"Toward Completing A Vision"

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

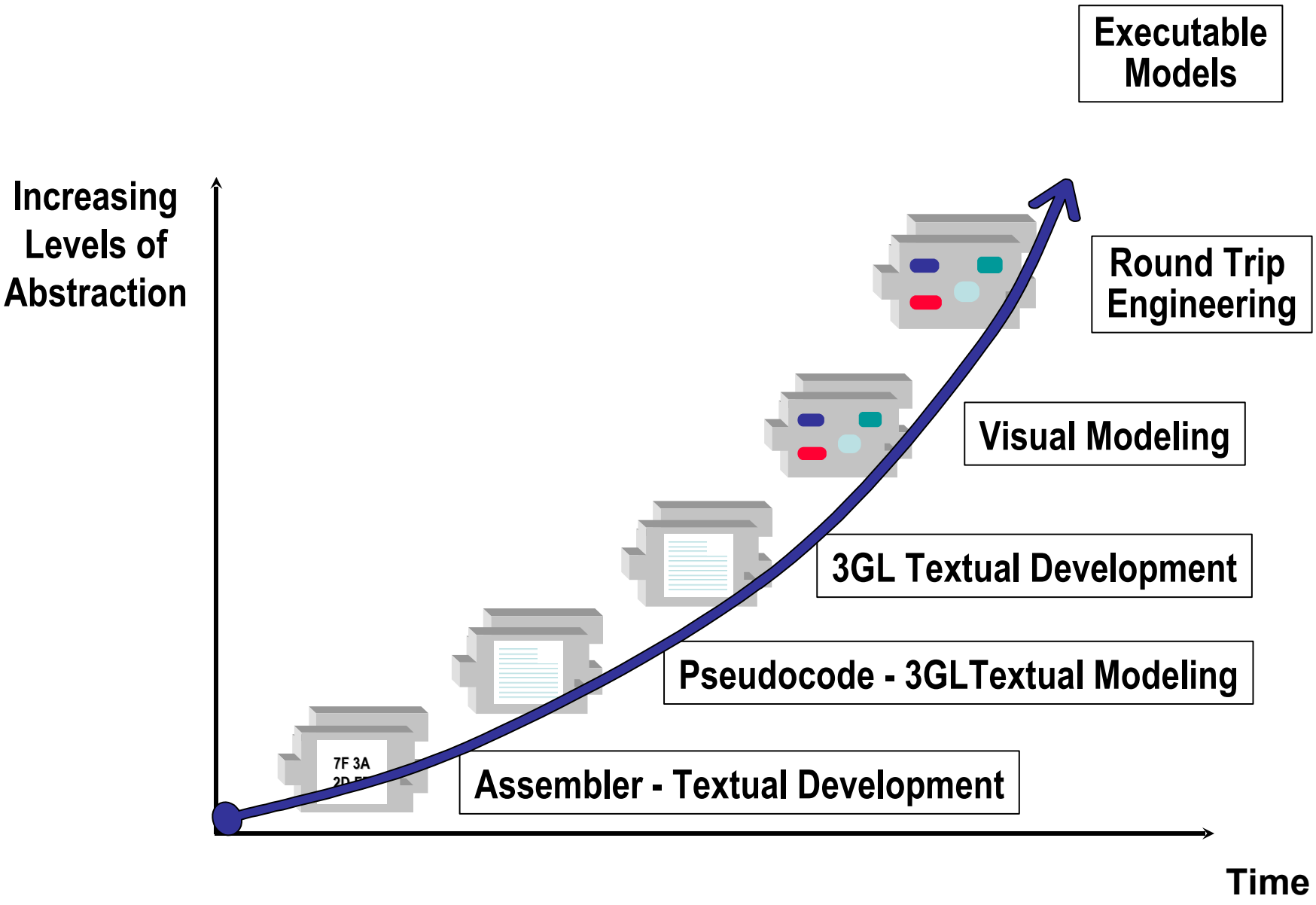
William T. Smith (smithtw@us.ibm.com)

