vision (v7)*





Analysis Design & Construction products



Analysis, Design, and Construction

“an investigation of constituent parts and their interrelationships making up a whole.”

- Business Process Analysis
- Model Problem Domain
- Write Requirements
- Model *Possible* Solution(s)

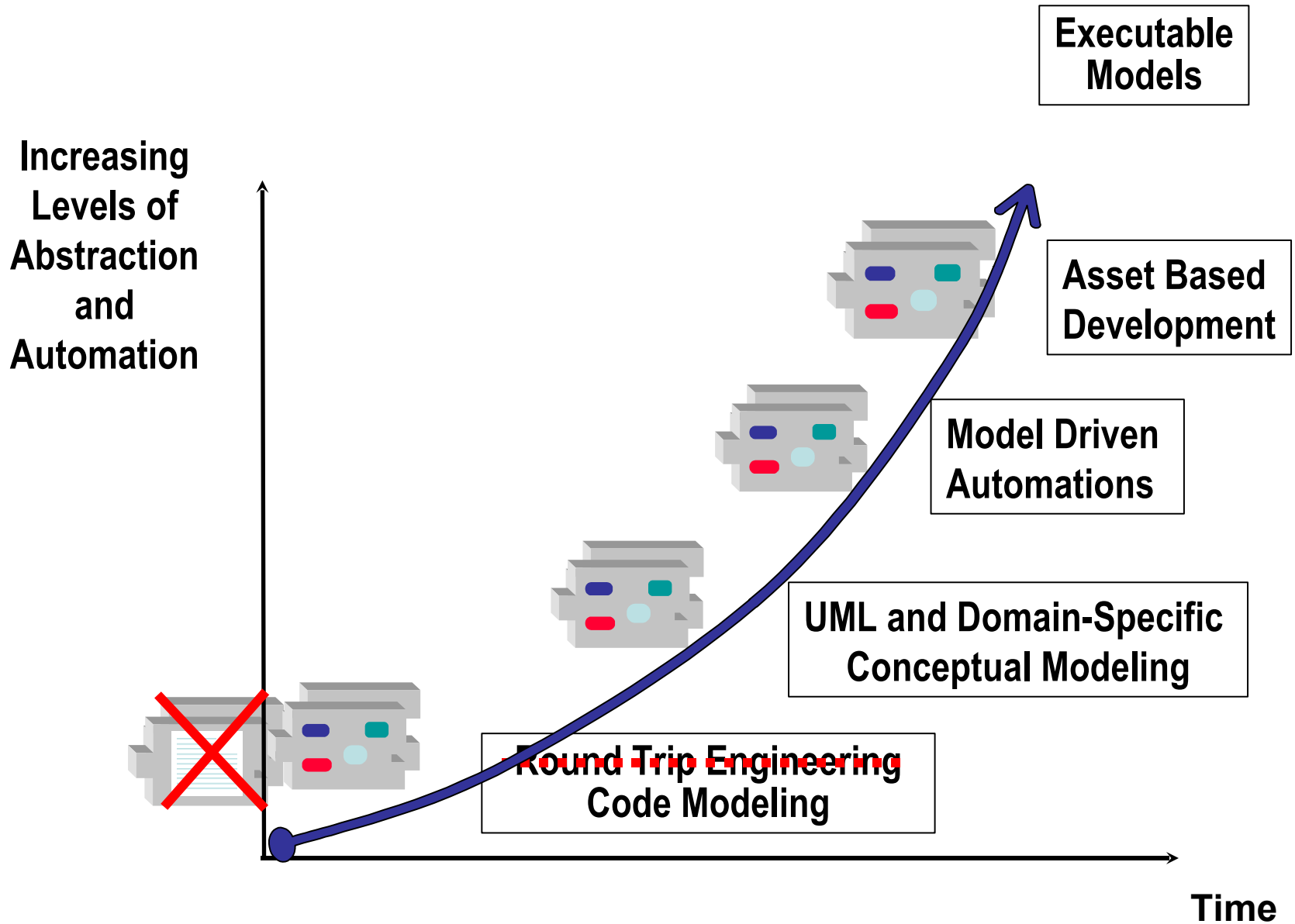
“the act of working out the form of something (as by making a sketch or outline or plan)”

- Model solution *intent*

“the act of constructing or building something”

- Implement Solution
- Model solution *reality*

The Abstraction Curve Revisited



Key Elements of the MDD Vision

- **Business value**
 - Simply supporting UML modeling and RTE is no longer enough
 - MDD must return more repeatability
 - MDD must return more automation of repetitive development tasks
 - MDD must bridge and integrate domains (business domains, activity domains, problem domains, solution and technical domains)
 - MDD must be better integrated with other aspects of the development process and the tools that support them
- **Flexibility**
 - Support multiple MDD “theories of operations”, corresponding to multiple development governance philosophies
- **Extensibility**

Rational Software Modeler, Architect evolve as...

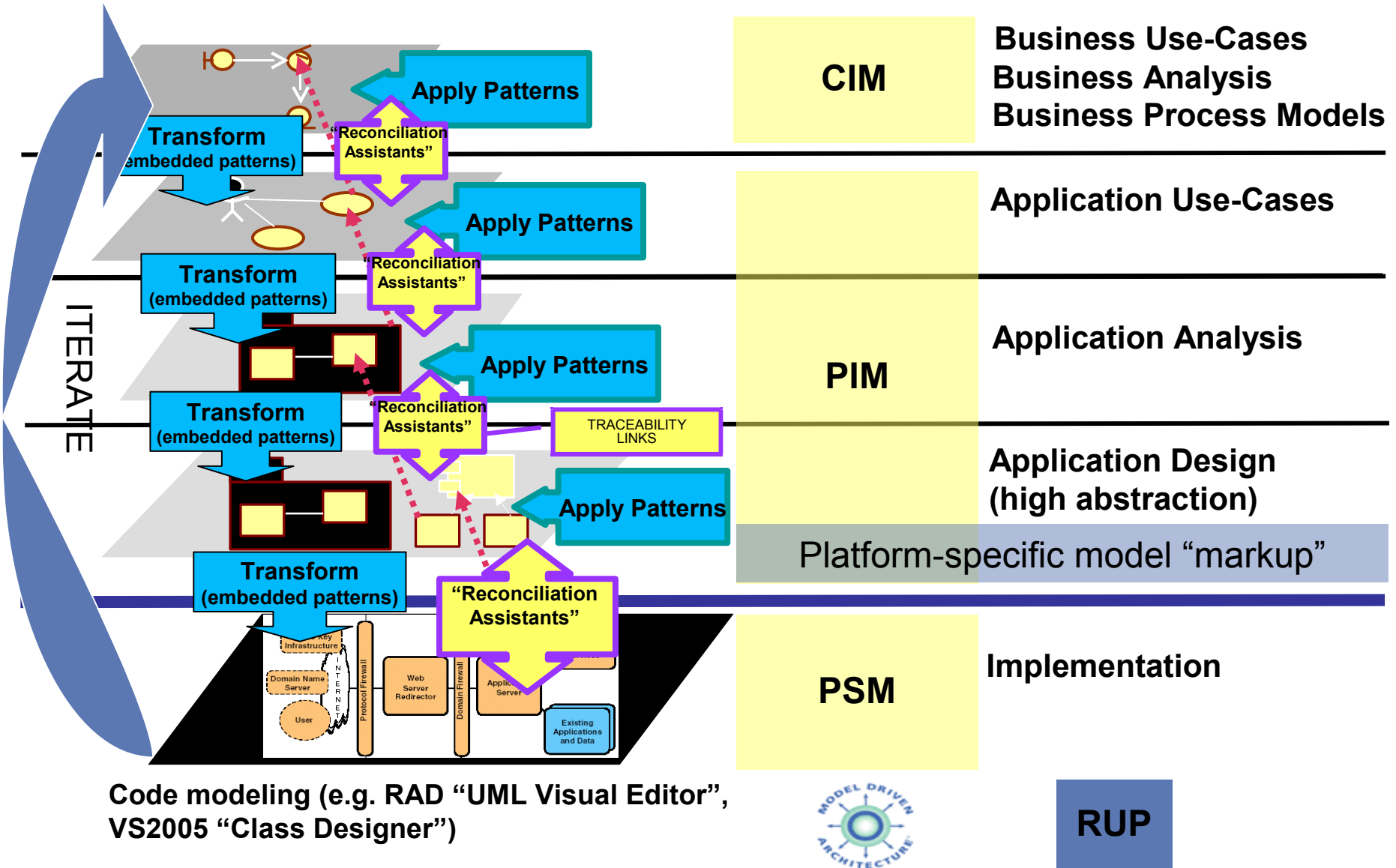
The screenshot displays the IBM Rational Software Modeler interface. The main workspace shows two UML class diagrams. The first is an entity class named 'Item' with attributes: catalogNumber (Integer), title (String), description (String), value (Integer), and startingBid (Integer). The second is an entity class named 'Status' with attributes: <id> statusid (Integer) and name (String). An association line connects the two classes, labeled '+ fk_itemStatus'. The interface includes a menu bar, a toolbar, a palette on the right with various diagram elements, and a project explorer on the left.

- **Eclipse-based**
- **Leveraging Eclipse infrastructure to support model-based integration and automation capabilities**
- **Using open source UML2 metamodel**
- **Using code modeling (DSLs) instead of UML with “RTE”**
- **Reflecting an updated, more ambitious vision for Model Driven Development**

Key Elements of the MDD Vision

- **Business value**
 - Simply supporting UML modeling and RTE is no longer enough
 - MDD must return more repeatability
 - MDD must return more automation of repetitive development tasks
 - MDD must bridge and integrate domains (business domains, activity domains, problem domains, solution and technical domains)
 - MDD must be better integrated with other aspects of the development process and the tools that support them
- **Flexibility**
 - Support multiple MDD “theories of operations”, corresponding to multiple development governance philosophies
- **Extensibility**

Model Driven Development With Rational



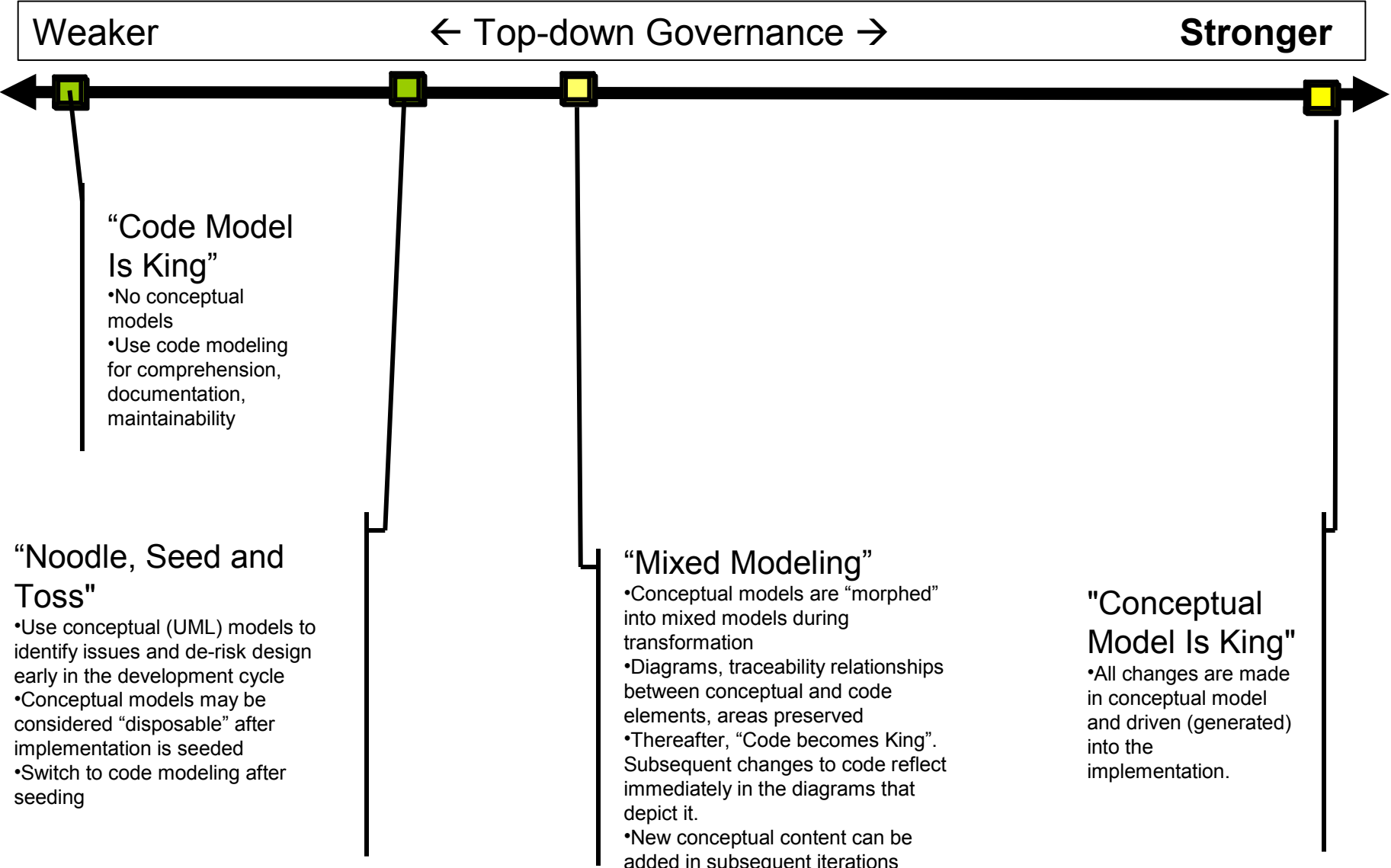
Key Elements of the MDD Vision

- **Business value**
 - Simply supporting UML modeling and RTE is no longer enough
 - MDD must return more repeatability
 - MDD must return more automation of repetitive development tasks
 - MDD must bridge and integrate domains (business domains, activity domains, problem domains, solution and technical domains)
 - MDD must be better integrated with other aspects of the development process and the tools that support them

- **Flexibility**
 - Support multiple MDD “theories of operations”, corresponding to multiple development governance philosophies