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)*

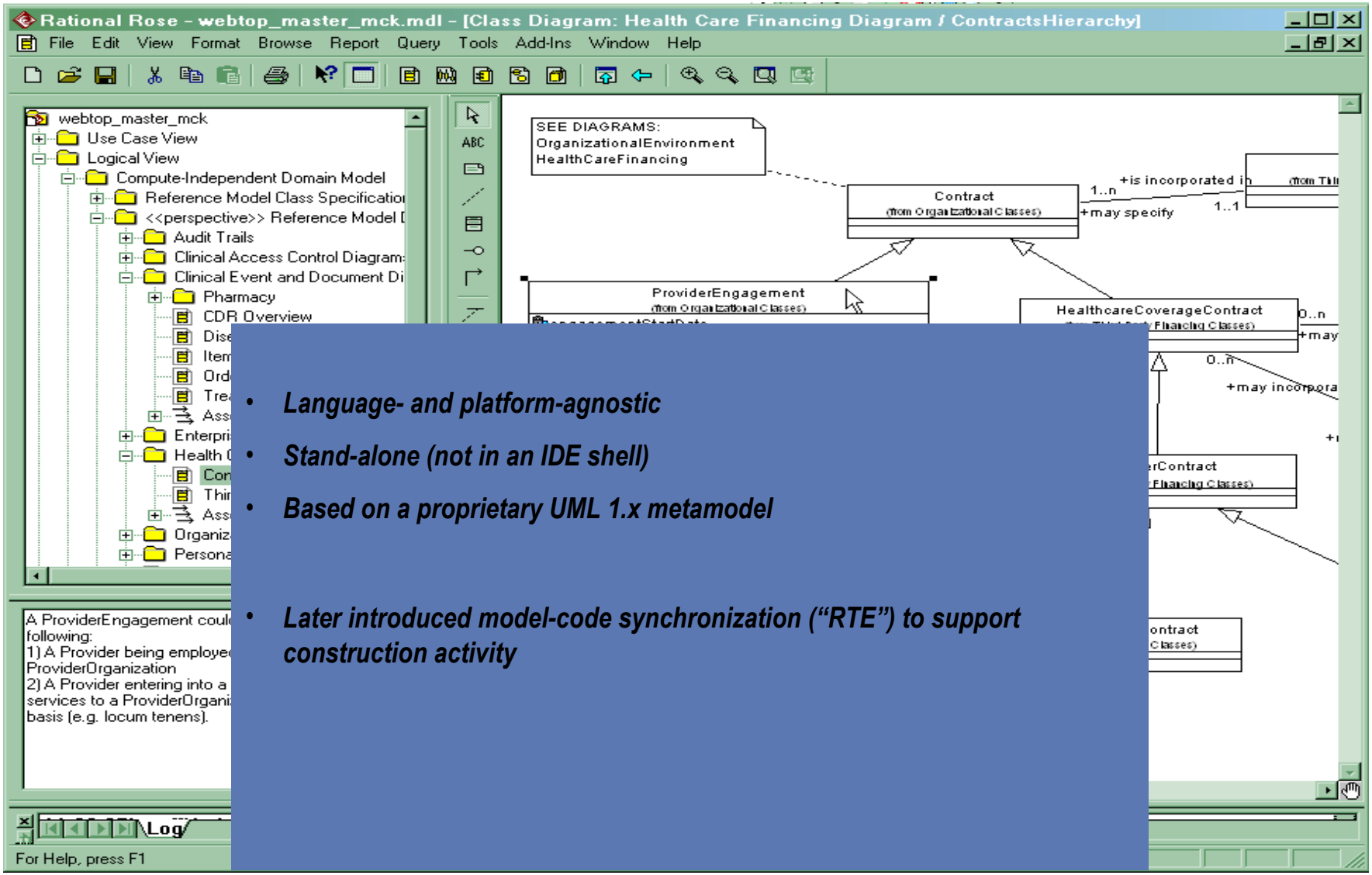
The Abstraction Curve



Market Landscape, 1995-1997

- *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…

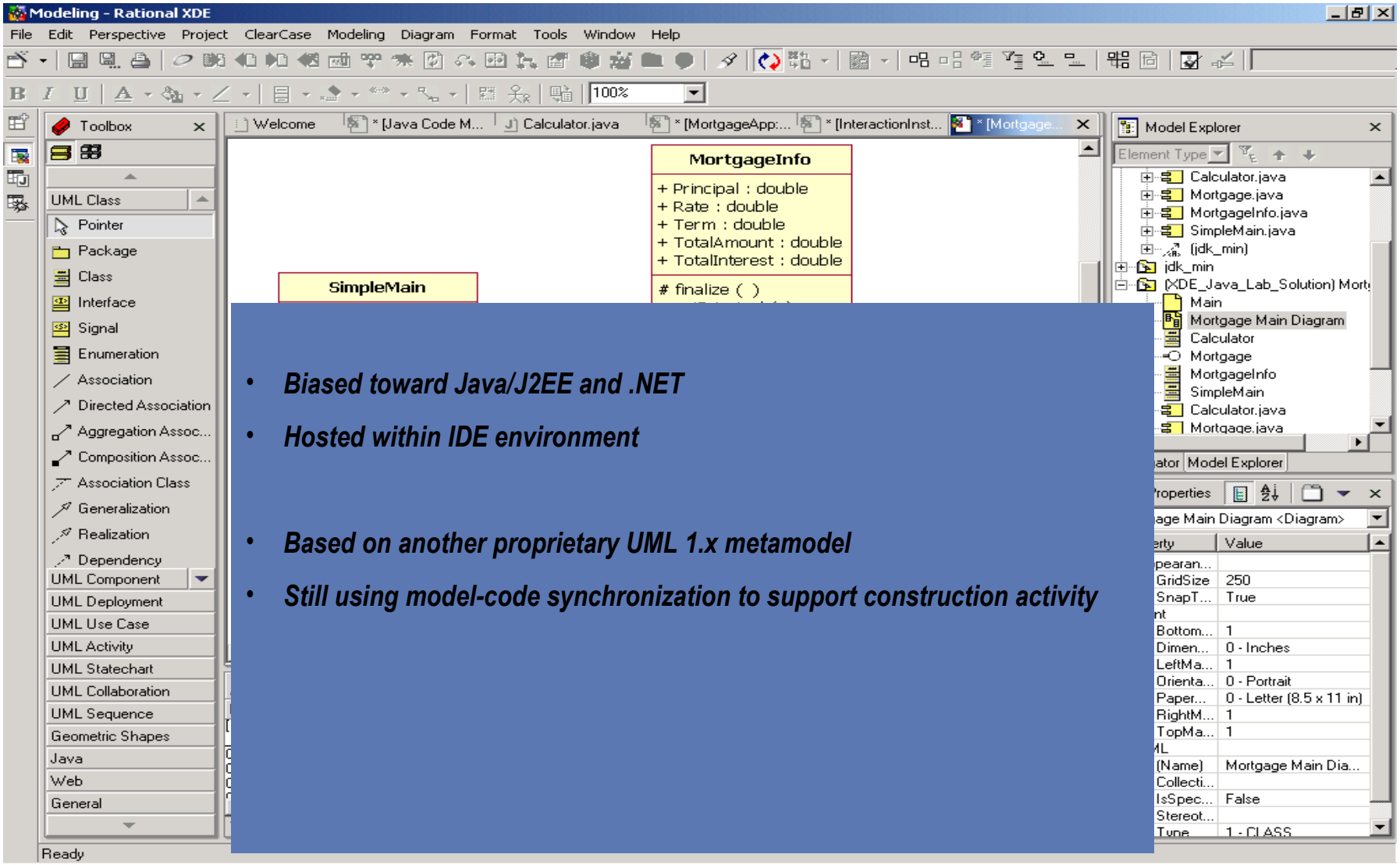


- *Language- and platform-agnostic*
- *Stand-alone (not in an IDE shell)*
- *Based on a proprietary UML 1.x metamodel*
- *Later introduced model-code synchronization ("RTE") to support construction activity*

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…



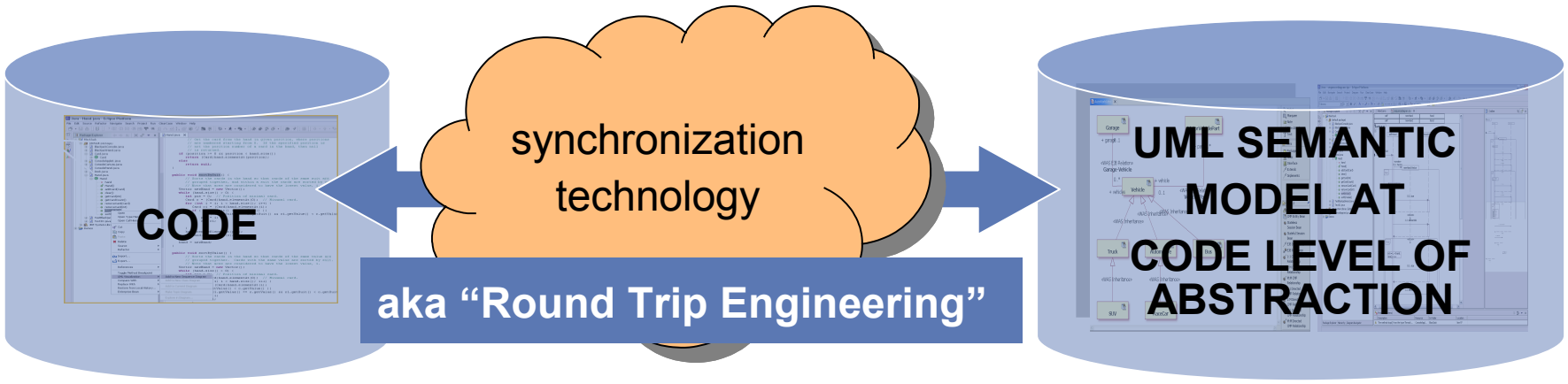
- *Biased toward Java/J2EE and .NET*
- *Hosted within IDE environment*

- *Based on another proprietary UML 1.x metamodel*
- *Still using model-code synchronization to support construction activity*

Market Landscape, 2002-…

- *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

Outsourcing & offshoring (GDD)

Bridging business, application, & data modeling

Eclipse and componentization

Maturation of frameworks and runtimes (more abstractions) MDA, DSLs, SOAs, SW Product Lines

Other open source software (incl PHP/perl/python)

Packaged app vendors coming into play

UML "reality check"

Commoditization of commercial development tools

Lighter-weight processes (Agile, XP, SCRUM, …)