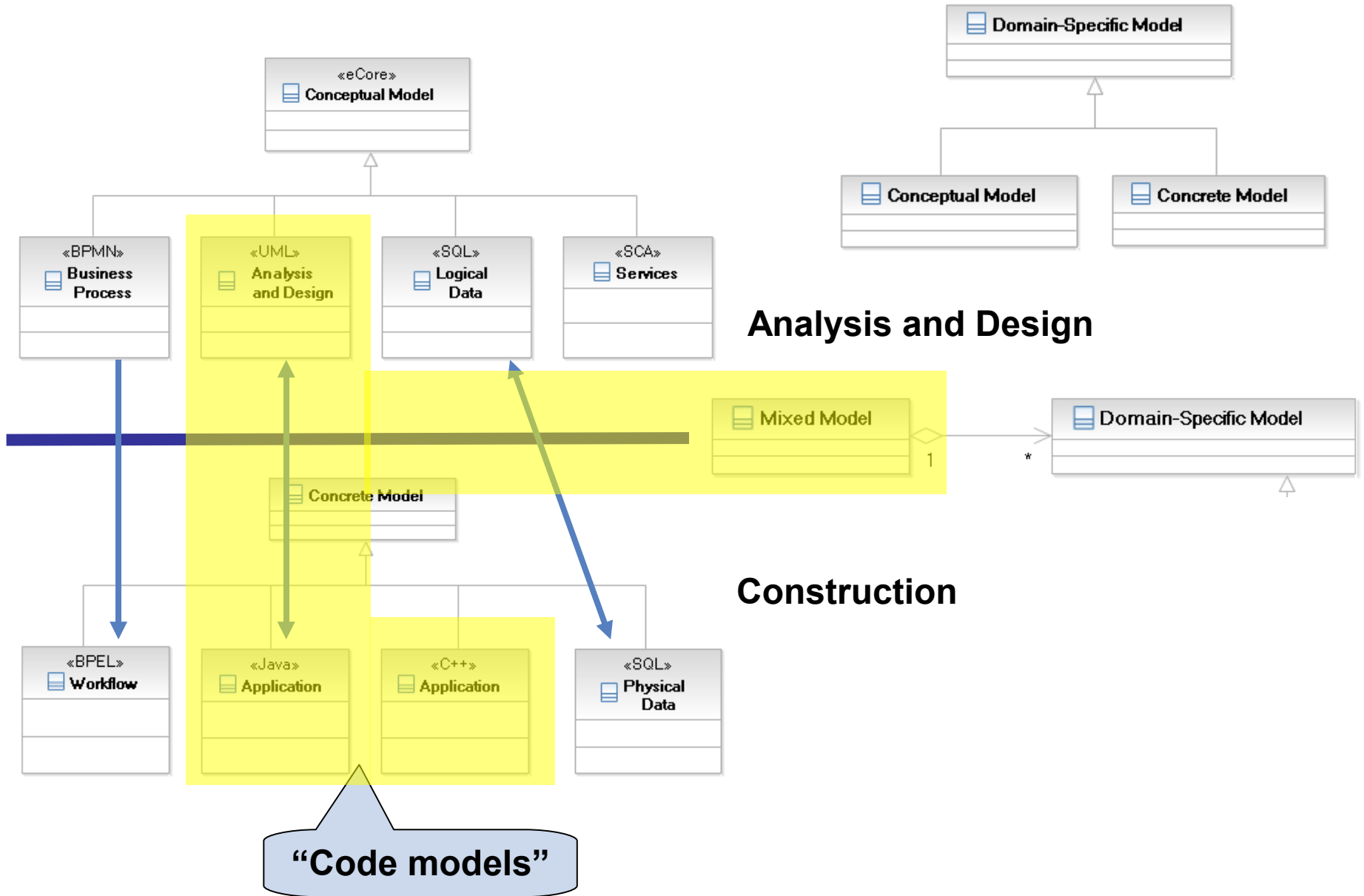
- **Extensibility**

MDD “Theories of Operations” Map To Governance Philosophies

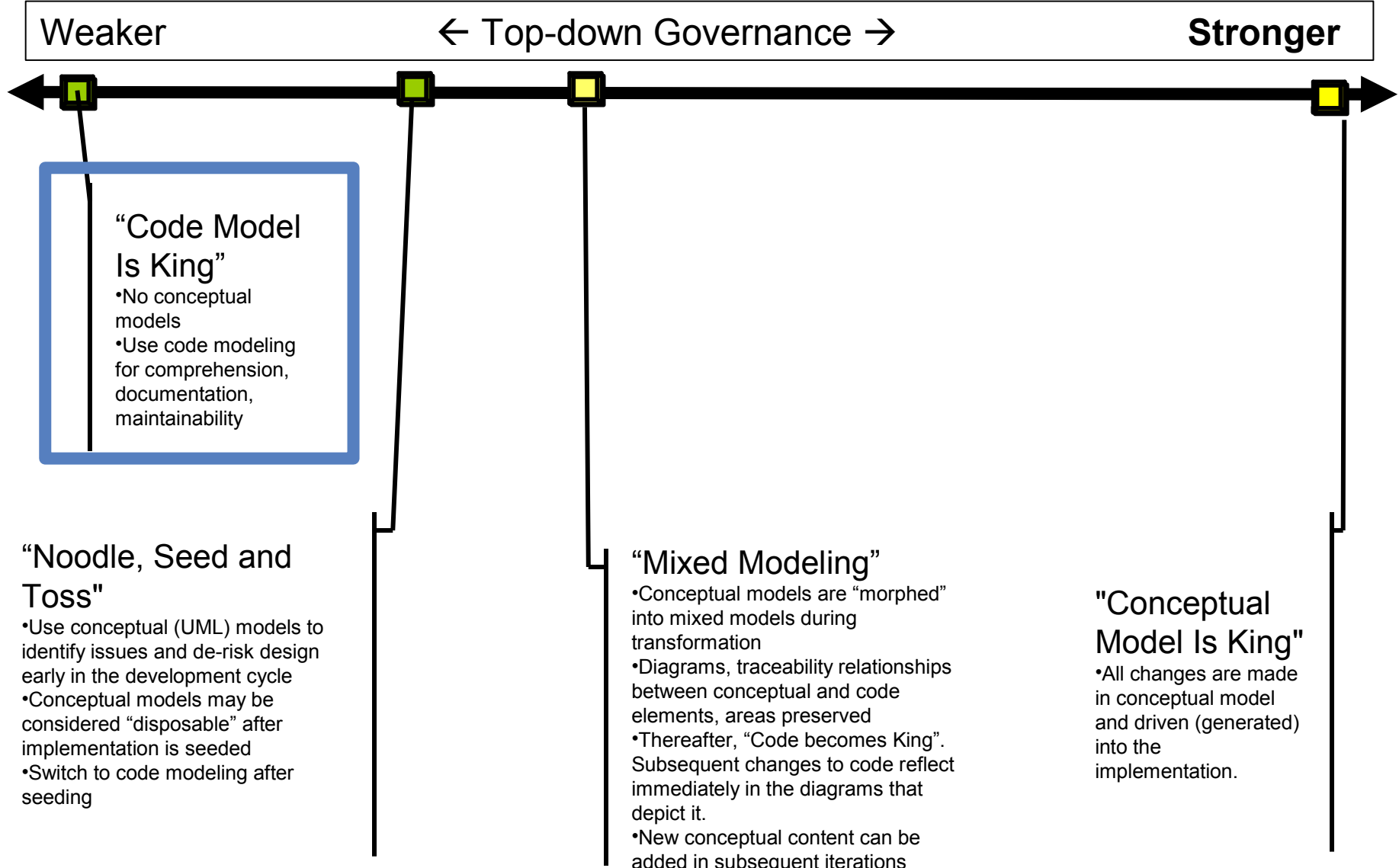


- **Conceptual Model (generalization)**
 - A model based on a semantic domain that is more abstract and typically more general-purpose than a 3GL or other semantic domain that directly reflects an implementation technology
- **UML Model**
 - A type of conceptual model based on UML semantics (i.e. a UML metamodel)
 - Contains UML semantic elements plus UML diagrams whose notational elements reflect underlying UML semantic elements
 - In RSx, persisted as “UML model files” (with .emx extensions) typically contained in UML-natured projects
- **Domain-Specific Model (generalization)**
 - A model based on a particular semantic domain (i.e. a metamodel of that domain) be it UML, Java, CLR, C++, CORBA IDL, DDL, WSDL, SCA, ...
 - IBM thinks of UML as an ‘analysis and design’ semantic domain
 - IBM thinks of Java, CLR, etc. as ‘construction’ semantic domains
 - In RSx, implementation details may vary -- metamodels may be defined in EMF or sometimes other technologies
 - In RSx, defining the scope of any logical model is somewhat arbitrary. A logical UML model may span multiple UML model files that reside in one or more UML projects. A logical Java model might reside in one or more Java projects
- **Code Model**
 - A model based on 3GL semantics (e.g. a Java metamodel or CLR metamodel)
 - Contains 3GL semantic elements and relationships, plus ‘free standing’ diagrams whose notational elements reflect underlying 3GL semantic elements
 - In RSx, serialized and persisted as source code files (semantics) and diagram files (with .dnc extensions) contained in 3GL-natured projects
- **Mixed Model**
 - A shorthand way of referring to a model that contains elements of semantic domain “X” plus diagrams that reflect elements of domain “X” as well as other semantic domains
 - To date, RSx has not implemented any features that persist mixed-domain semantics within a single file

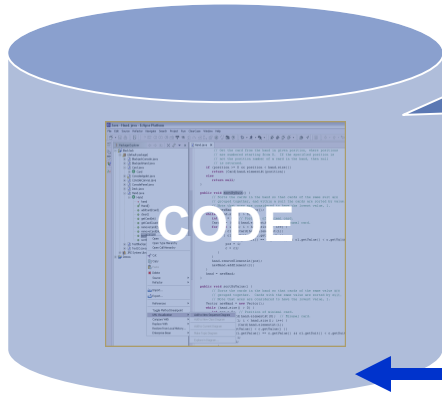
Models Taxonomy



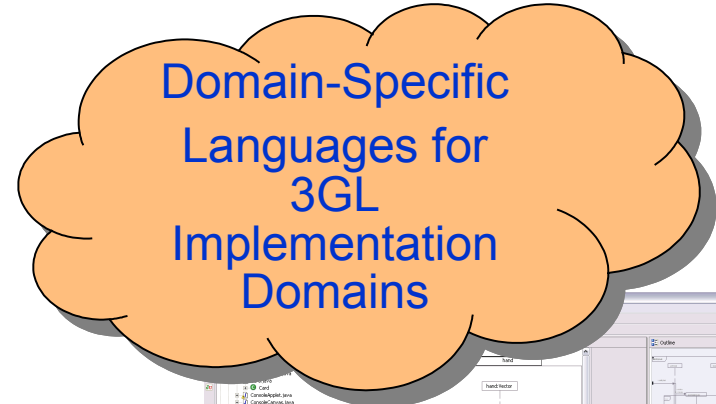
MDD “Theories of Operations” Map To Governance Philosophies



Code Modeling with Rational

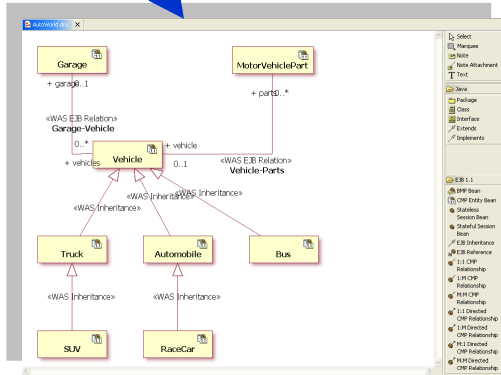
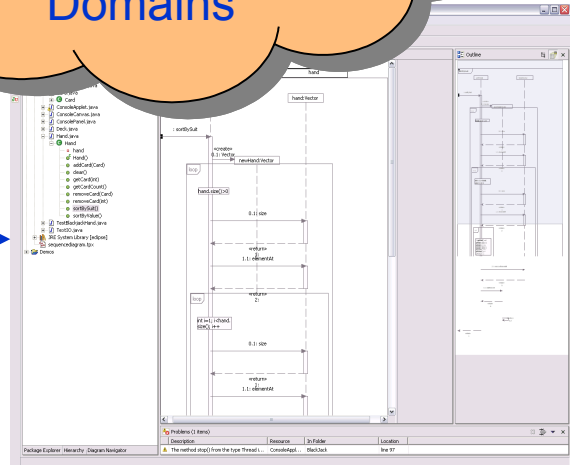


The only source of semantic information is here



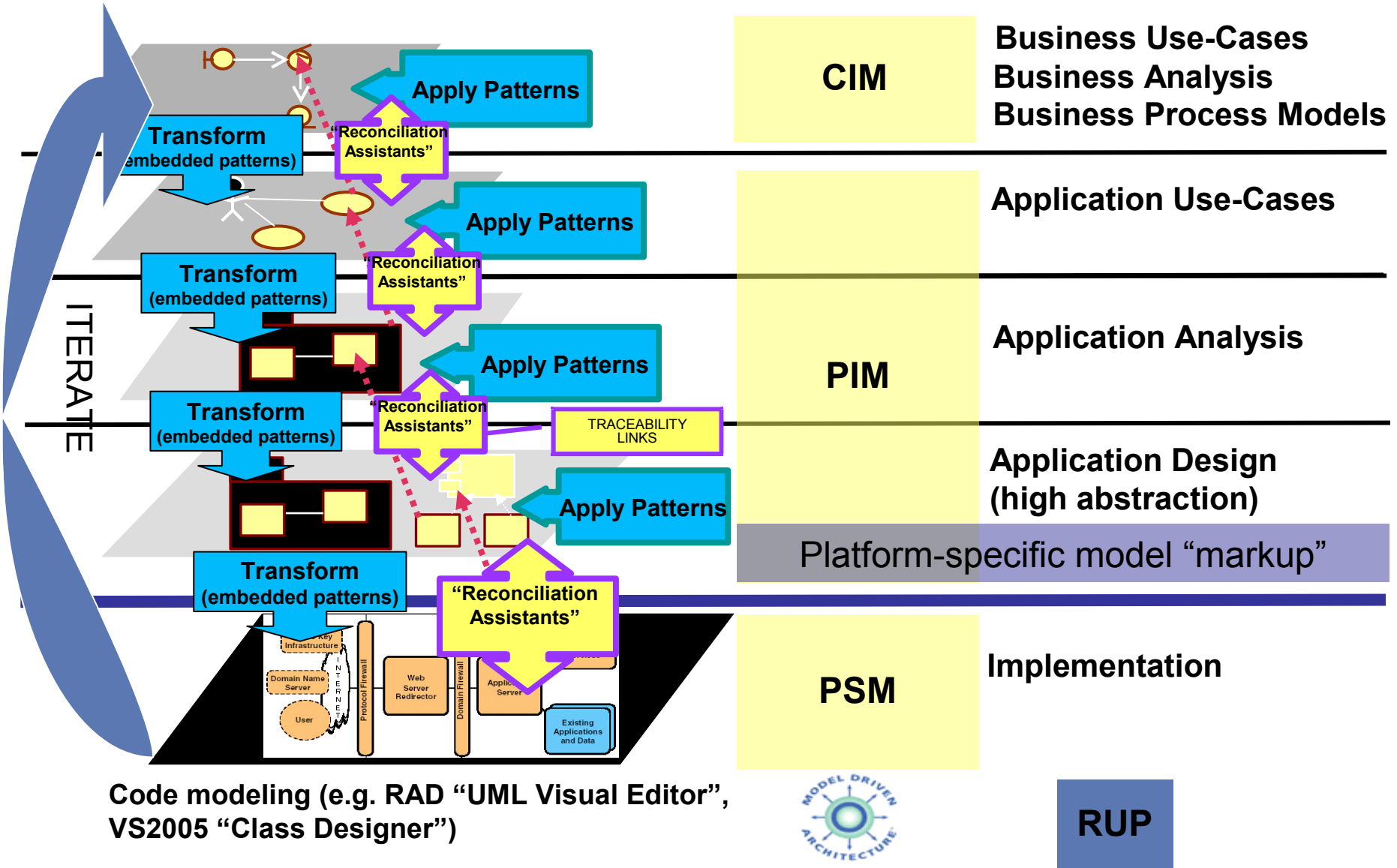
Domain-Specific Languages for 3GL Implementation Domains

Diagrams that directly reflect and in some cases edit the source code



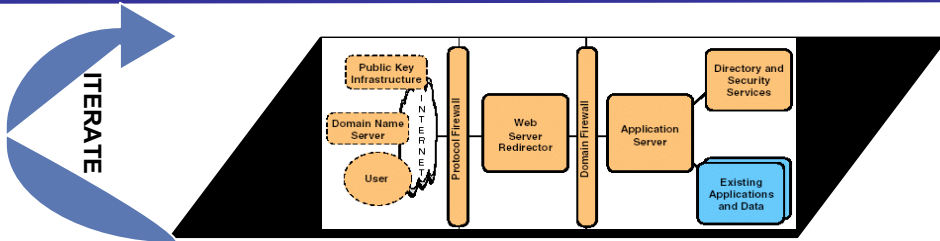
- Code and diagrams are always in sync
- Team workflows are simple
- Code refactoring takes care of everything

“Code Model Is King”



“Code Model Is King”

Analysis, design



Code modeling (e.g. RAD “UML Visual Editor”, VS2005 “Class Designer”)

Implementation

RUP “Agile”
Configuration



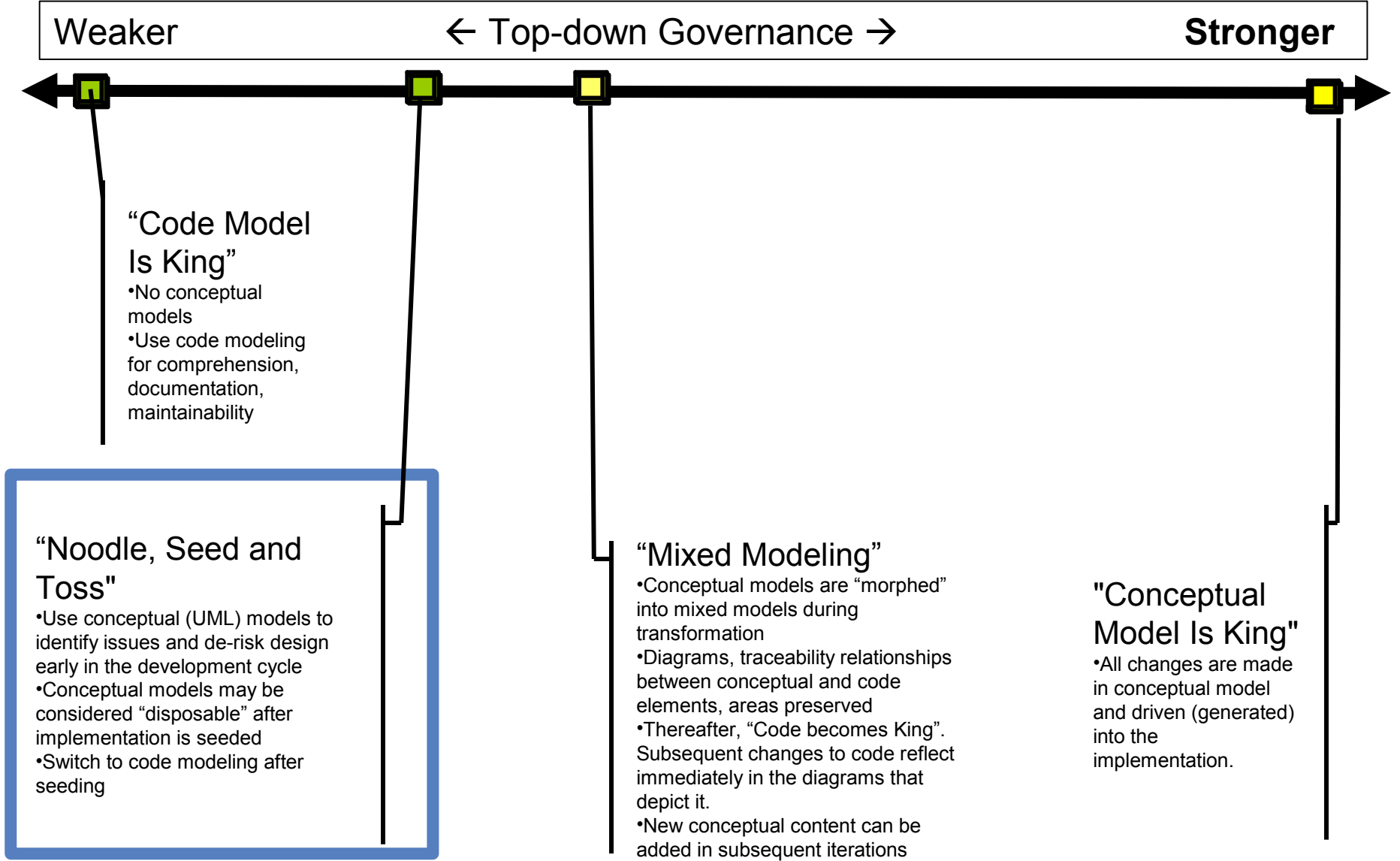
Demo:

“Code Model is King”