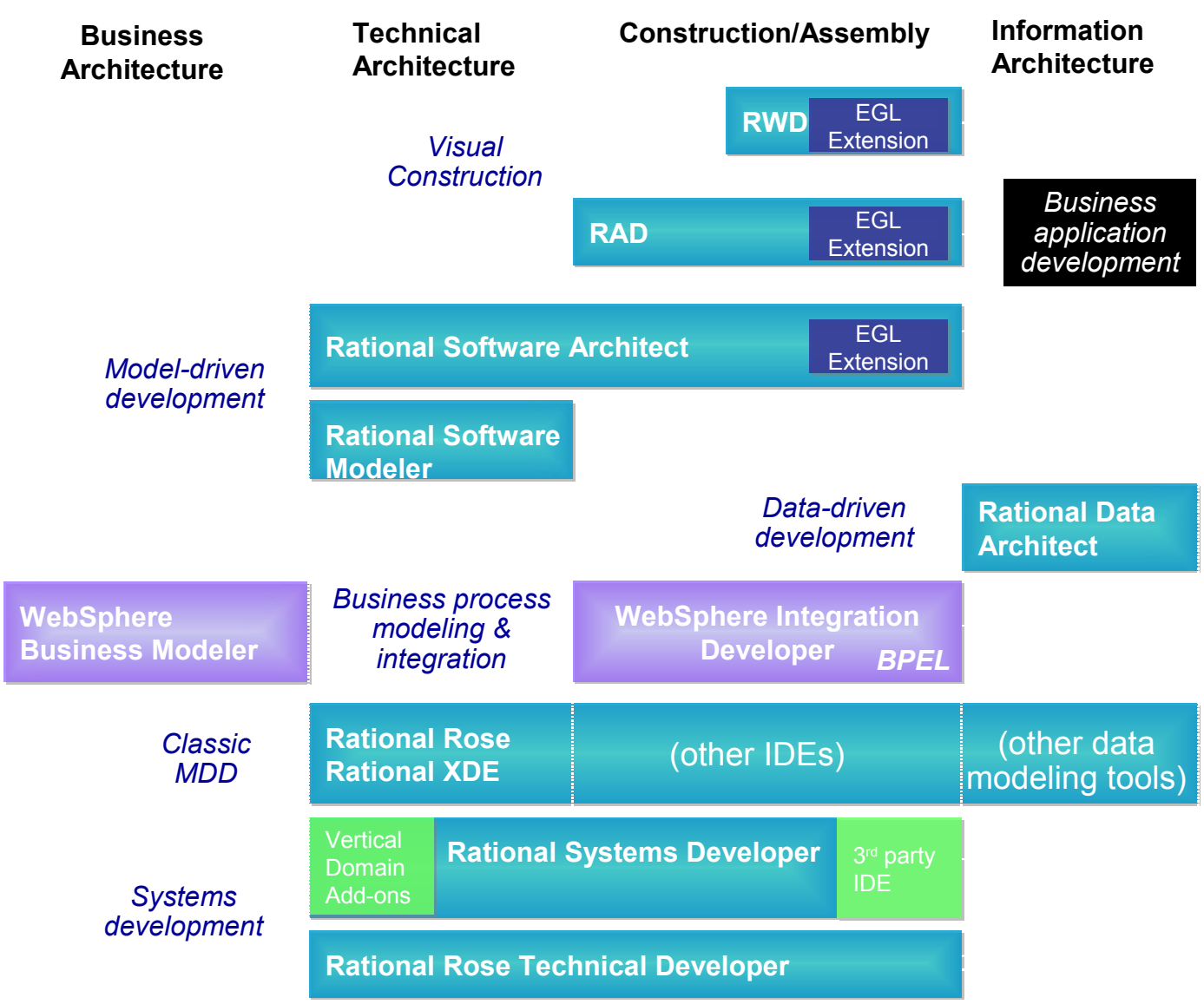
Microsoft Ventures into MDD

Systems development mkt evolution

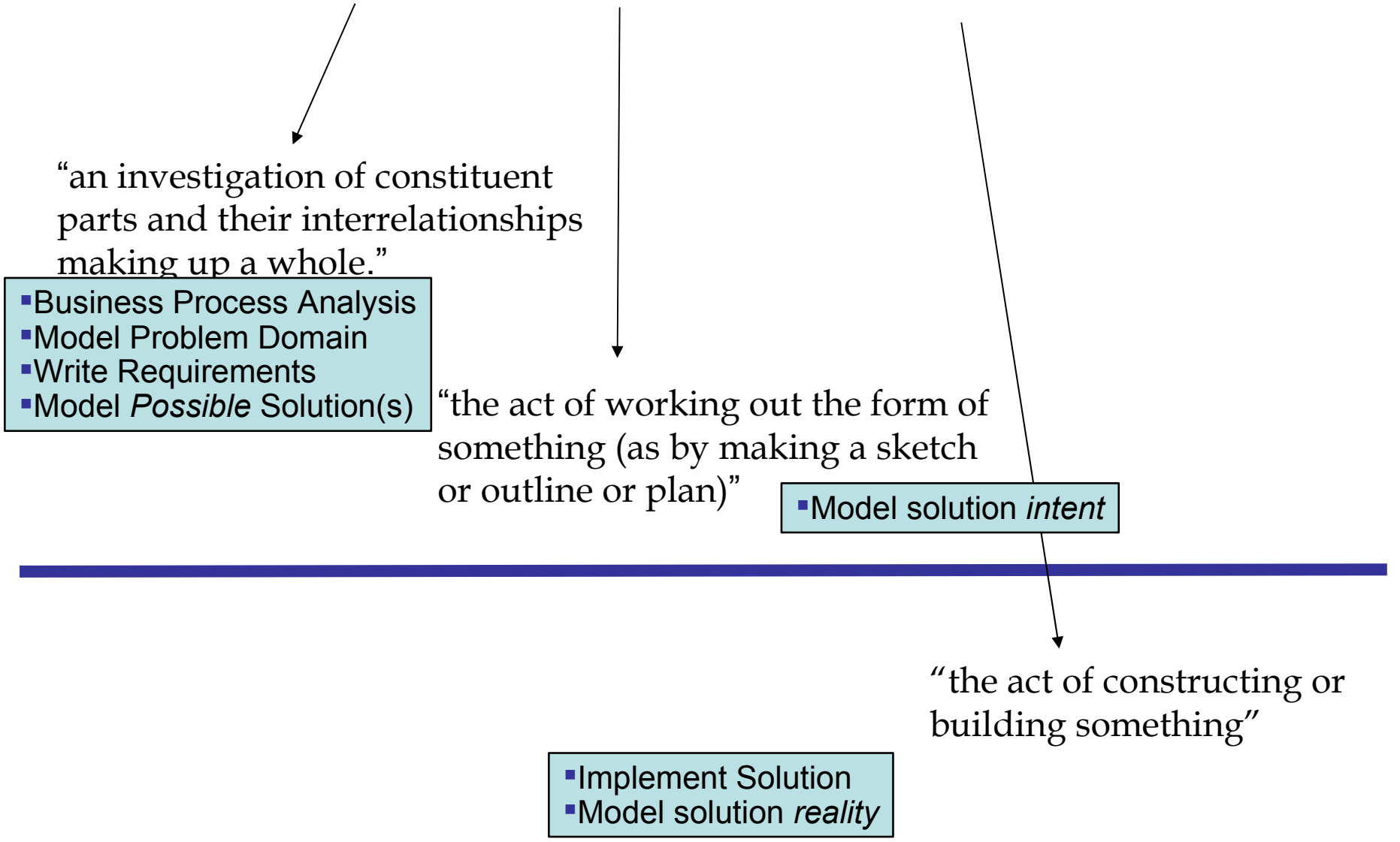
Agenda

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

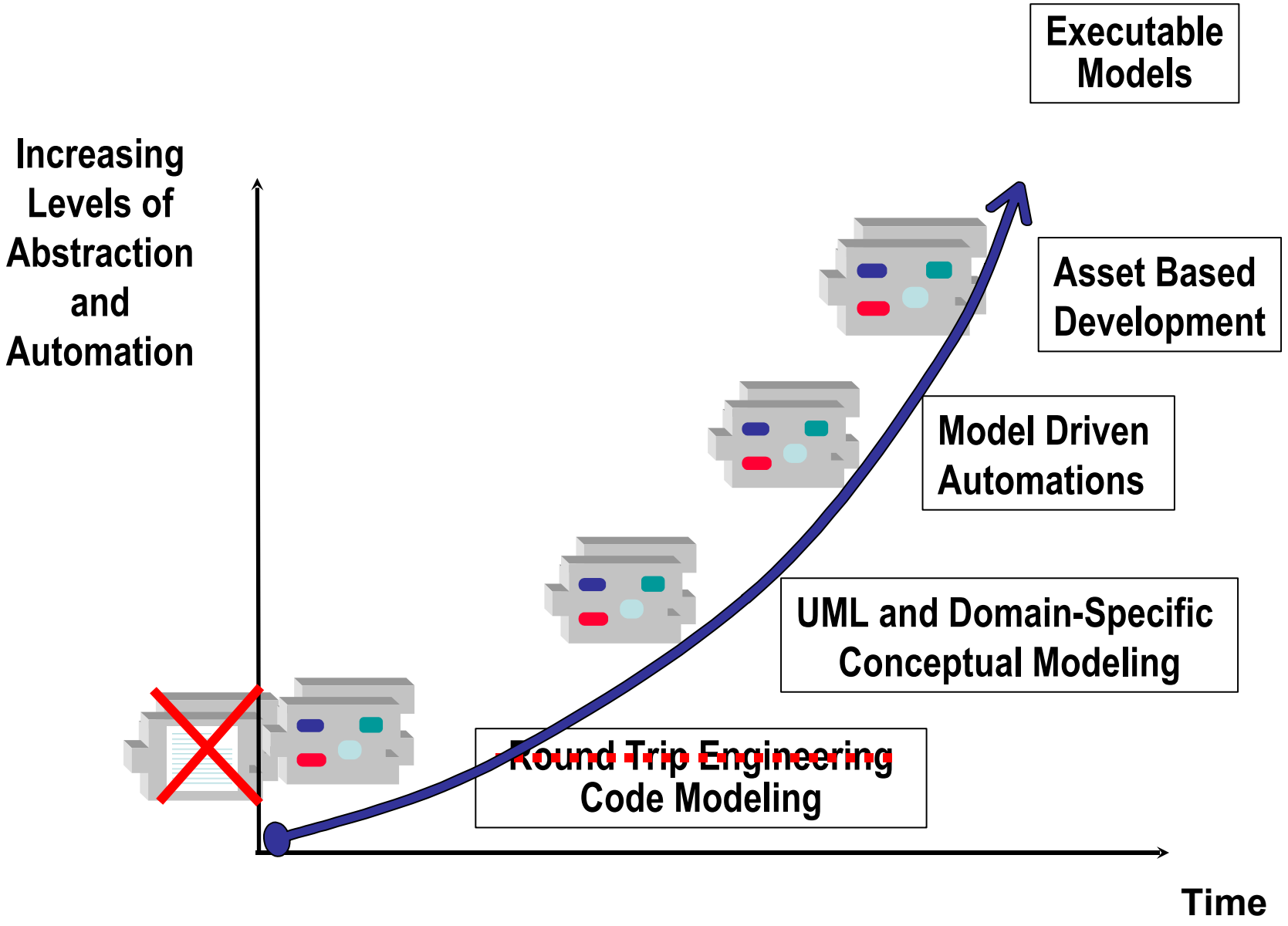
Analysis Design & Construction products



Analysis, Design, and Construction



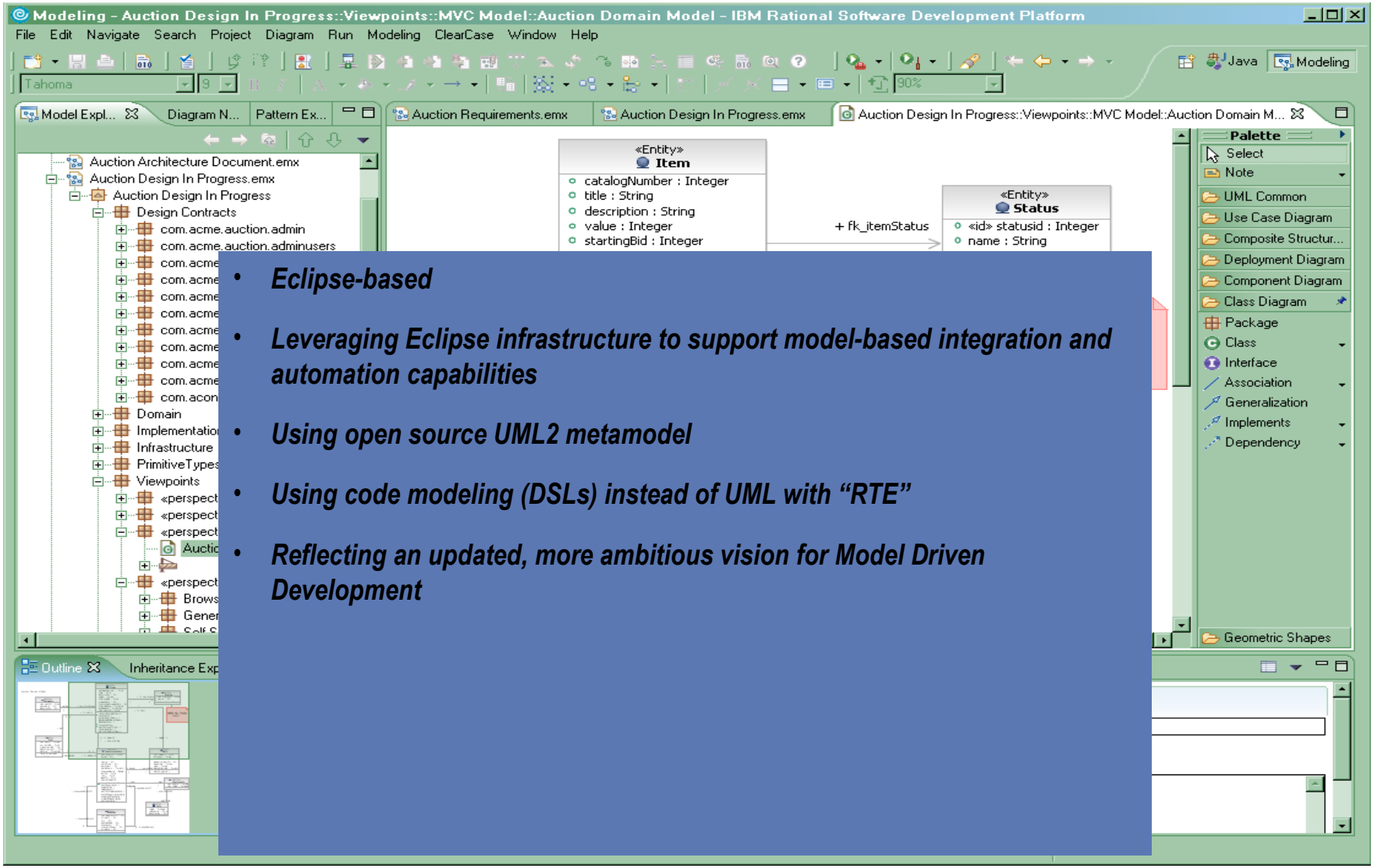
The Abstraction Curve Revisited



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…

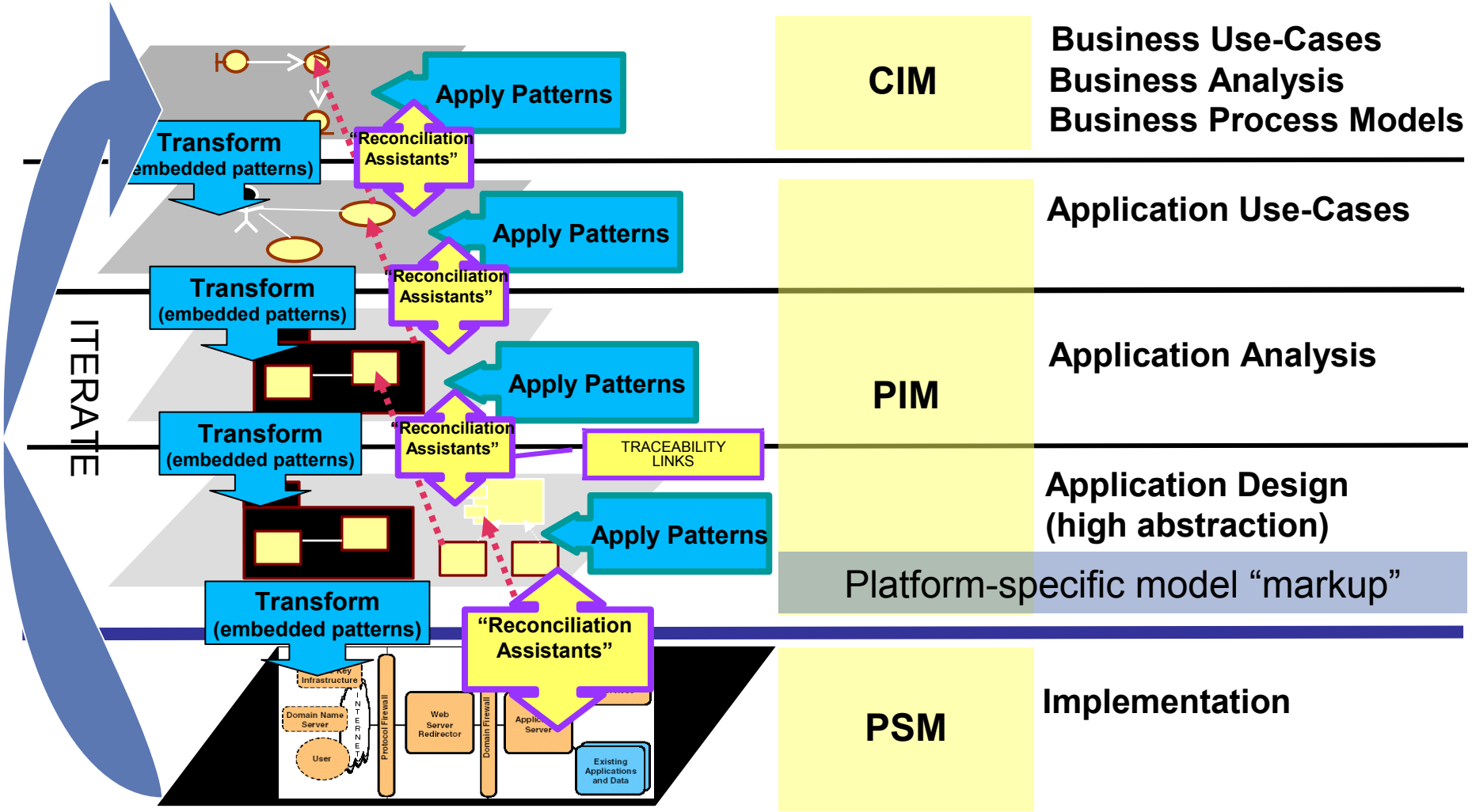


- *Eclipse-based*
- *Leveraging Eclipse infrastructure to support model-based integration and automation capabilities*
- *Using open source UML2 metamodel*
- *Using code modeling (DSLs) instead of UML with "RTE"*
- *Reflecting an updated, more ambitious vision for Model Driven Development*