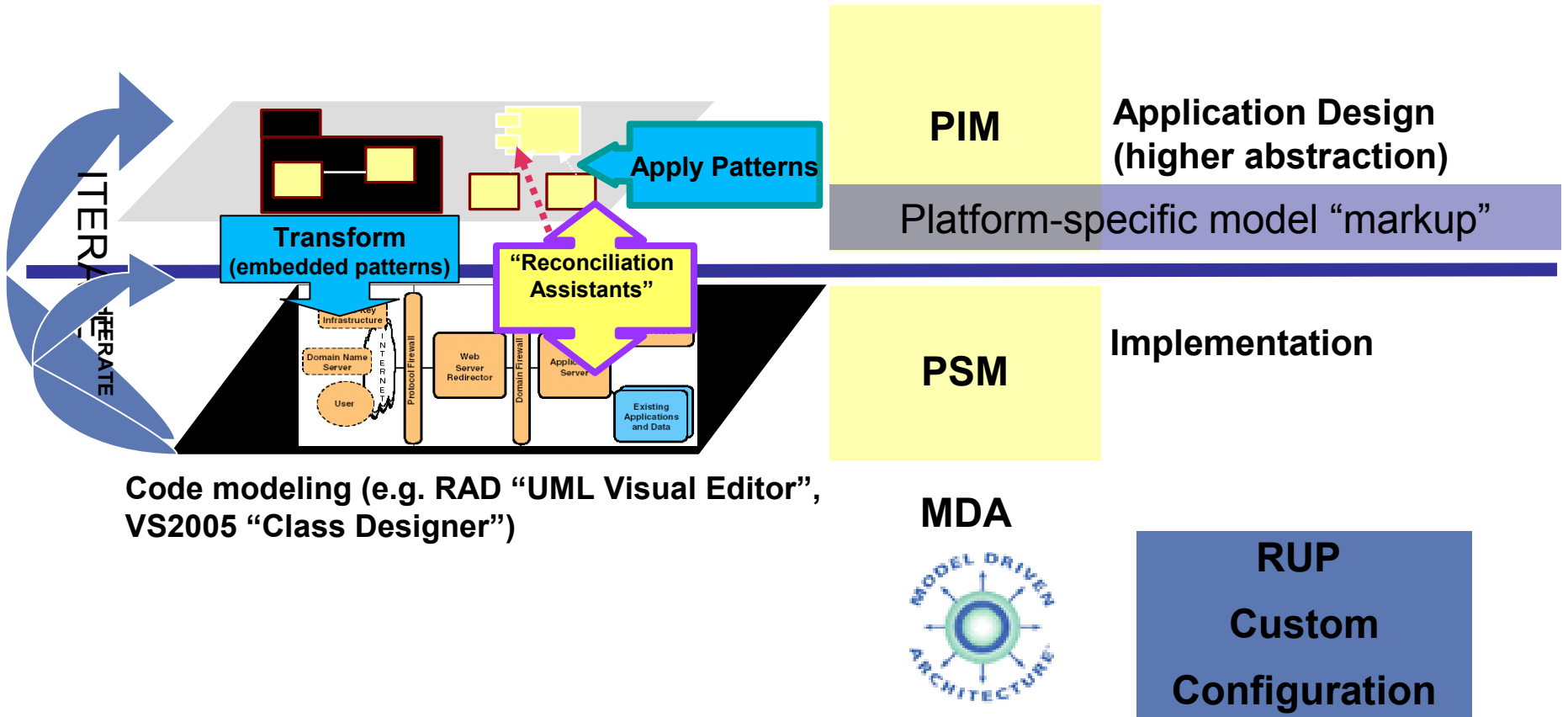
“Code Model Is King” Value Proposition

- *Documentation, Compliance:*
 - “My manager/architect wants UML diagrams”
 - “My {customer | general contractor | regulator} requires UML diagrams”
- *Comprehension*
 - “I don’t understand the code that Jacques wrote”
 - Where’s Jacques?
- *Maintainability, creativity*
 - “I’m a developer, I don’t want to have to learn UML and work with model files, but I like the freedom and clarity that diagrams bring to my design process”
- *Low/No Cost*
 - I don’t have to learn UML semantics
 - I don’t have to work with model files or perform model diff-merges
 - It’s a completely natural fit into my development environment

MDD “Theories of Operations” Map To Governance Philosophies



"Noodle, Seed, and Toss"





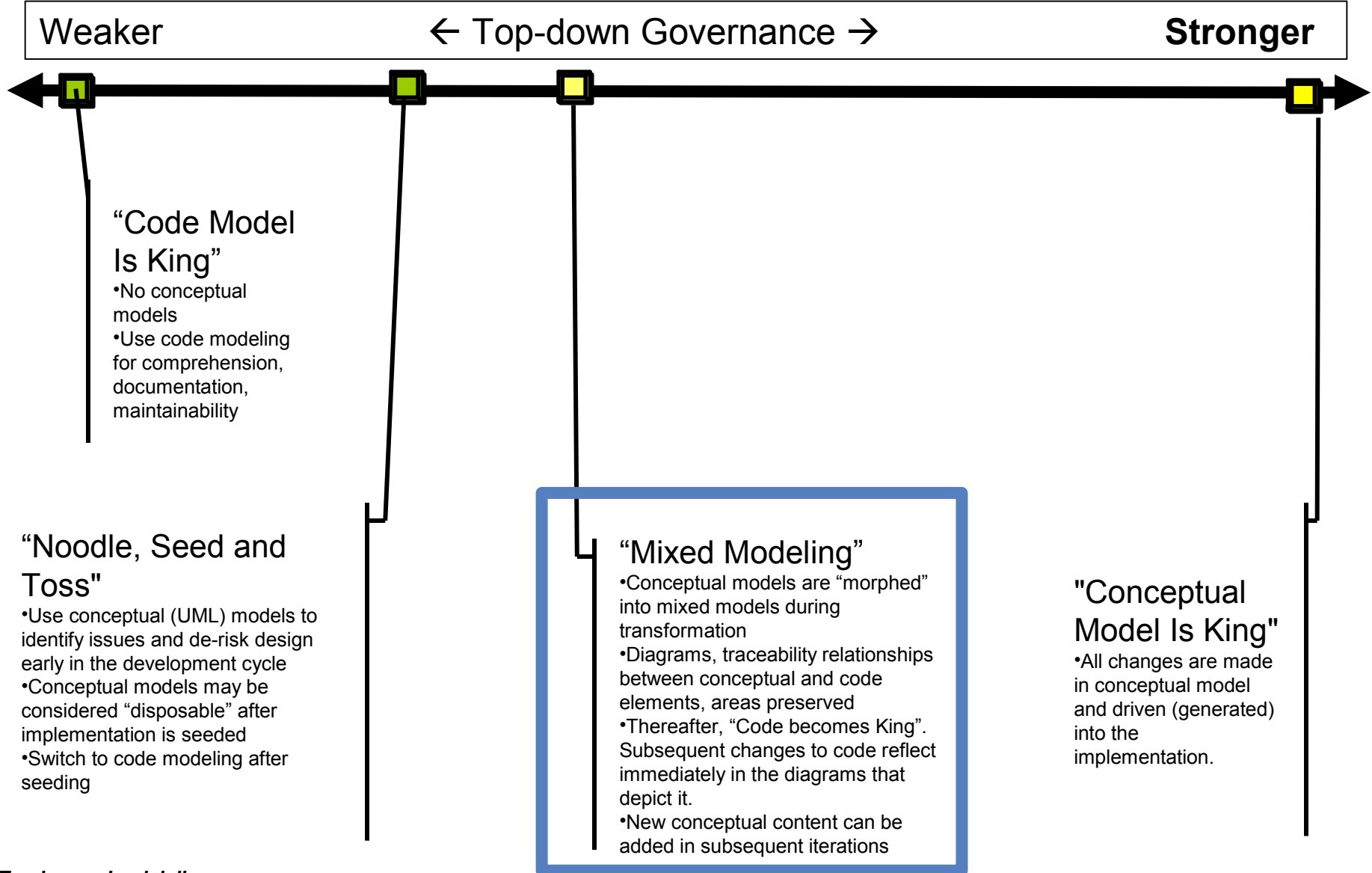
Demo:

“Noodle, Seed, and Toss”

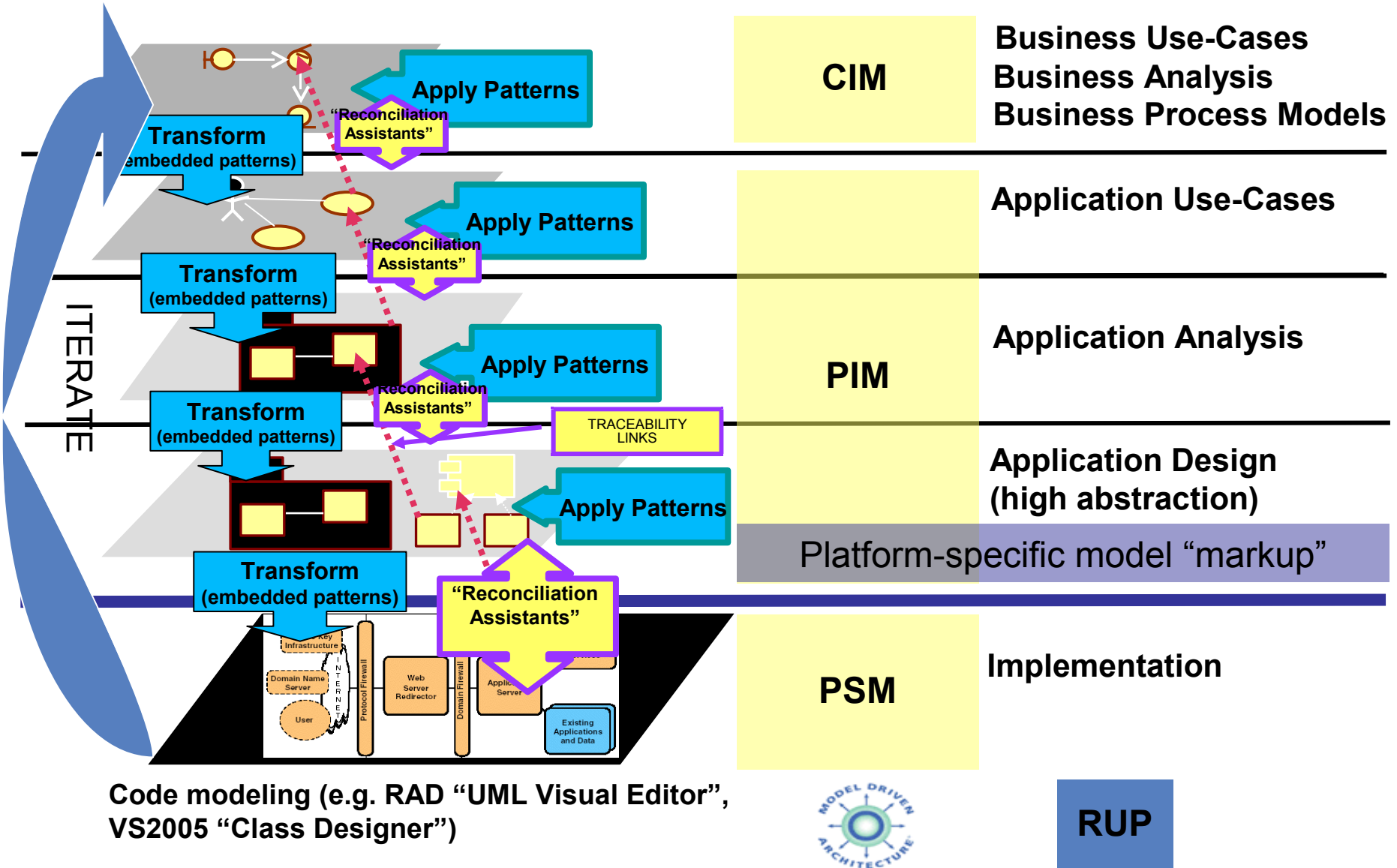
“Noodle, Seed, and Toss” Value Proposition

- *“Light” Governance:*
 - “My architect gave me a bunch of UML that I must use as a starting point...”
 - “My {customer | general contractor | regulator} gives me design contracts in the form of UML models”
 - “We want to be able to do use-case and/or activity and/or state machine and/or instance modeling”
- *Risk avoidance*
 - Conceptual models can be modified more quickly and at less cost than code models, during early stages of specification and design
- *Automation*
 - “I can make one-time use of patterns and transformations based on a UML source model”
- *Low/No Cost*
 - “Once we throw away the original conceptual model, we no longer have to work with model files or perform model diff-merges”

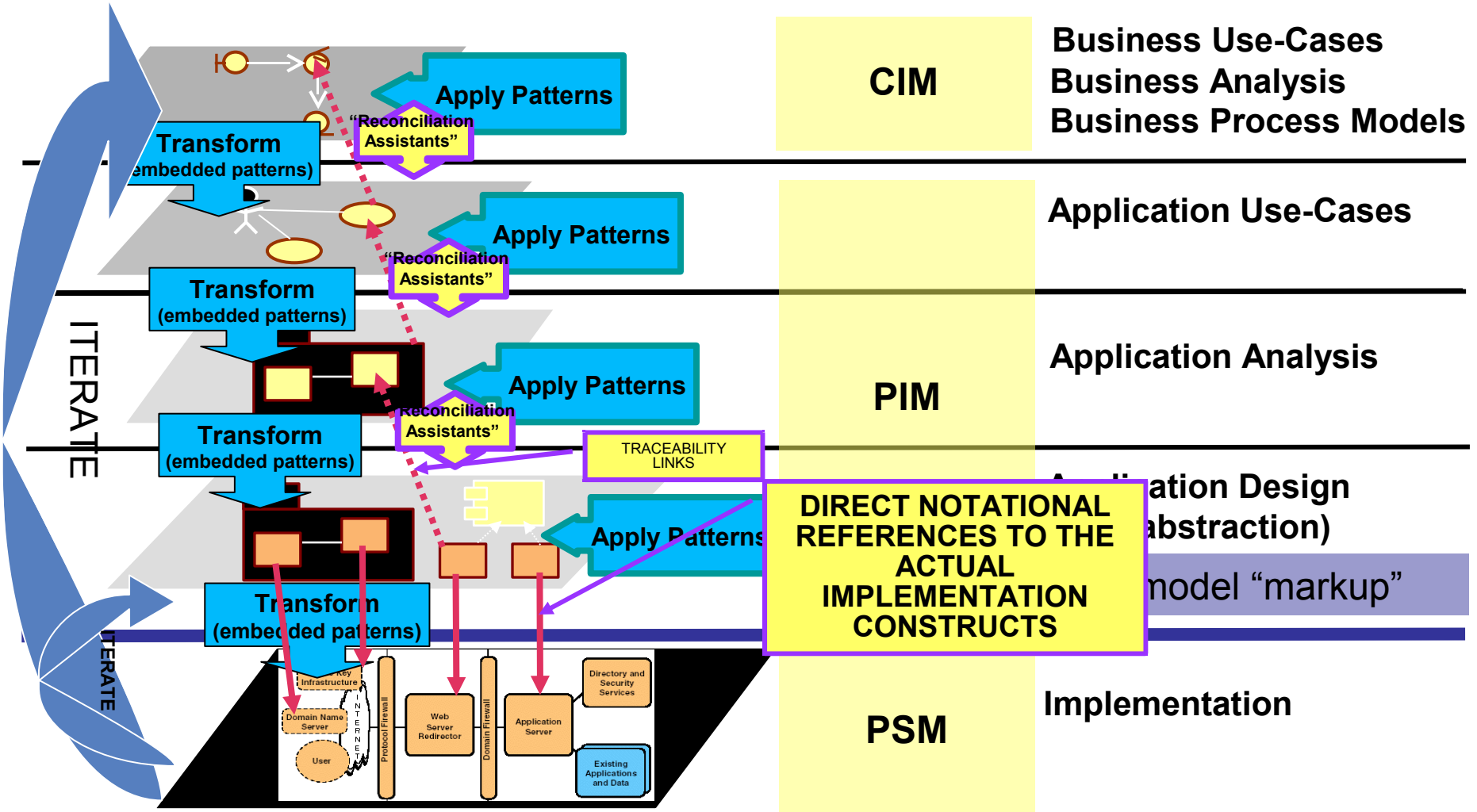
MDD “Theories of Operations” Map To Governance Philosophies



“Mixed Modeling”



“Mixed Modeling”



Code modeling (e.g. RAD “UML Visual Editor”, VS2005 “Class Designer”)



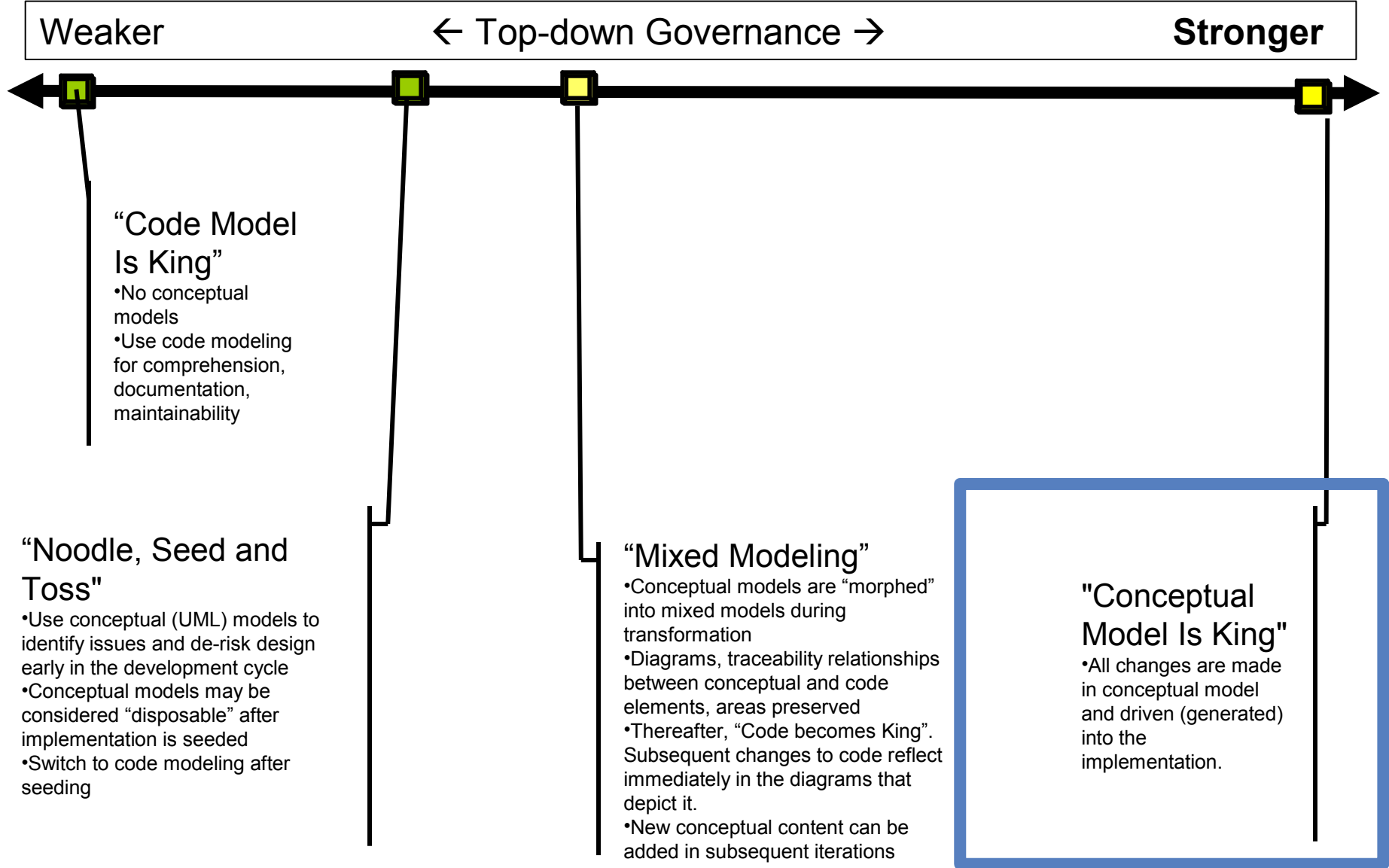


Demo: “Mixed Modeling”

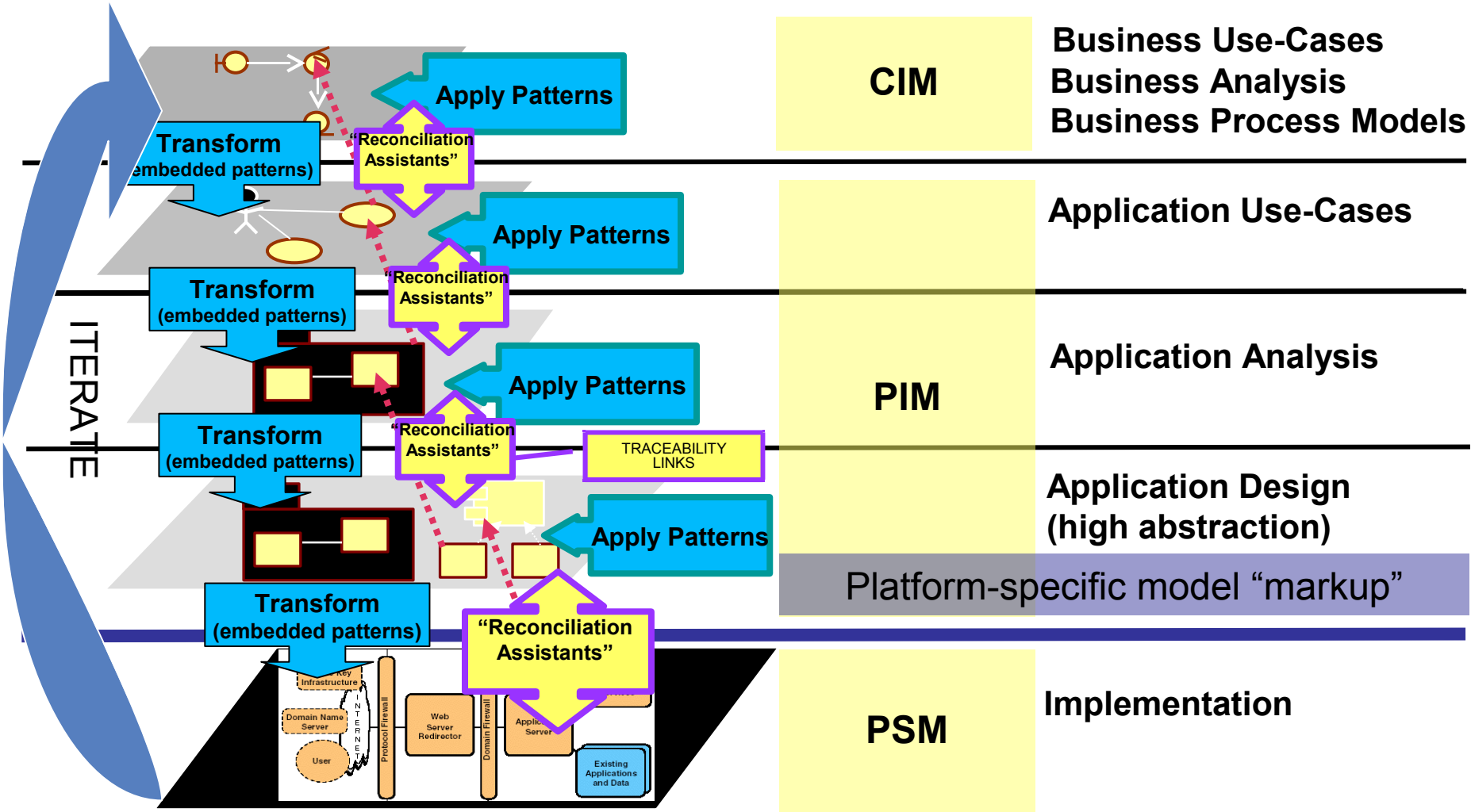
“Mixed Modeling” Value Proposition

- “Moderate” Governance:
 - “We can iteratively create new designs in UML and convert them into implementations in a specific domain.”
 - “Our abstract views of use-cases, activity flows, state machines, class and component models can be depicted directly alongside direct graphical representations of implementation-level constructs
 - “The diagrams that show implementation-level concerns can never get out of sync with the implementation artifacts themselves”
 - Developers don’t have to learn UML semantics, just notations
 - But...
 - “As the implementation evolves, we have no baseline of the original intent at that level of abstraction against which to compare the current state, *unless* we retain a copy of the original source design model and do an unassisted visual comparison of that to the implementation-level diagrams.”
- *Low/No Cost*
 - “We used RTE in Rose/XDE and liked it but it was really difficult to use in a team environment because people kept changing the model and code simultaneously. This works far better.”

MDD “Theories of Operations” Map To Governance Philosophies



“Conceptual Model Is King”



Code modeling (e.g. RAD “UML Visual Editor”, VS2005 “Class Designer”)



“Conceptual Model Is King” Value Proposition

- *Maximal Governance:*
 - “Architects have complete control over how the design contract is implemented”
- *Potential For High Automation:*
 - This theory of operations is appropriate when you expect to develop very high-value automations (transformations) that generate a very high percentage of behavioral code
 - Developing high-value automations can be costly
 - Realizing ROI depends upon potential for re-using the assets across multiple projects
 - SOAs are great candidates (lots of repetitive ‘plumbing’ code)
 - Other types of “software product lines”

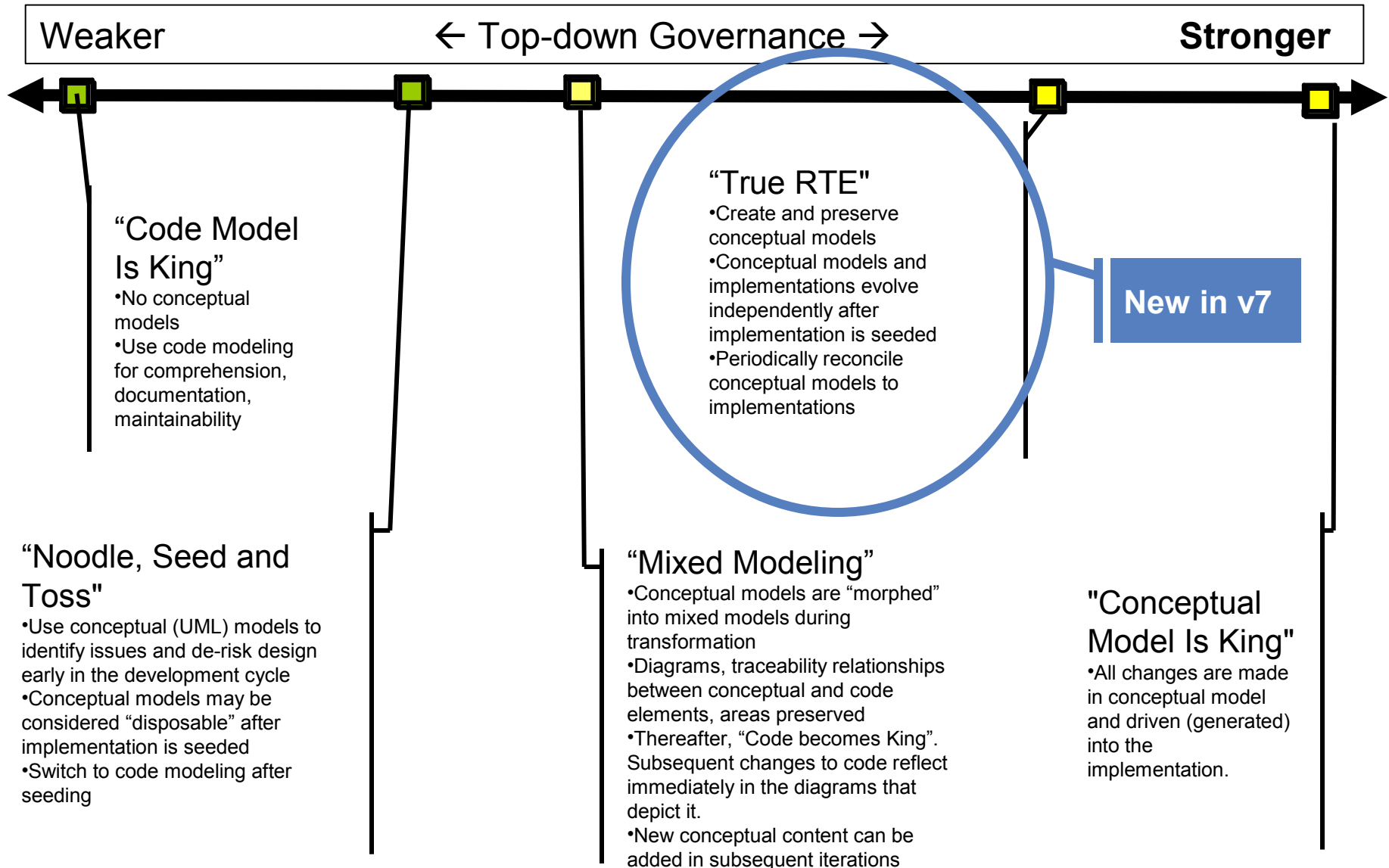
- *The Rational MDD Heritage*
- *The New MDD Product Family: Vision, Overview, and Current State (v6)*
- *The New MDD Product Family: Toward Completing The Vision (v7)*



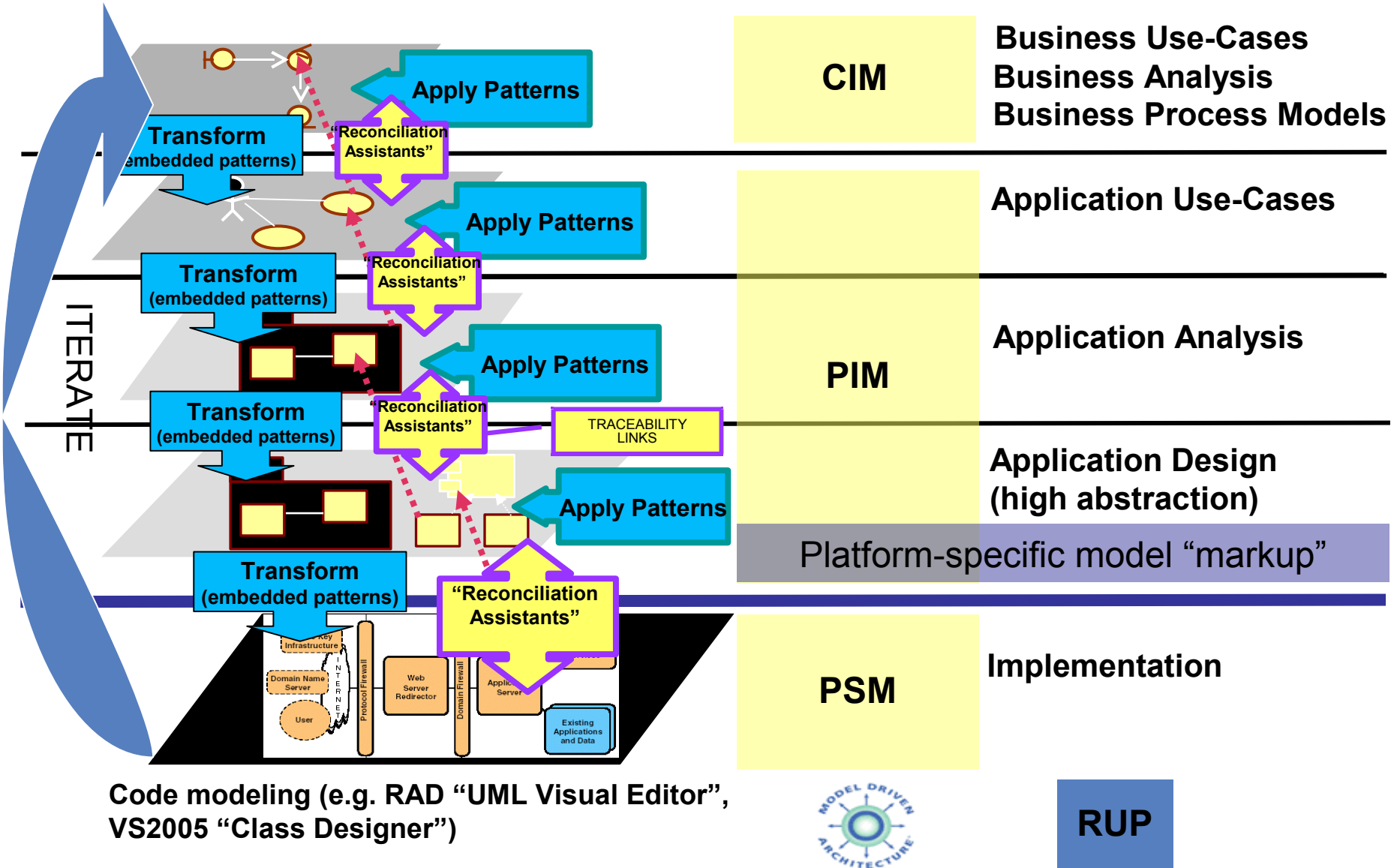
Key Elements of the MDD Vision

- **Business value**
 - Simply supporting UML modeling and RTE is no longer enough
 - MDD must return more repeatability
 - MDD must return more automation of repetitive development tasks
 - MDD must bridge and integrate domains (business domains, activity domains, problem domains, solution and technical domains)
 - MDD must be better integrated with other aspects of the development process and the tools that support them
- **Flexibility**
 - Support multiple MDD “theories of operations”, corresponding to multiple development governance philosophies
- **Extensibility**

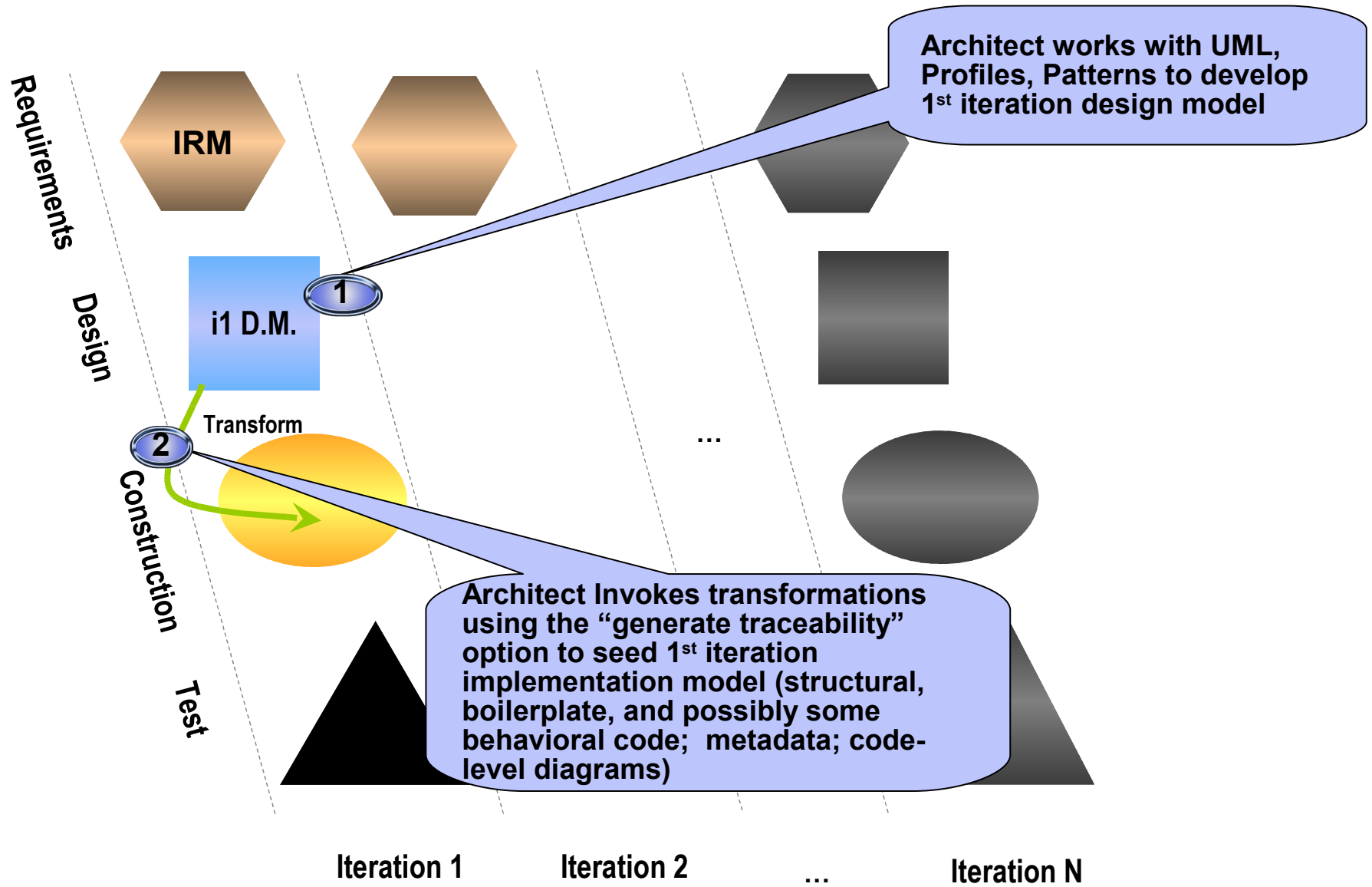
MDD “Theories of Operations” Map To Governance Philosophies



"True RTE"