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



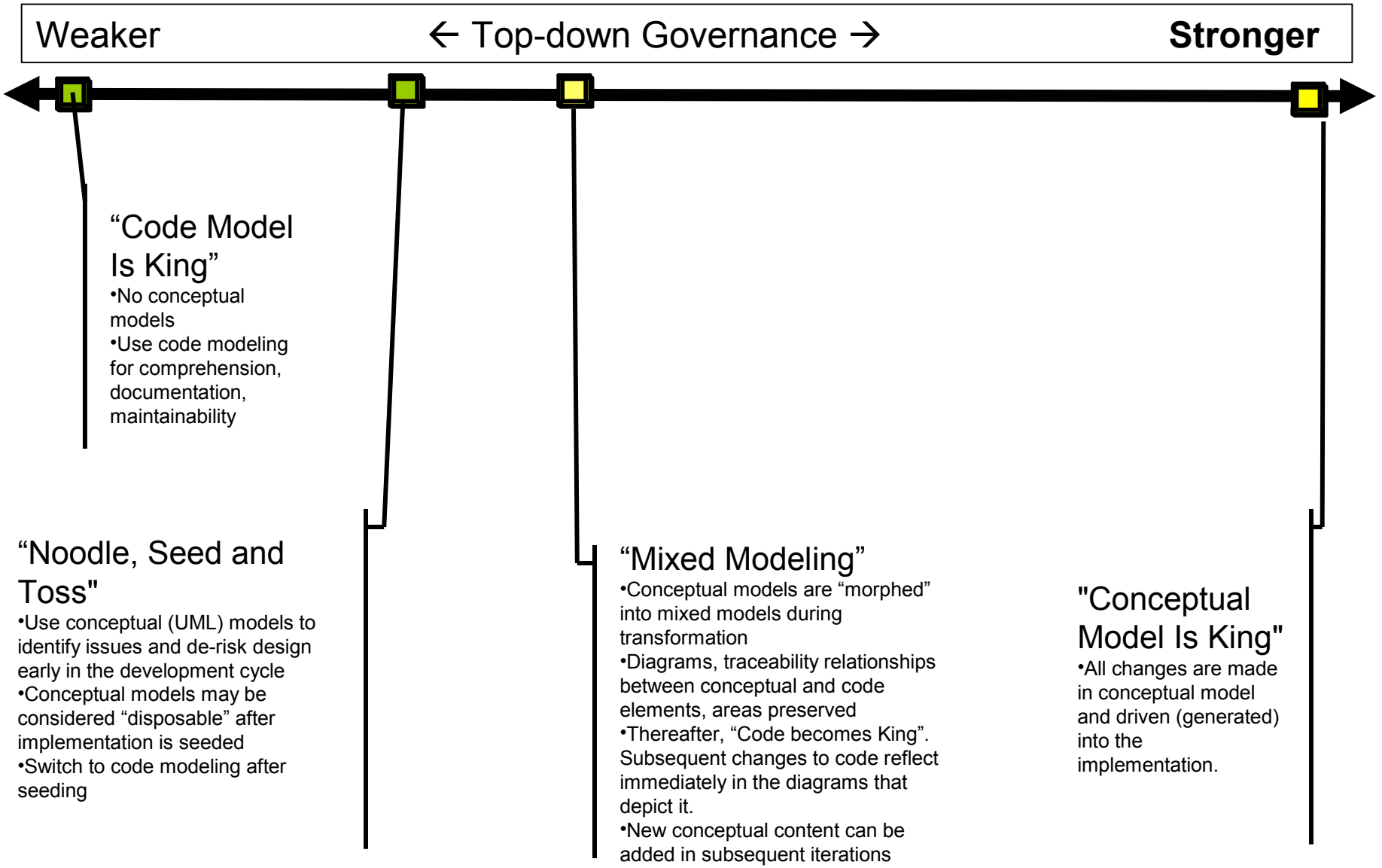
Code modeling (e.g. RAD "UML Visual Editor", VS2005 "Class Designer")

RUP

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



Terms

- ***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

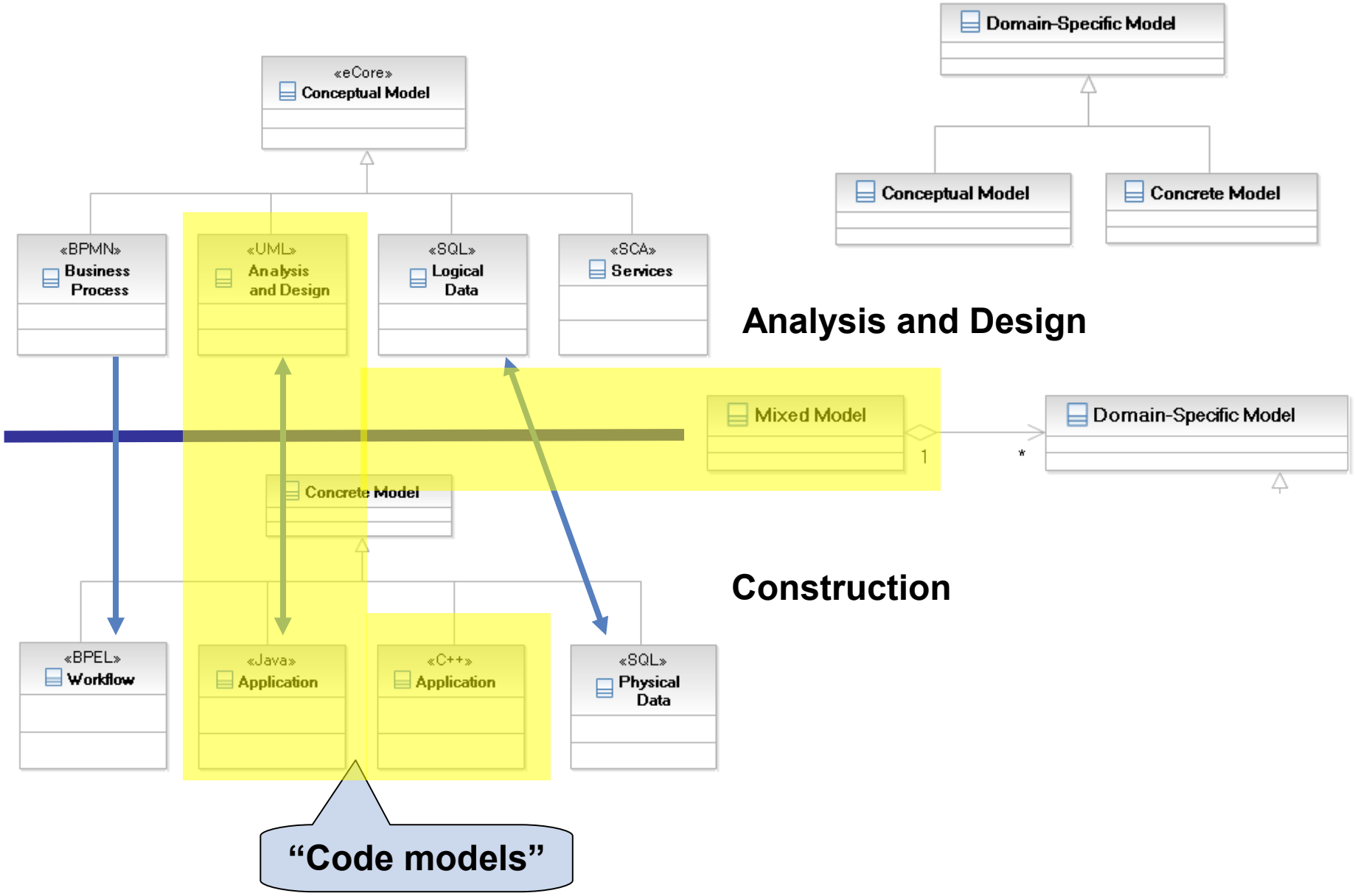
- ***UML Model***
  - A type of conceptual model based on UML semantics (i.e. a UML metamodel)
  - Contains UML semantic elements plus UML diagrams whose notational elements reflect underlying UML semantic elements
  - In RSx, persisted as "UML model files" (with .emx extensions) typically contained 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