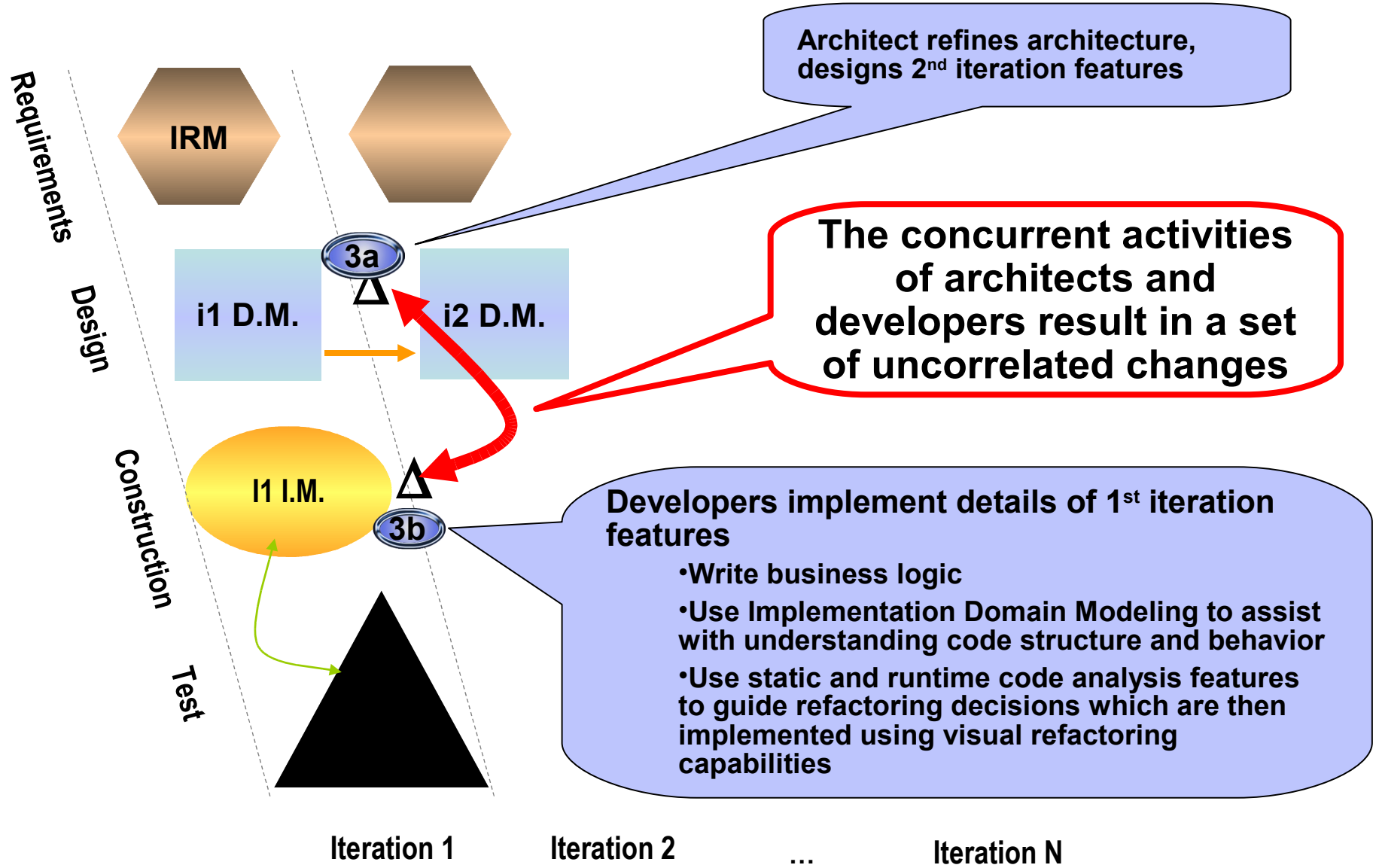
"True RTE" (1,2)



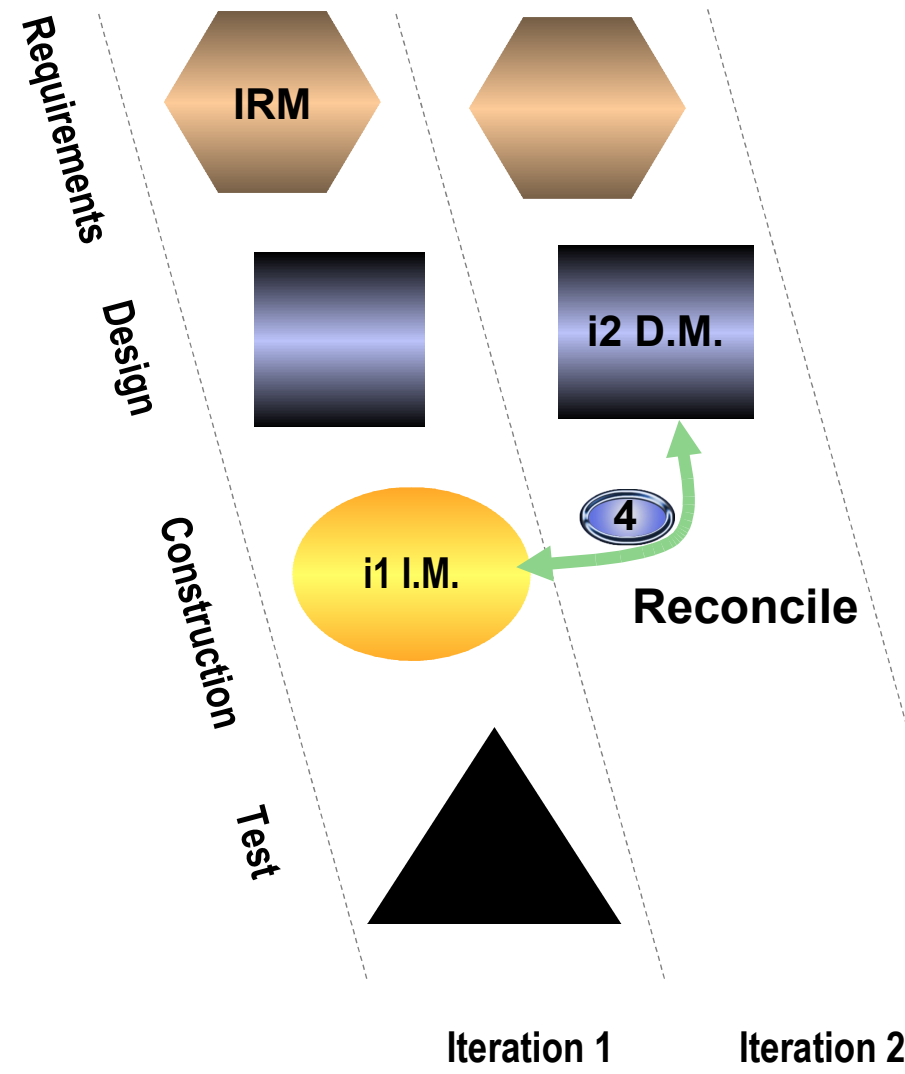
Architect works with UML, Profiles, Patterns to develop 1st iteration design model

Architect Invokes transformations using the "generate traceability" option to seed 1st iteration implementation model (structural, boilerplate, and possibly some behavioral code; metadata; code-level diagrams)

"True RTE" (3)



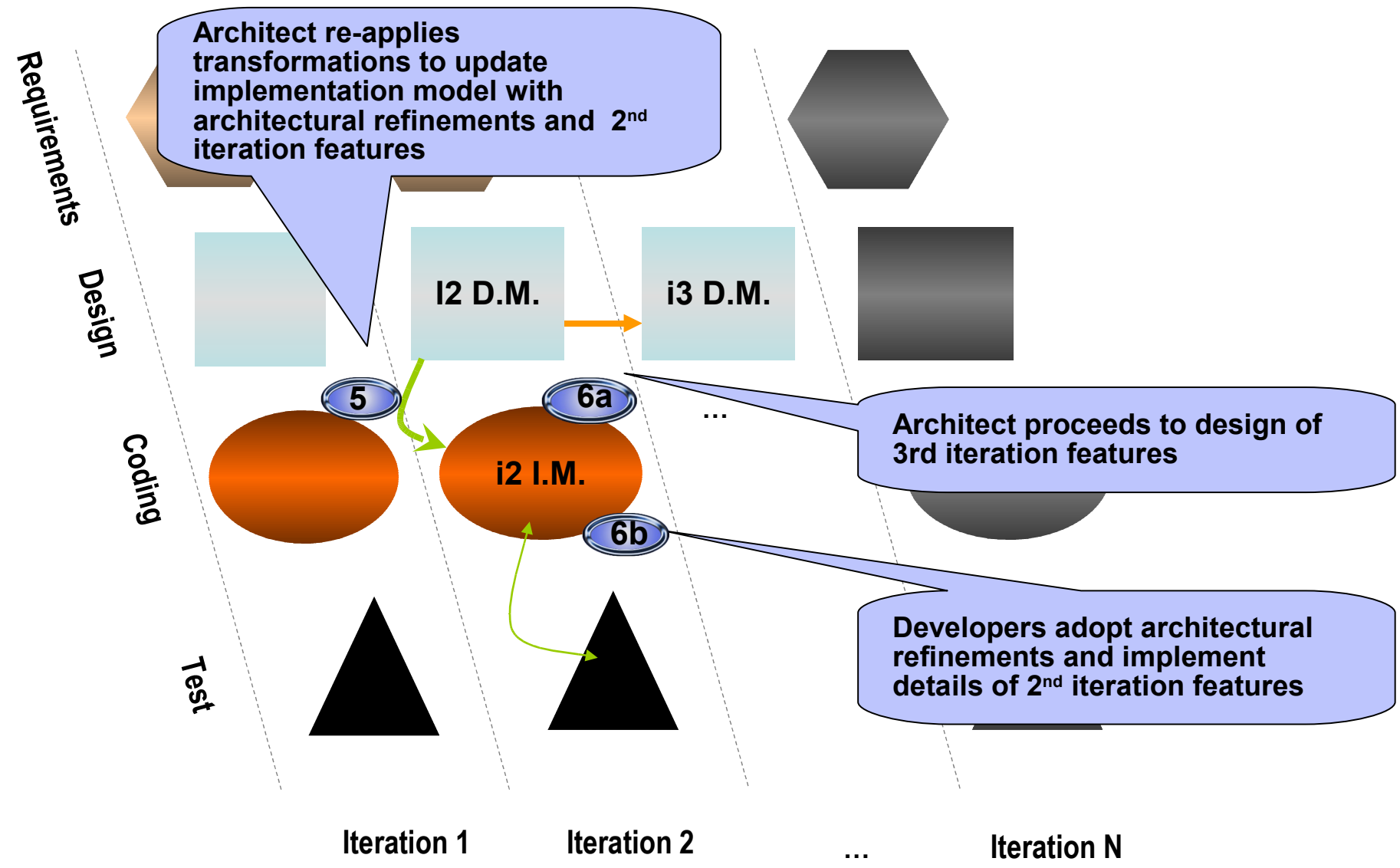
"True RTE" (4)



Reconciliation Tooling

- Inverse transformation applied to implementation creates in-memory model at design level of abstraction
- Architect performs compare-merge of in-memory model and current-state persisted design model, identifying and reconciling ...
 - Implementation constructs not traceable to design model (i.e. new types, methods, ... introduced by the developers):
 - harvest into design if desirable
 - Ignore if not 'architecturally significant'
 - Constructs traceable to model but no longer in agreement with design:
 - update design if desirable
- Re-apply forward transformation...
 - "Harvested" implementation elements thus preserved
 - Undesirable implementation changes thus overwritten

"True RTE" (5,6)





Demo: “True RTE”

“True RTE” Value Proposition

- “Flexible” Governance:
 - “We can iteratively create new designs in UML and convert them into implementations in a specific domain.”
 - “Our abstract views of use-cases, activity flows, state machines, class and component models can be depicted directly alongside direct graphical representations of implementation-level constructs
 - “The diagrams that show implementation-level concerns can never get out of sync with the implementation artifacts themselves”
 - Developers don’t have to learn UML semantics, just notations
 - But...
 - “As the implementation evolves, we have no baseline of the original intent at that level of abstraction against which to compare the current state, *unless* we retain a copy of the original source design model and do an assisted visual comparison of that to the implementation-level diagrams.”
- ~~Low/No Cost~~
 - “We used RTE in Rose/XDE and liked it but it was really difficult to use in a team environment because people kept changing the model and code simultaneously. This works far better.”

Key Elements of the MDD Vision

- **Business value**
 - Simply supporting UML modeling and RTE is no longer enough
 - MDD must return more repeatability
 - MDD must return more automation of repetitive development tasks
 - MDD must bridge and integrate domains (business domains, activity domains, problem domains, solution and technical domains)
 - MDD must be better integrated with other aspects of the development process and the tools that support them
- **Flexibility**
 - Support multiple MDD “theories of operations”, corresponding to multiple development governance philosophies
- **Extensibility**

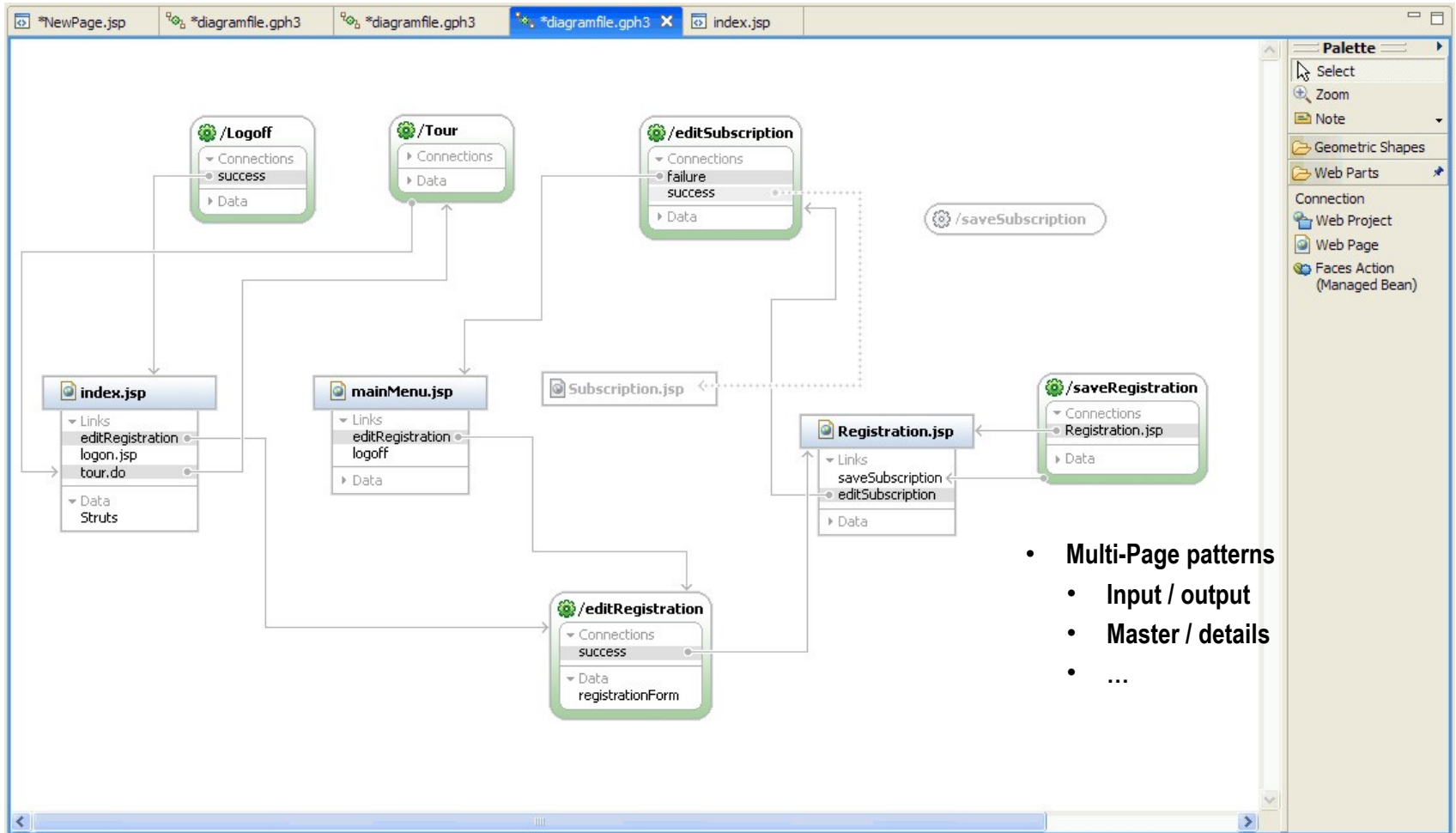
- *Java 5 support (annotations, generics, enums, static import, varargs...)*
- *Operation signature preference (UMI)*
- *Ability to depict classes in ext*
- *Java developer tools available from diag.*
- *Inline editing of java fields and methods (with optional invocation of refactoring)*
- *Many additional properties added to properties view*
 - *Properties view tabs for s of packages, classes, interfaces, fields & methods*
- *Option to create field's type using import statement instead of fully qualified name*
- *Improved collection support*

Screen shot(s), prune and format enhancements...

Web Modeling in v7: Overhauled Web Diagram Editor (WDE)

Direct-edit mode

Visualize Data and Service consumption



- **Multi-Page patterns**
 - **Input / output**
 - **Master / details**
 - ...

Usage goal: create a complete data/service driven application completely within WDE

UML Modeling in v7 (1 of 2)

- Adopt final UML2 spec (2.1)
- Search enhancements (leveraging new indexing work)
- Component diagrams: better stereotypes
- Deployment diagrams: Better instance modeling & stereotypes
- Sequence diagrams: lifeline collapse
- Object diagrams: **NEW!**



Properties Tasks Console Bookmarks

General
Stereotypes
Documentation
Constraints
Slots and Values
Appearance

<Instance Specification> Blank Model::componentinstance2

List Inherited Properties

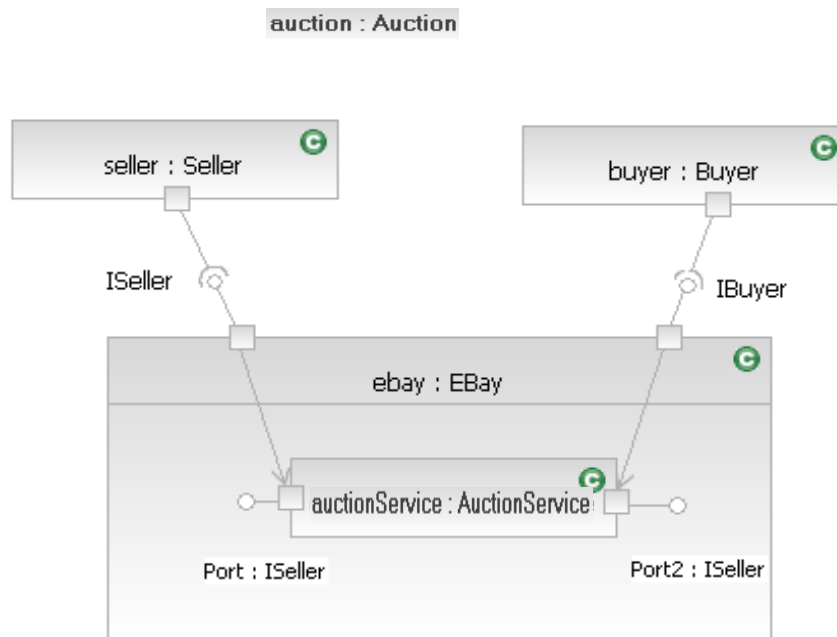
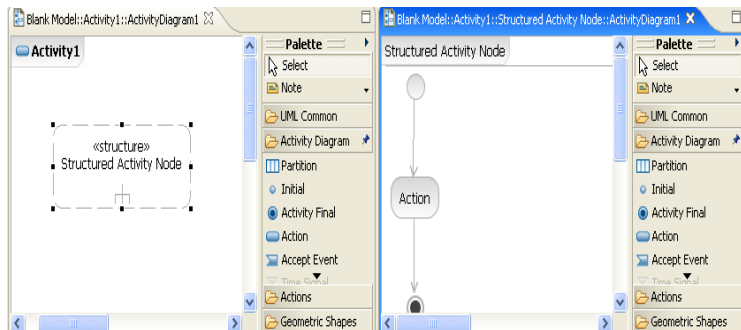
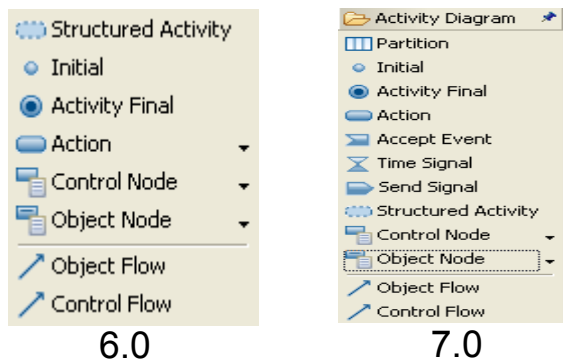
Name	Type	Value
<input checked="" type="checkbox"/> Blank Model::Component::manufacturer	ICompanyX	
<input checked="" type="checkbox"/> Blank Model::Component::partNo	int	

Activity Diagrams

- Additional element (action) types (~ intermediate level support)
- Sub-diagram support on structured activity nodes
- Partition shown on activity
- Multi-select
- Operation shown on call node

Structure Diagrams

- Part shape gives feedback for DnD operations (create port/type part); many more DnD operations
- Port property sheet changes
- Parts have a Shape compartment (show inside)
- Improved label layout
- Ball socket notation



Key Elements of the MDD Vision

- **Business value**
 - Simply supporting UML modeling and RTE is no longer enough
 - MDD must return more repeatability
 - MDD must return more automation of repetitive development tasks
 - MDD must bridge and integrate domains (business domains, activity domains, problem domains, solution and technical domains)
 - **MDD must be better integrated with other aspects of the development process and the tools that support them**
- **Flexibility**
 - Support multiple MDD “theories of operations”, corresponding to multiple development governance philosophies
- **Extensibility**

- **Sub-units** (*independence of physical structure from logical structure*)
- *Diff-merge enhancements*

**Screen shots – consult w/ Kim re:
what would be effective**

Key Elements of the MDD Vision

- **Business value**
 - Simply supporting UML modeling and RTE is no longer enough
 - MDD must return more repeatability
 - MDD must return more automation of repetitive development tasks
 - **MDD must bridge and integrate domains (business domains, activity domains, problem domains, solution and technical domains)**
 - MDD must be better integrated with other aspects of the development process and the tools that support them
- **Flexibility**
 - Support multiple MDD “theories of operations”, corresponding to multiple development governance philosophies
- **Extensibility**

Further Unification of Conceptual and Code Modeling

- Better Integration of
 - Search
 - “Type”
 - “Show related elements”
- *Support use of code model elements in conceptual UML Structure diagrams*
- *Common Modeling Infrastructure Improvements*
 - Common explorer
 - Pan tool
 - Paste into diagrams from external apps
 - Improved layout algorithms
 - Better diagram work area management

Key Elements of the MDD Vision

- **Business value**
 - Simply supporting UML modeling and RTE is no longer enough
 - MDD must return more repeatability
 - **MDD must return more automation of repetitive development tasks**
 - **MDD must bridge and integrate domains (business domains, activity domains, problem domains, solution and technical domains)**
 - MDD must be better integrated with other aspects of the development process and the tools that support them
- **Flexibility**
 - Support multiple MDD “theories of operations”, corresponding to multiple development governance philosophies
- **Extensibility**

Packaged Transformations in v7

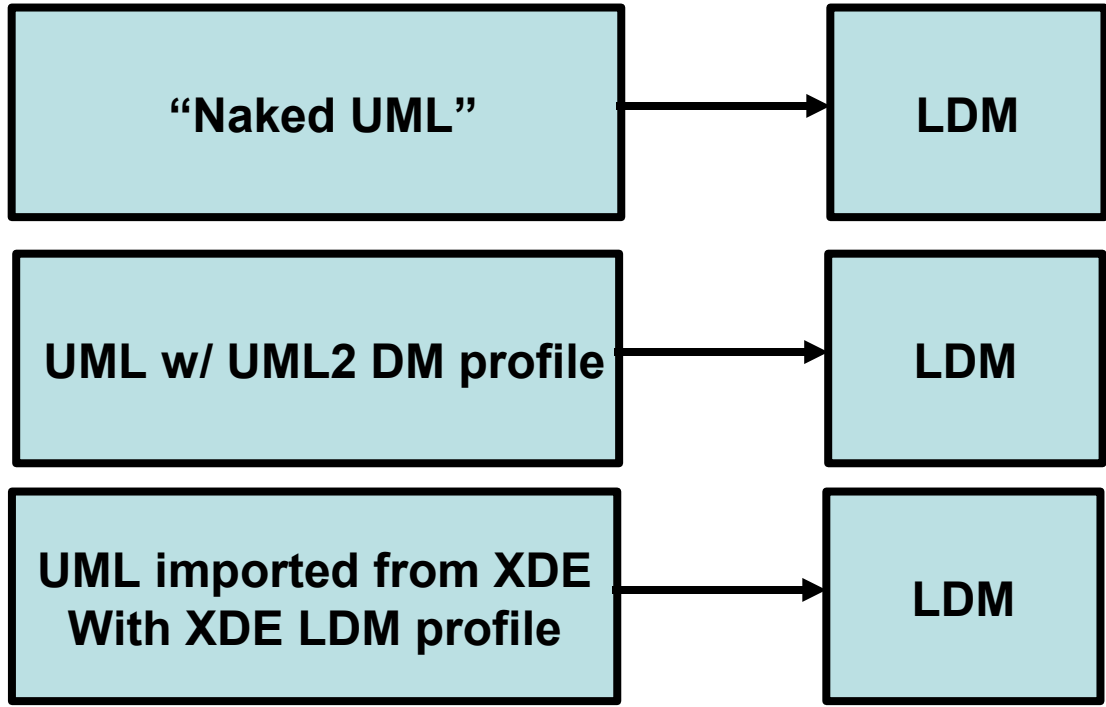
- *WSDL improved and productized*
- *XSD improved and productized*
- *Java updated (Java 5)*
- *Java inverse / reconciliation*
- *C++ updated (fuller support, non-destructive re-apply, inverse / reconciliation)*
- *EGL*

- *Later...*



- *C# forward / inverse / reconciliation*
- *Gen from state machines*
- *BPEL, further improvements to WSDL and XSD*

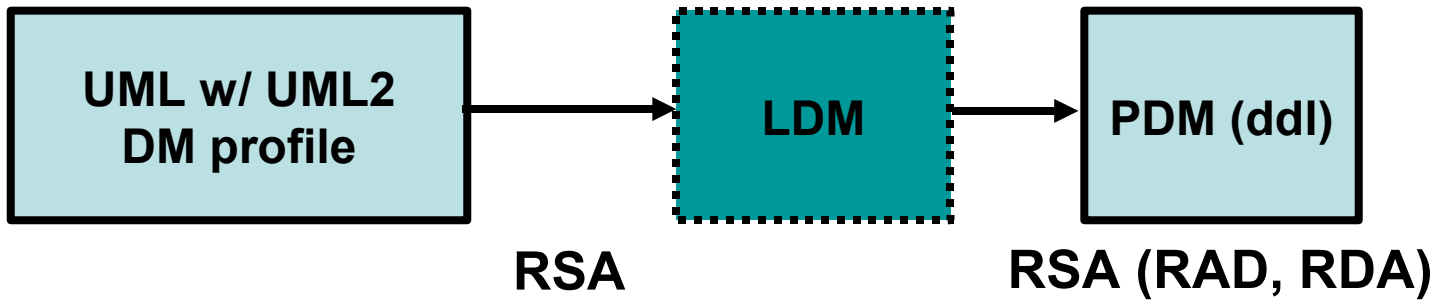
Data-Oriented Application Architects



RSM, RSA

RDA

- Transformations
- Shell-sharing by RSx and RDA

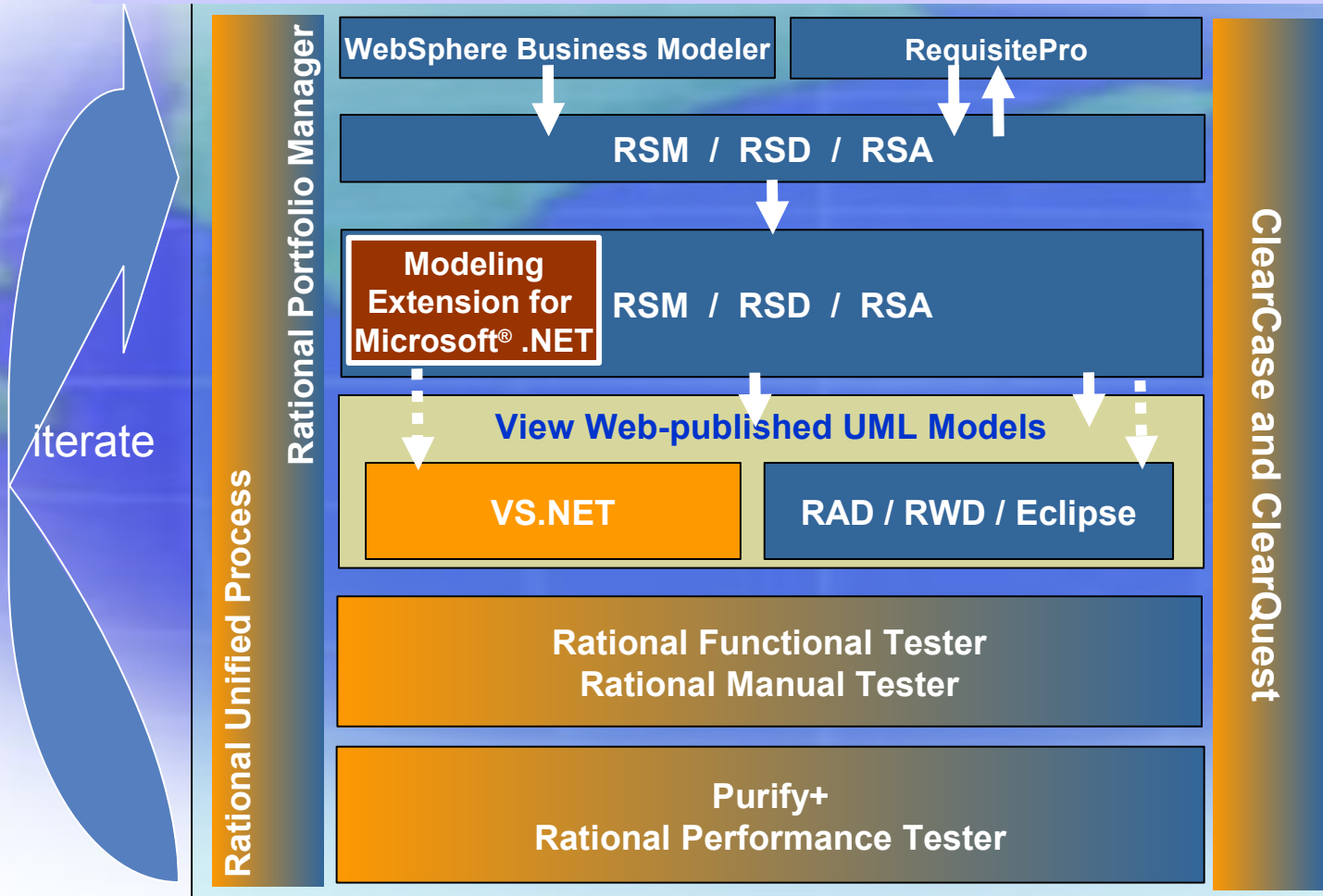


RSA

RSA (RAD, RDA)

Heterogeneous Development With the IBM Rational SDP

Learn, deploy, and maintain a complete IT governance solution consisting of an open standards-based set of integrated lifecycle tools, with consistent workflows and single-source product support, enabling development that can target a mix of platforms including Enterprise Java and .NET



“Code Model Is King”: Rational Application Developer

The screenshot displays the Rational Application Developer interface for a Java project. The top-left pane shows the Model Explorer with a tree view of the project structure, including packages like Design and classes like Class1, Class2, and Class3. The main workspace shows a UML class diagram with three classes (Class1, Class2, Class3) and an interface (Interface1). Class1 is associated with Class2, and Class2 is associated with Class3. A context menu is open over Class3, showing options like Add Java, Add Note, Visualize, Navigate, File, Edit, Delete from Diagram, Delete from Project, Refactor, Find/Replace, Format, Filters, Transform, Harvest, Show Properties View, and Properties. The bottom-left pane shows the Outline Explorer with a list of classes and interfaces. The bottom-right pane shows the Properties view with various appearance and advanced settings. A Java palette is visible on the right side of the workspace, listing various UML elements like Package, Class, Interface, Extends, Implements, and Association.

Java project

Direct depictions of code, UML-like notation

Java palette

Diagrams stored in .dnx files

Java context menus

“Code Model Is King”: Visual Studio

The screenshot shows the Visual Studio IDE with a C# project named 'MyWinApp'. The main window displays a UML class diagram with the following elements:

- Program**: A dashed box representing a static class.
- Form**: A class that inherits from **ContainerControl**.
- Form1**: A class that inherits from **Form** and has an association with **Program**.

Callouts in the image highlight the following features:

- C# palette**: Points to the Class Designer toolbox on the left, which contains various C# constructs like Pointer, Class, Enum, Interface, etc.
- C# project**: Points to the Solution Explorer on the right, showing the project structure.
- Diagrams stored in .cd files**: Points to the 'ClassDiagram1.cd' file in the Solution Explorer.
- Direct depictions of code, UML-like notation**: Points to the UML class diagram in the center.

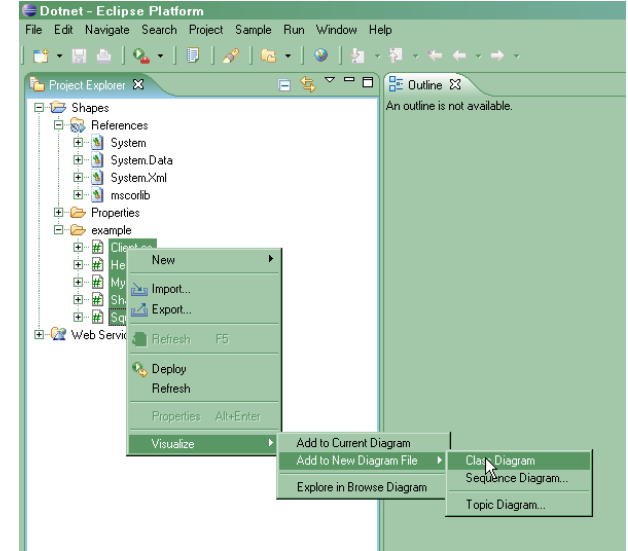
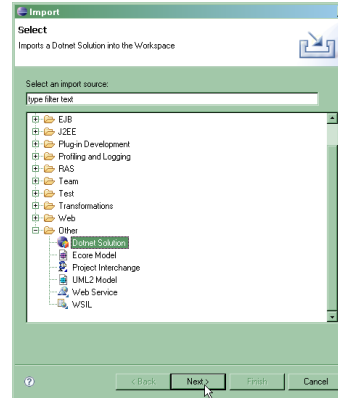
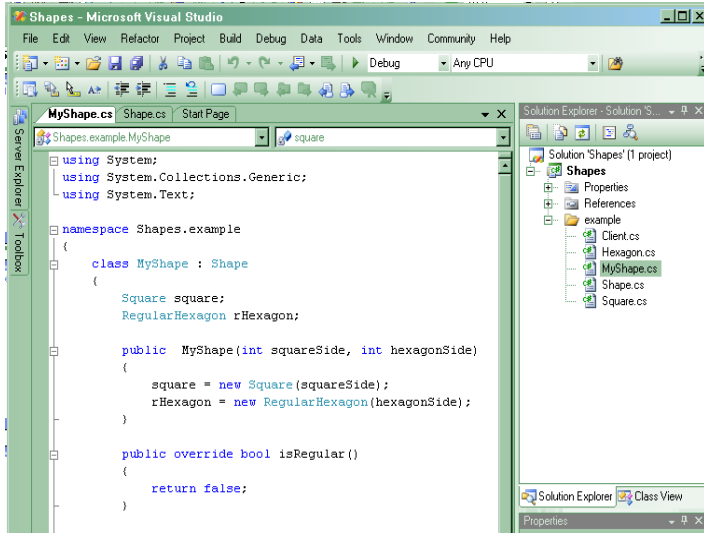
The bottom of the screenshot shows the Error List and Class Details panels.