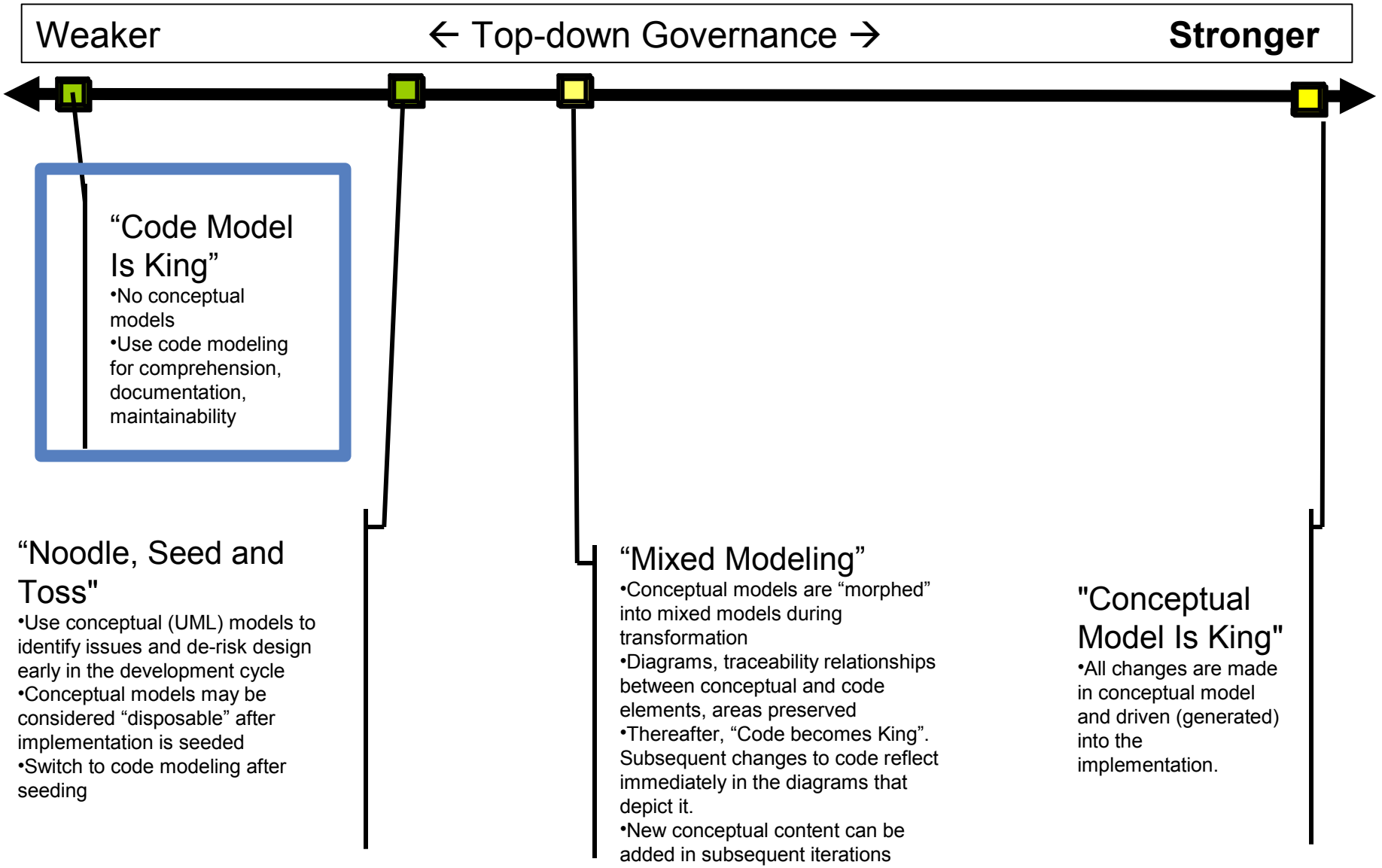
- ***Code Model***
  - A model based on 3GL semantics (e.g. a Java metamodel or CLR metamodel)
  - Contains 3GL semantic elements and relationships, plus 'free standing' diagrams whose notational elements reflect underlying 3GL semantic elements
  - In RSx, serialized and persisted as source code files (semantics) and diagram files (with .dnx 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