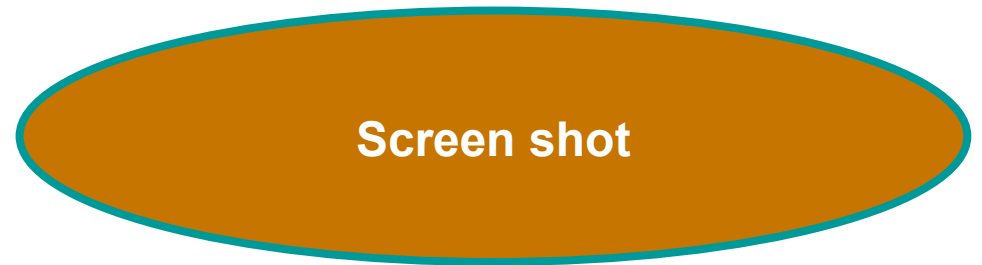
Model Driven Development for Microsoft® .NET

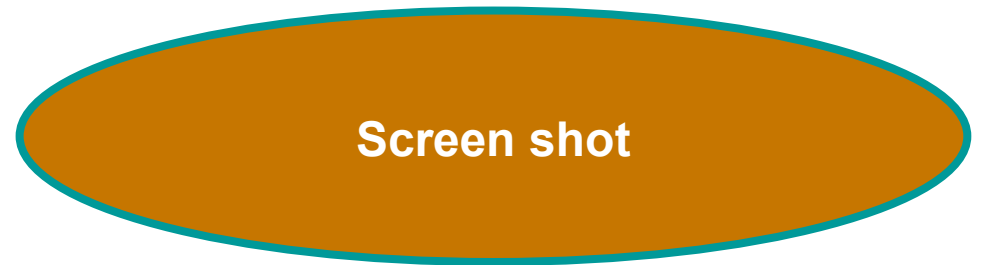
- UML to C# transformation (supports “Seed code...” and “Conceptual Model Is King”)
- C# to UML inverse transformation, reconcile (supports “True RTE”)
- .NET CTS type visualization
 - Read-only visualization based on assemblies (binaries)
 - Use as modeling ‘library’ to resolve type references
- C# source visualization
 - Initially read-only
 - Supports “Mixed Modeling”
- Migration of XDE code models
 - Initially C#
 - Import code model as normal
 - Subsequent pass converts code model UML elements into references to CTS types and C# sources (ready for “Mixed Modeling”)
- Support for VB, managed C++ remains TBD at this point



Importing and Visualizing a Solution









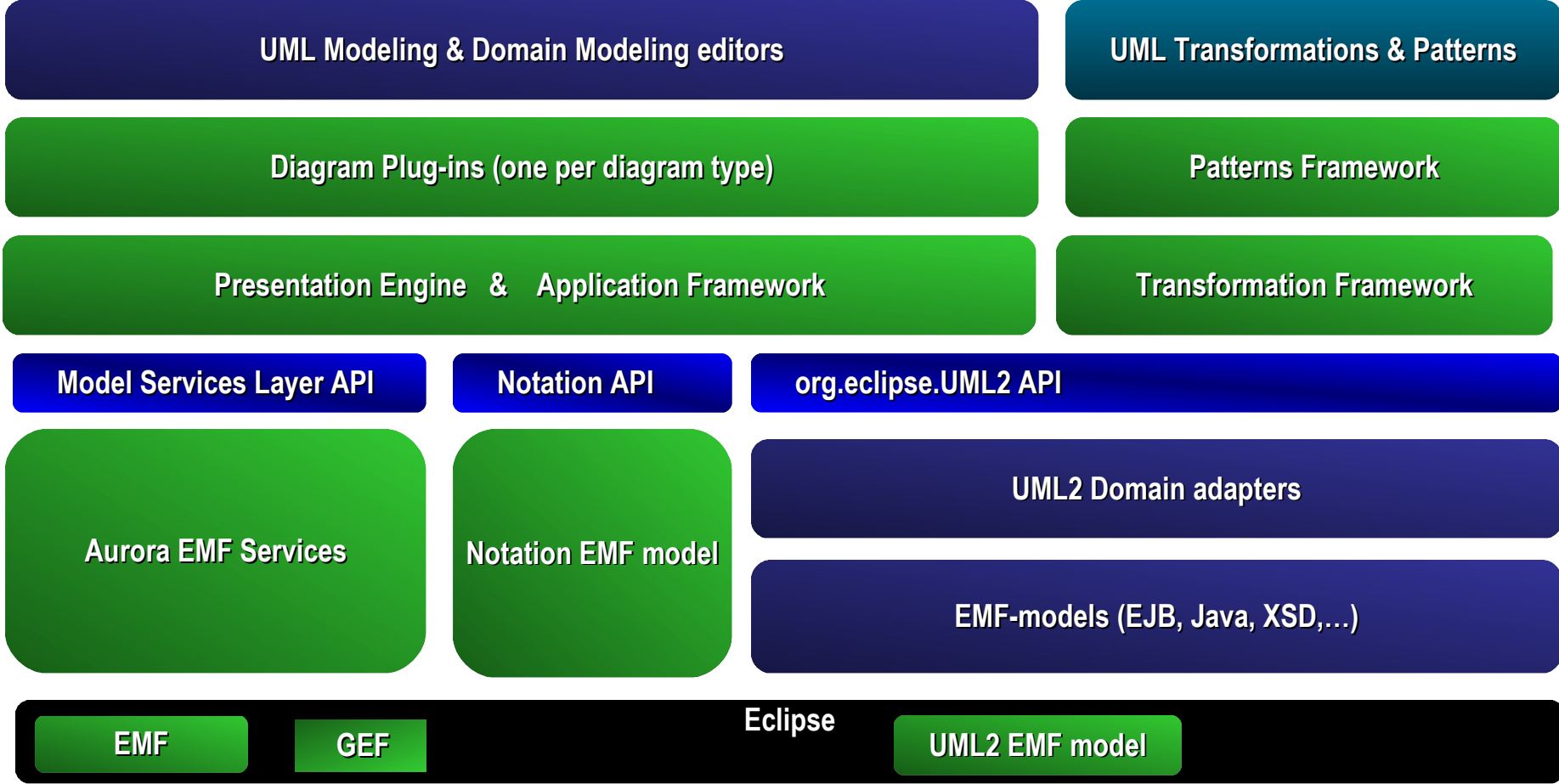
Demo:

MDD for .NET

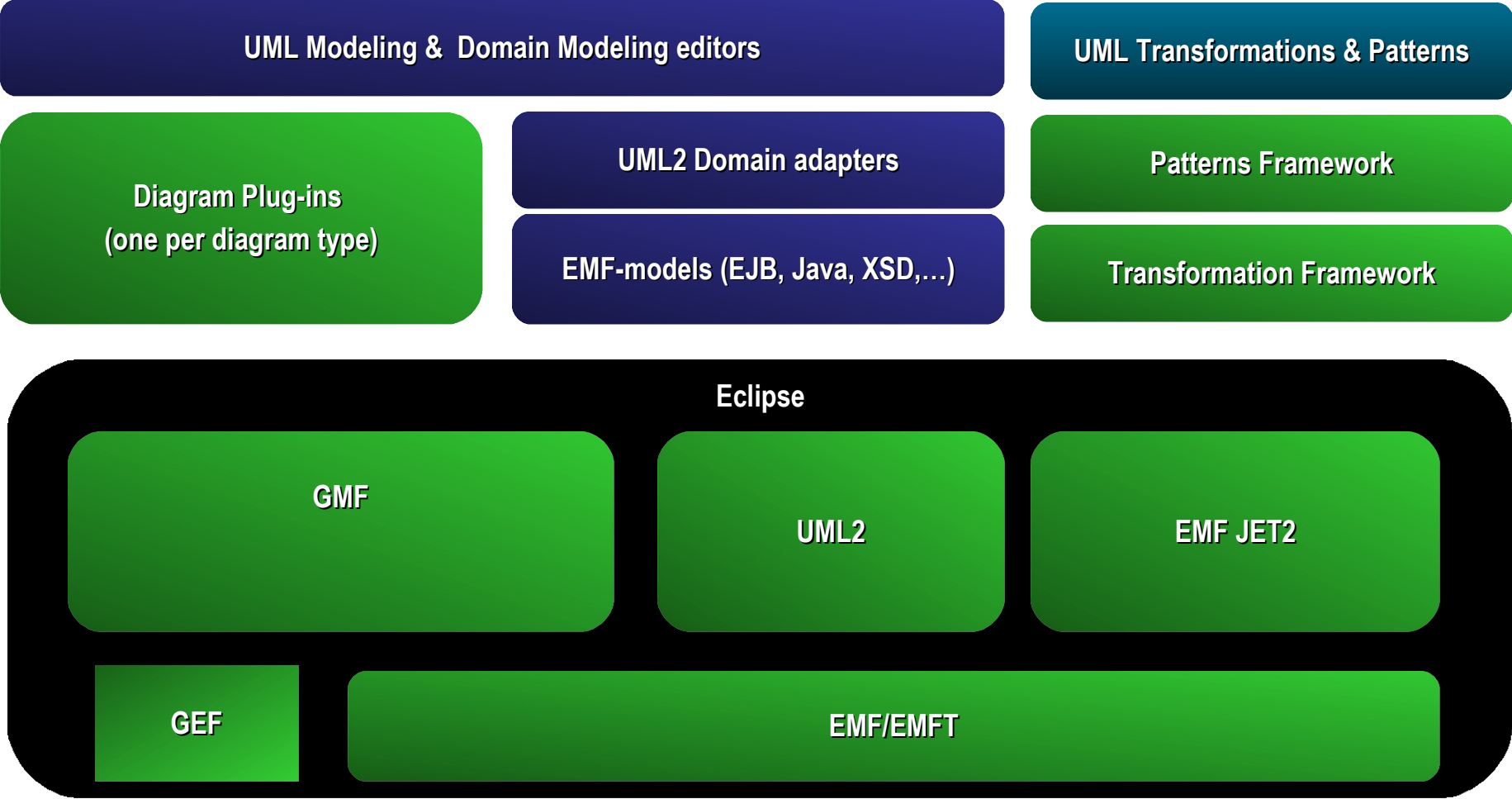
Key Elements of the MDD Vision

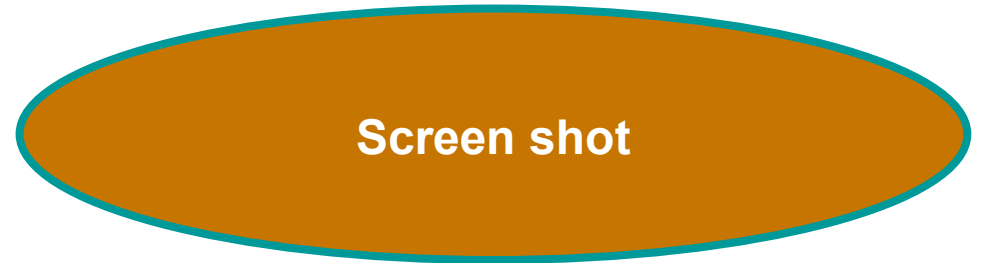
- **Business value**
 - Simply supporting UML modeling and RTE is no longer enough
 - MDD must return more repeatability
 - MDD must return more automation of repetitive development tasks
 - MDD must bridge and integrate domains (business domains, activity domains, problem domains, solution and technical domains)
 - MDD must be better integrated with other aspects of the development process and the tools that support them
- **Flexibility**
 - Support multiple MDD “theories of operations”, corresponding to multiple development governance philosophies
- **Extensibility**

High-level 6.0 architecture

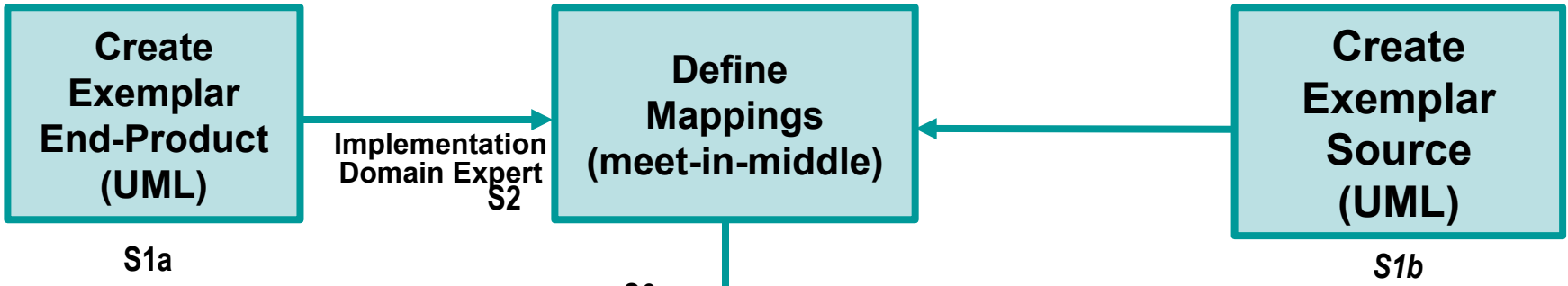


High-level 7.0 architecture







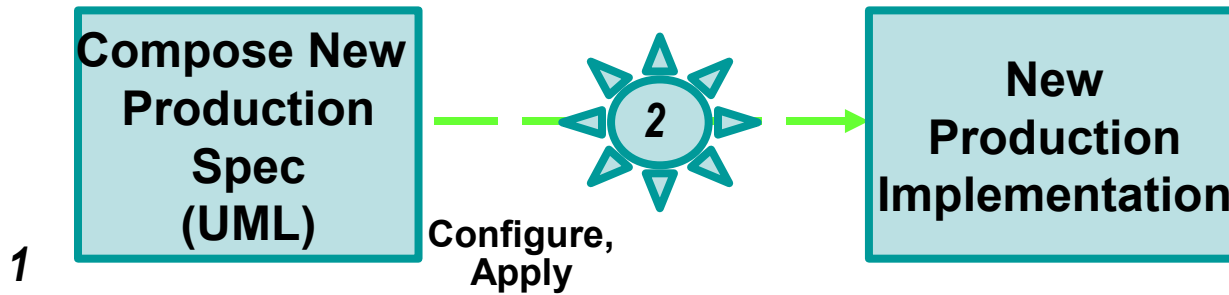


Who ←----- Implementation Domain Expert -----> ←----- Specification Domain Expert ----->

Mappings *Sophisticated (business value)*

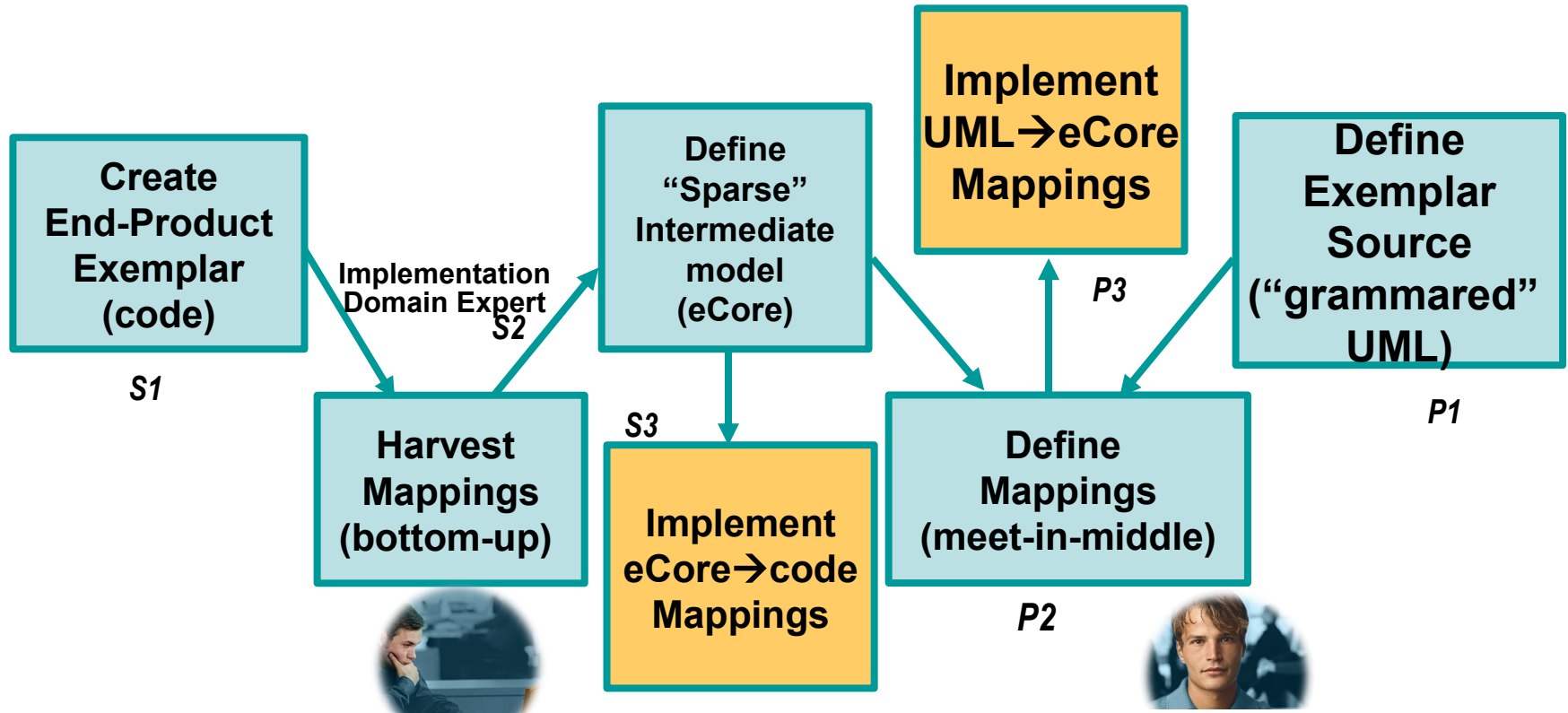
Programming Models **Graphical Mapping Tools, Java**

Domains UML UML UML

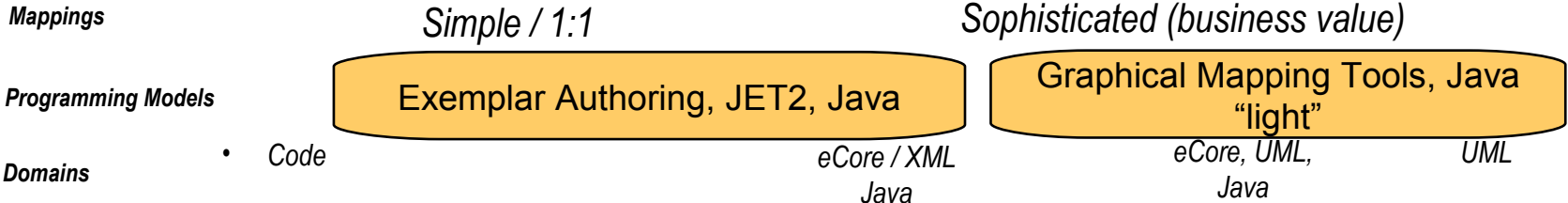


Specification Domain Expert

Author UML "Grammared" model → UML "Intermediate" model → code



Who ←----- Implementation Domain Expert -----> Spec. Domain Expert ----->



Apply UML "Grammared" model → UML "Intermediate" model → code



1



Problem Domain Expert

Transparent

Agenda



- *The Rational MDD Heritage*
- *The New MDD Product Family: Vision, Overview, and Current State (v6)*
- *The New MDD Product Family: Toward Completing The Vision (v7)*
- *Listening To Our Customers*



Key Elements of the MDD Vision

- **Business value**
 - Simply supporting UML modeling and RTE is no longer enough
 - MDD must return more repeatability
 - MDD must return more automation of repetitive development tasks
 - MDD must bridge and integrate domains (business domains, activity domains, problem domains, solution and technical domains)
 - MDD must be better integrated with other aspects of the development process and the tools that support them
 - **Flexibility**
 - Support multiple MDD “theories of operations”, corresponding to multiple development governance philosophies
 - **Extensibility**
- **Total Cost of Ownership**

We Are Listening



“Improve performance”

“Automate product deployment”



“We want license enforcement”

“Work in my existing Eclipse environment”

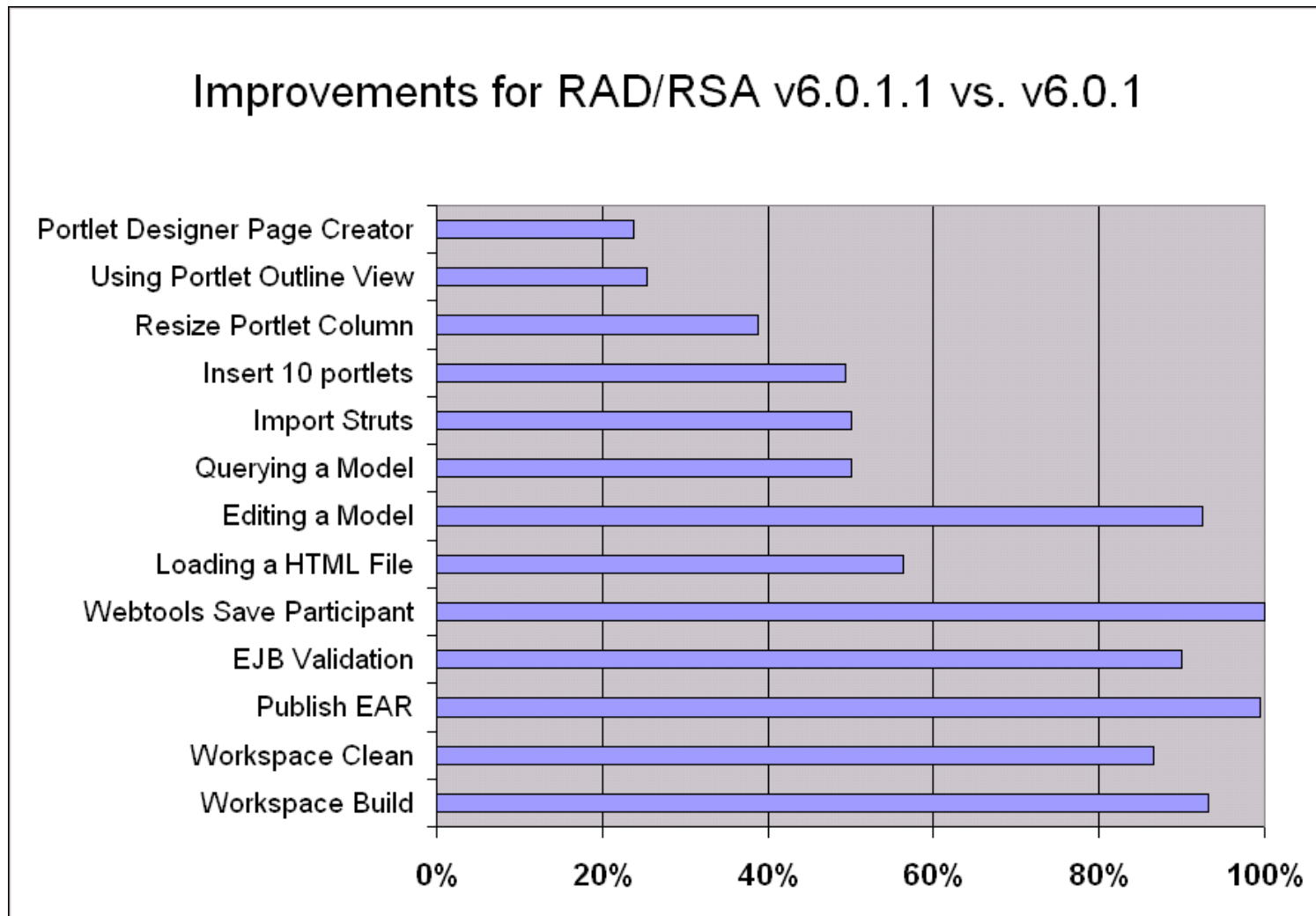


“Needs too much RAM”

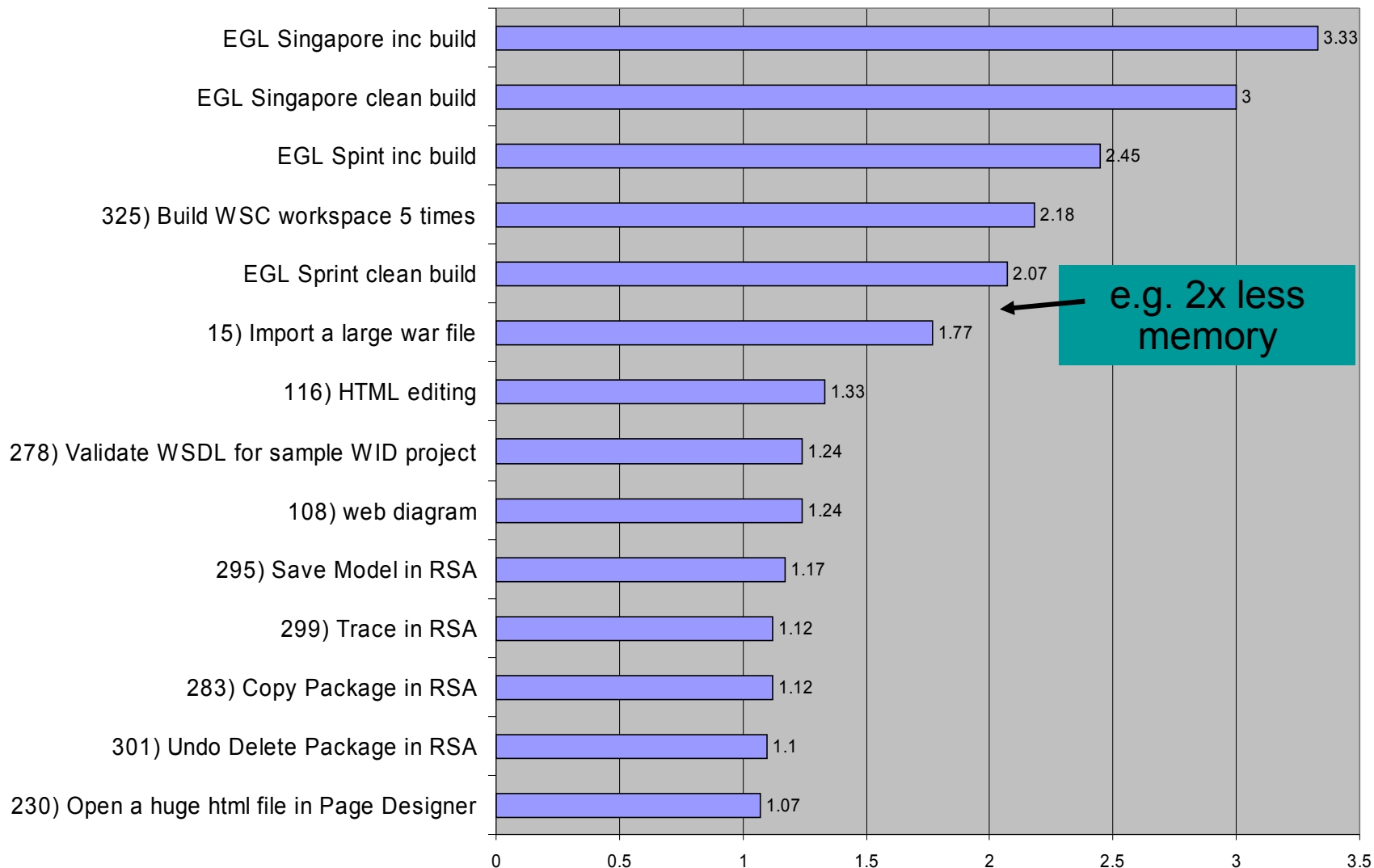
“Smaller on-disk footprint”

“I only want to buy or install what I want to use – don’t make me take more”

Performance Results in V6.0.1.1 (Speed)



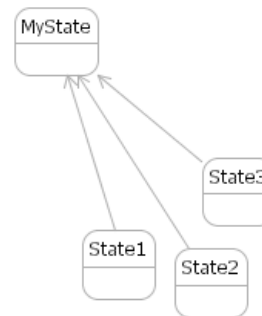
Performance Improvements in V6.0.1.1 (Space)



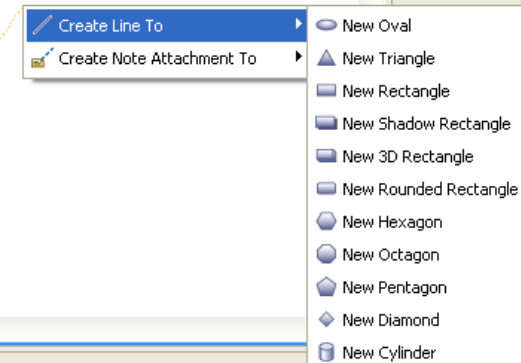
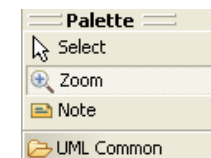
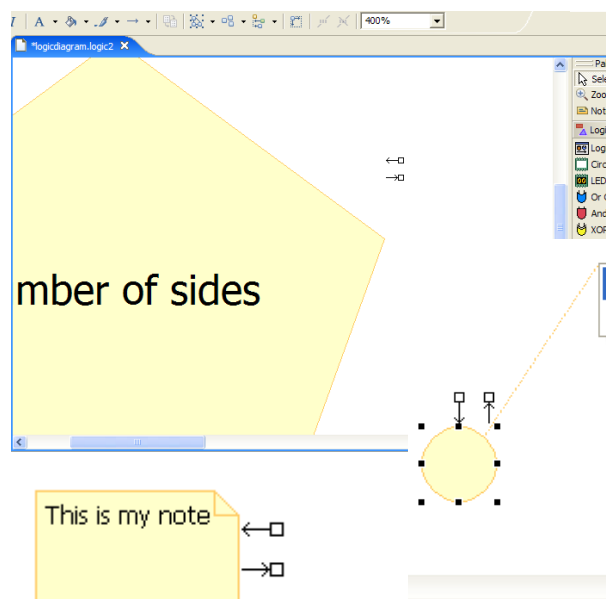
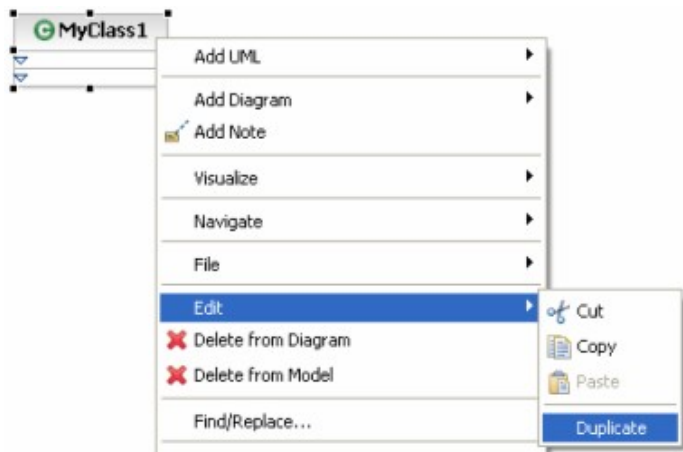
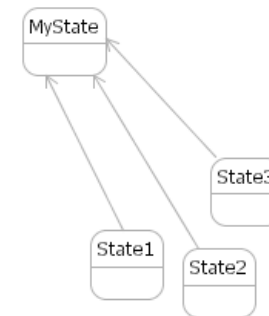
General Modeling Usability

- Improved relationship anchor support
- “Change Metatype” refactoring action
- Zoom tool, animated zoom, animated arrange
- “Duplicate element” action when drawing
- Connector assistants for Notes and Geo Shapes

6.0



7.0



Product Installation and Deployment (1)

The screenshot shows the 'Features' tab of the IBM Installation Manager. The main area is a tree view of features with checkboxes and a 'Size' column. A 'Dependencies' pane is on the right. A 'Disk Space Information' table is at the bottom left. Callouts highlight specific features and their management options.

Features	Size
<input checked="" type="checkbox"/> IBM Rational Software Architect	82.1 / 983.4 MB
<input checked="" type="checkbox"/> Modeling	19.6 / 19.6 MB
<input checked="" type="checkbox"/> UML Modeling	
<input checked="" type="checkbox"/> UML to UML transformations and patterns	
<input checked="" type="checkbox"/> C++ modeling and transformations	
<input checked="" type="checkbox"/> Java modeling and transformations	
<input checked="" type="checkbox"/> EJB modeling and transformations	
<input checked="" type="checkbox"/> WSDL/XSD modeling and transformations	
<input checked="" type="checkbox"/> DoDAF	
<input checked="" type="checkbox"/> UML to CORBA transformations	
<input type="checkbox"/> Modeling integrations	
<input type="checkbox"/> WebSphere Business Integration	
<input type="checkbox"/> Rational Rose model import	
<input type="checkbox"/> Rational XDE model import	
<input type="checkbox"/> Rational SoDa integration	
<input type="checkbox"/> C/C++ Development Tools (CDT)	
<input checked="" type="checkbox"/> J2EE and Web Services Development Tools	0.0 / 0.0 B
<input type="checkbox"/> Java client application editor	0.0 / 0.0 B
<input type="checkbox"/> Web development tools	19.6 / 19.6 MB
<input checked="" type="checkbox"/> Web development tools	
<input checked="" type="checkbox"/> Struts Tools	
<input checked="" type="checkbox"/> Crystal Reports tools	
<input type="checkbox"/> Portal tools	0.0 / 170.3 MB
<input checked="" type="checkbox"/> J2EE Connector (J2C) tools	23.1 / 23.1 MB
<input checked="" type="checkbox"/> Application Analysis/Developer Test	0.0 / 0.0 B
<input type="checkbox"/> Code review	
<input checked="" type="checkbox"/> Test and Performance Tools Platform (TPTP)	
<input checked="" type="checkbox"/> Architectural rules	
<input type="checkbox"/> Team integrations	0.0 / 0.0 B
<input checked="" type="checkbox"/> Rational ClearCase SCM Adapter	
<input checked="" type="checkbox"/> Rational RequisitePro integration	
<input checked="" type="checkbox"/> Rational Unified Process (RUP) Process Advisor and Process Br	
<input checked="" type="checkbox"/> Reusable Asset Specification (RAS) tools	
<input type="checkbox"/> Extensibility features	0.0 / 0.0 B
<input checked="" type="checkbox"/> Transformation authoring	
<input type="checkbox"/> APL Migration	

Volume	Available	Required
C:	39.0 GB	82.1 MB
C:	39.0 GB	82.1 MB

Transformation authoring
Provides tools to create custom transformations and customize existing transformations. Transformations automate the task of generating model content and implementation code.

Features become optionally installable

Configurable Silent Installs

Affords management of footprint

Product Installation and Deployment (2)

The screenshot shows the 'Environment' step of the IBM Installation Manager. The window title is 'IBM Installation Manager' and it has a menu bar with 'File' and 'Help'. Below the title bar, there's a sub-header 'Environment' with the instruction 'Select language packs for the install location and select Eclipse location'. A progress bar shows steps: Install, Licenses, Location, Environment (current), Features, and Summary. Under the 'Languages' section, there's a list of language packs with checkboxes: English (checked), Traditional Chinese, Czech, French, German, Hungarian, Italian, Japanese, Korean, Simplified Chinese, Polish, Portuguese, Russian, and Spanish. Under the 'Eclipse' section, there's a checkbox for 'Use Existing Eclipse' which is checked. Below this are two text boxes: 'Eclipse IDE:' and 'Eclipse IDE JVM:', each with a 'Browse...' button. A callout bubble points to the 'Use Existing Eclipse' checkbox with the text 'Install into existing Eclipse instance'. At the bottom, there are navigation buttons: '< Back', 'Next >', 'Install', and 'Cancel'.

Install into existing Eclipse instance

Product Installation and Deployment (3)

Authoring Agent

File Help

Quick Start Installation Packages

Post-v7

IBM Desktop Management System

What would you like to do?

- IT administrator can pre-configure customized installations for specific user-communities
- Managed / scheduled updates

Create Installation Package

Check for Updates

View Installation Packages

We Are Listening



“Improve performance”

“Automate product deployment”



“We want license enforcement”

“Work in my existing Eclipse environment”

“Needs too much RAM”

“Smaller on-disk footprint”

“I only want to buy or install what I want to use – don’t make me take more”





Questions



Thank You

*Presenter: William T. Smith
smithtw@us.ibm.com*