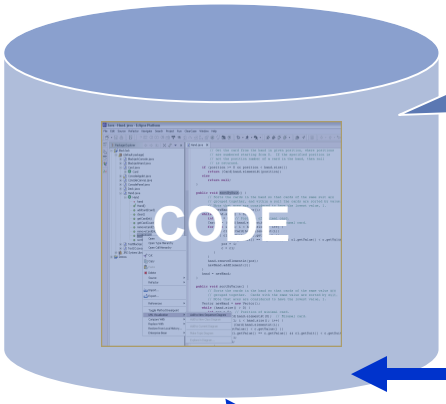
Models Taxonomy



MDD “Theories of Operations” Map To Governance Philosophies

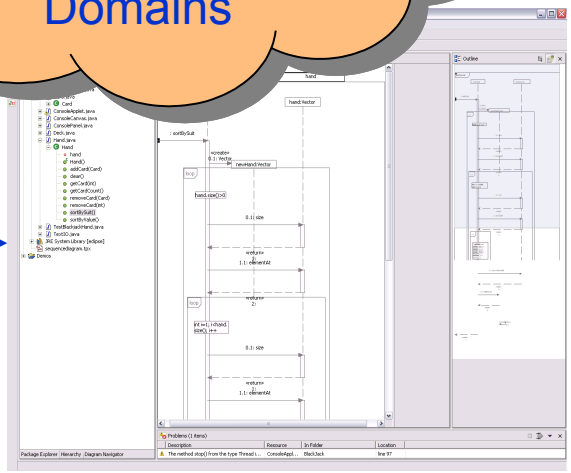


Code Modeling with Rational

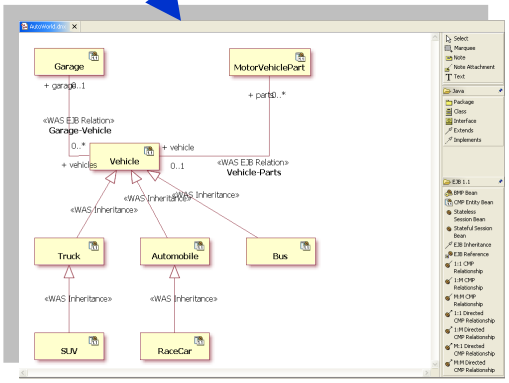


The only source of semantic information is here

Domain-Specific Languages for 3GL Implementation Domains

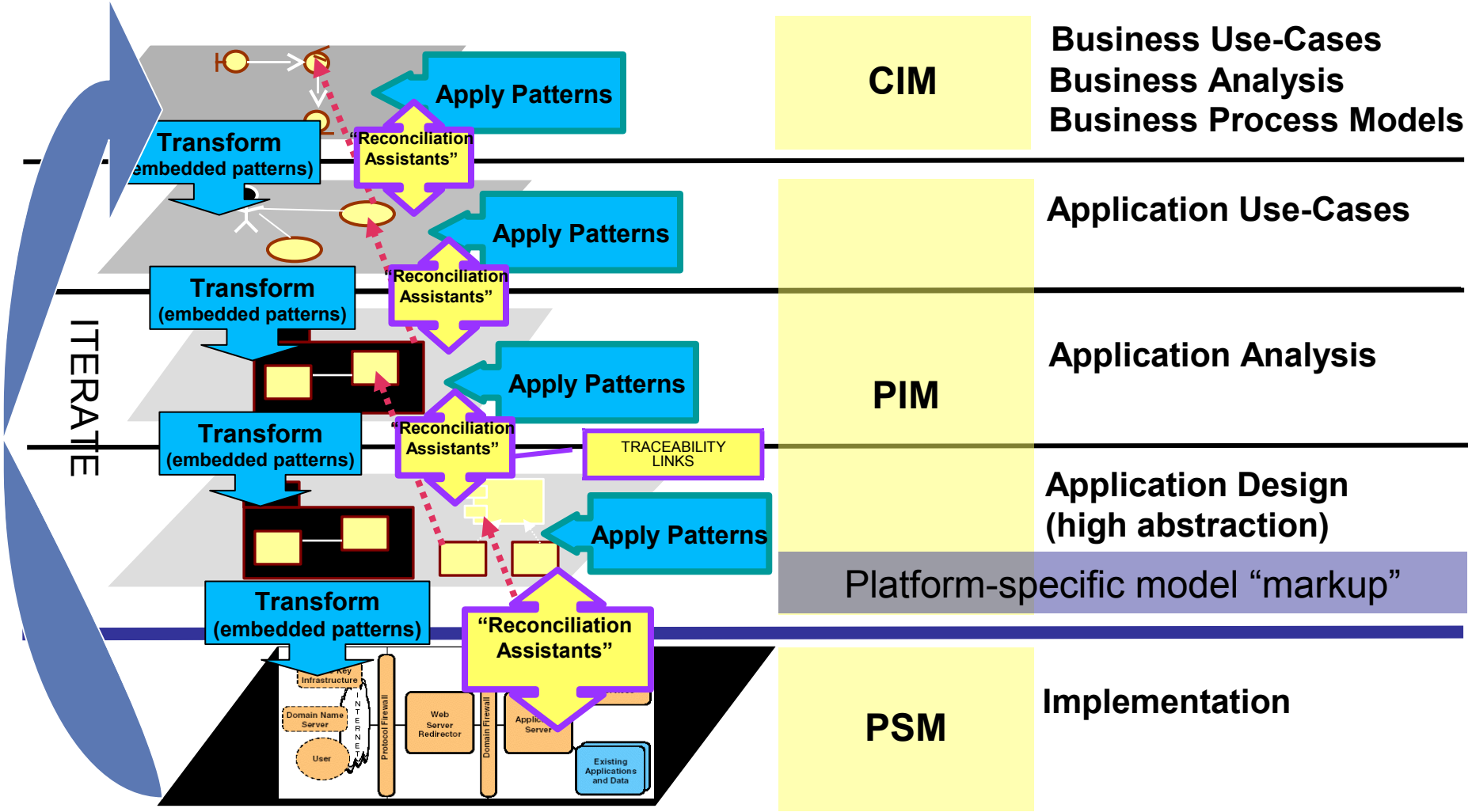


Diagrams that directly reflect and in some cases edit the source code



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

“Code Model Is King”

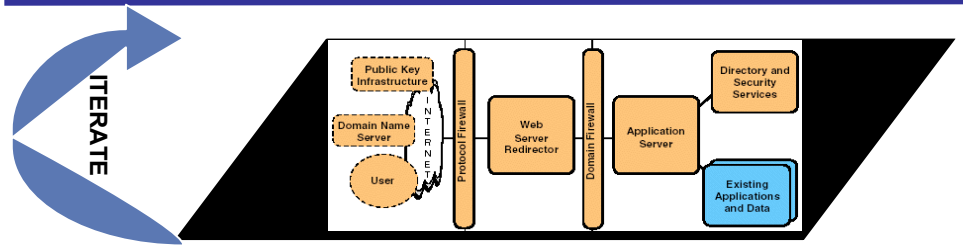


Code modeling (e.g. RAD “UML Visual Editor”, VS2005 “Class Designer”)

RUP

“Code Model Is King”

Analysis, design



Implementation

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

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

"Code Model Is King"

- No conceptual models
- Use code modeling for comprehension, documentation, maintainability

"Noodle, Seed and Toss"

- Use conceptual (UML) models to identify issues and de-risk design early in the development cycle
- Conceptual models may be considered "disposable" after implementation is seeded
- Switch to code modeling after seeding

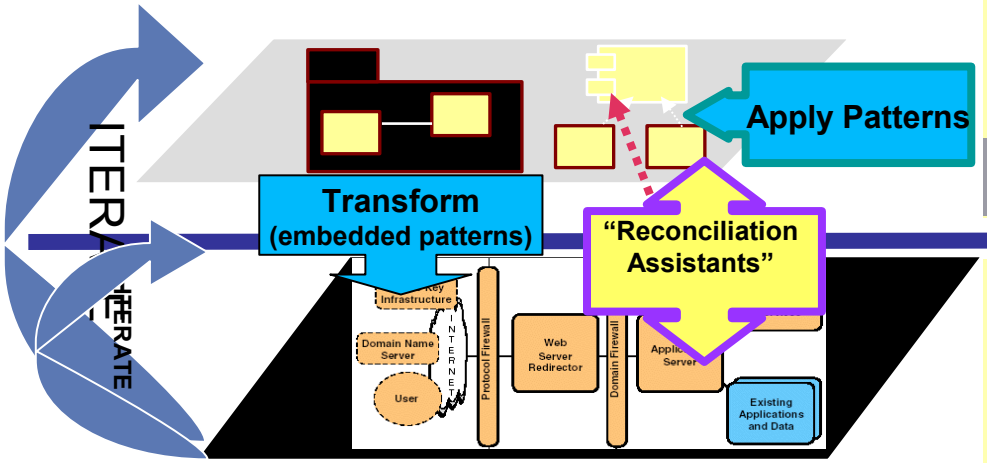
"Mixed Modeling"

- Conceptual models are "morphed" into mixed models during transformation
- Diagrams, traceability relationships between conceptual and code elements, areas preserved
- Thereafter, "Code becomes King". Subsequent changes to code reflect immediately in the diagrams that depict it.
- New conceptual content can be added in subsequent iterations

"Conceptual Model Is King"

- All changes are made in conceptual model and driven (generated) into the implementation.

"Noodle, Seed, and Toss"



PIM — **Application Design (higher abstraction)**

Platform-specific model "markup"

PSM — **Implementation**

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

MDA

RUP
Custom
Configuration

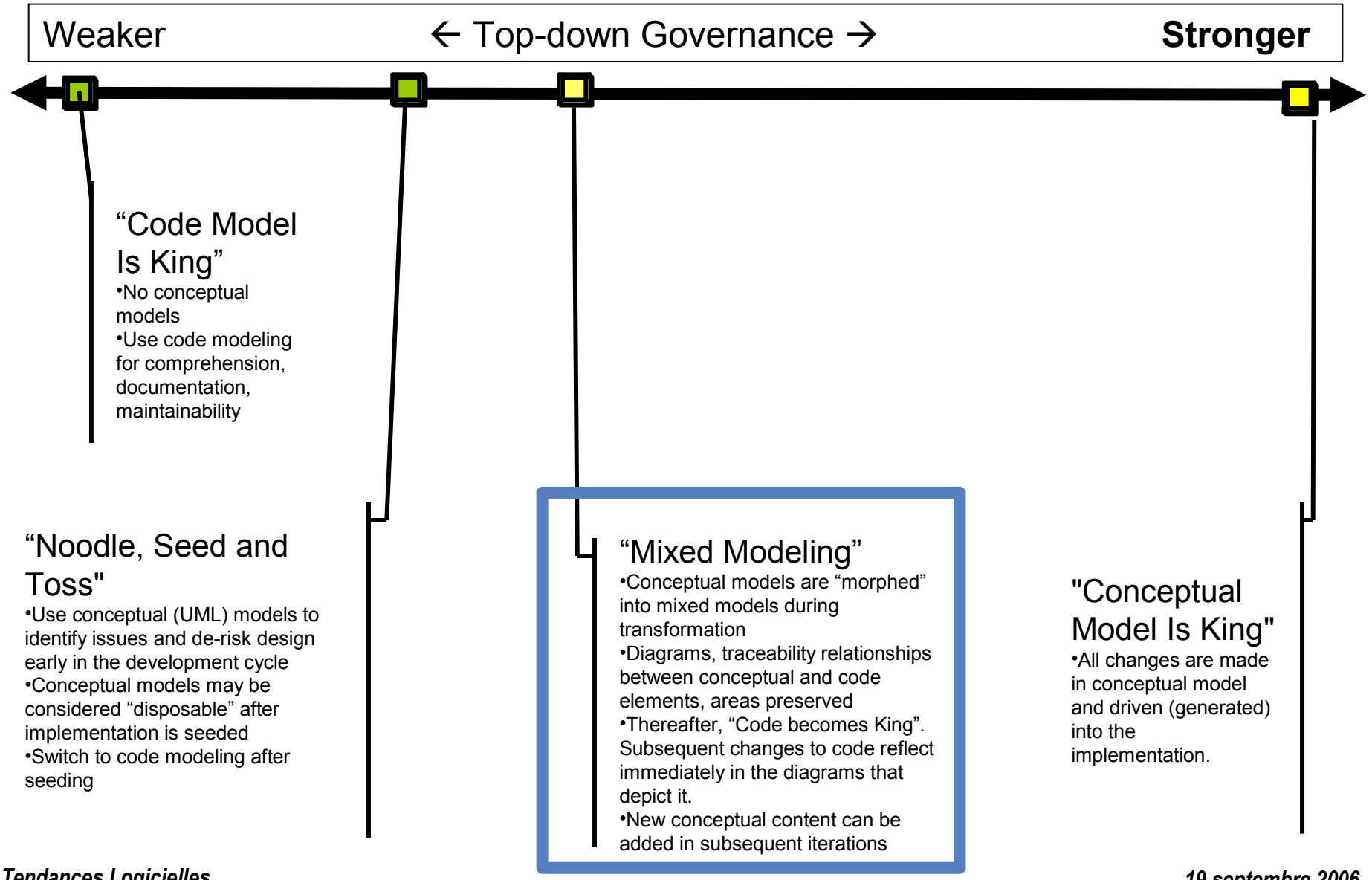
Demo:

“Noodle, Seed, and Toss”

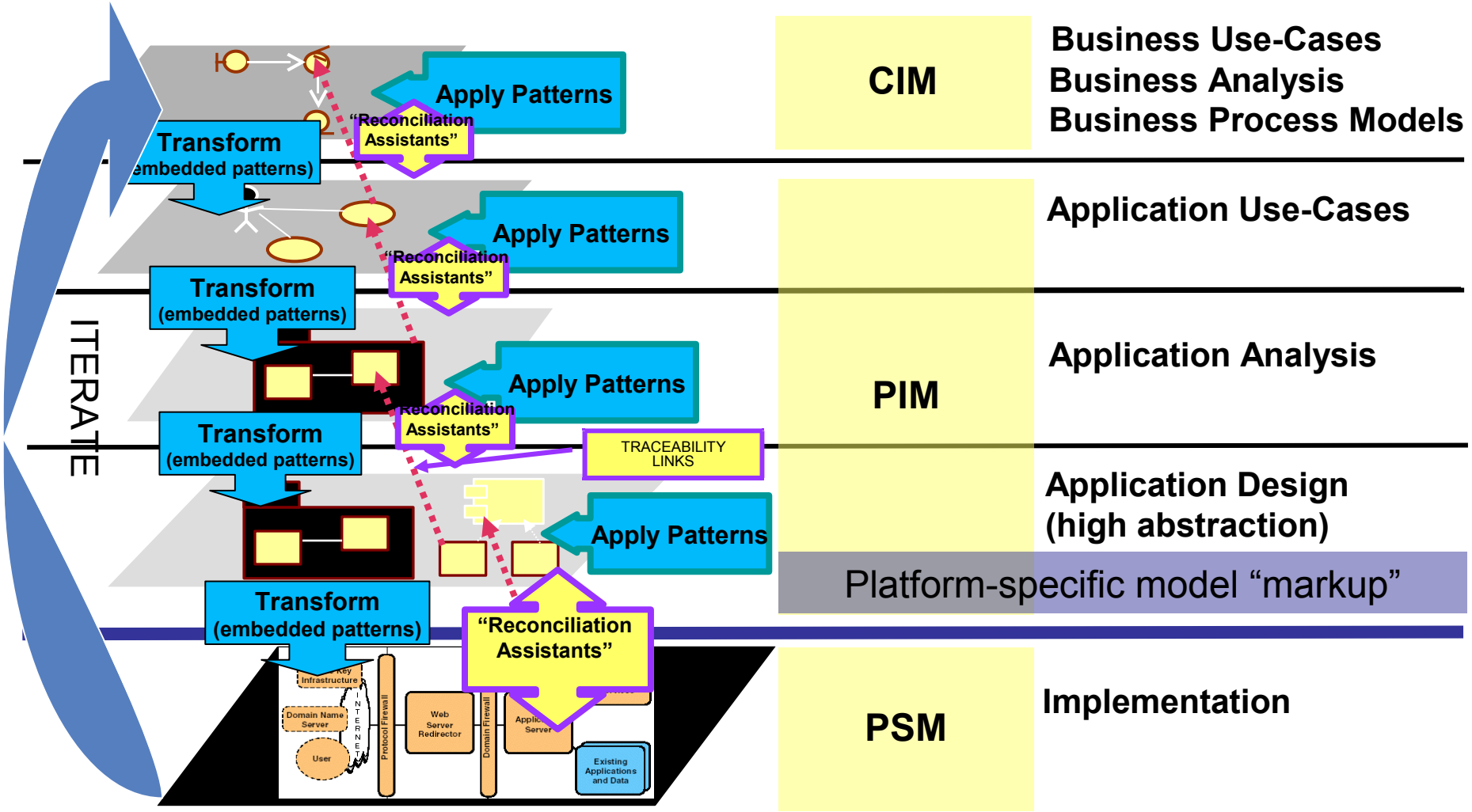
"Noodle, Seed, and Toss" Value Proposition

- *"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



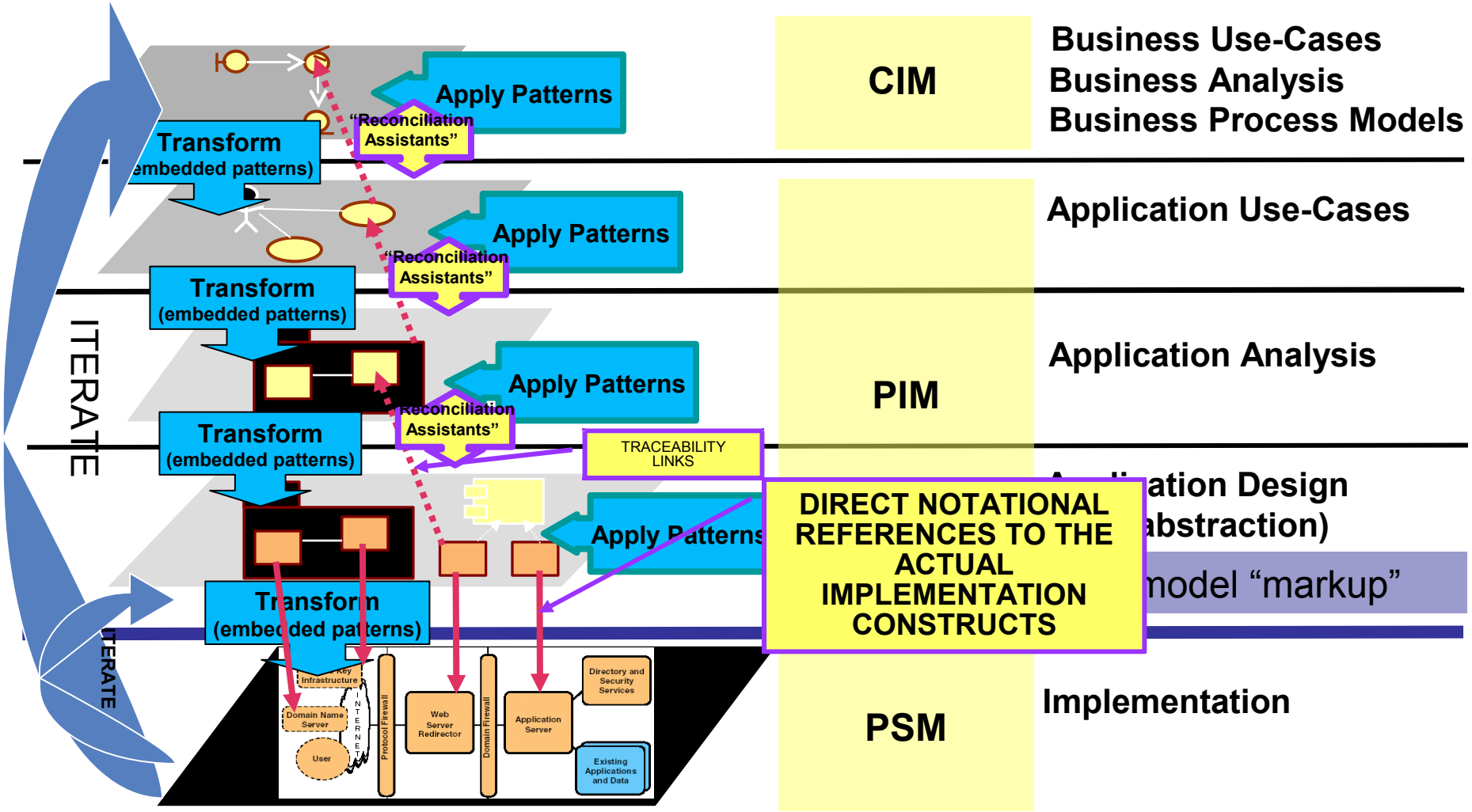
“Mixed Modeling”



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

RUP

“Mixed Modeling”



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

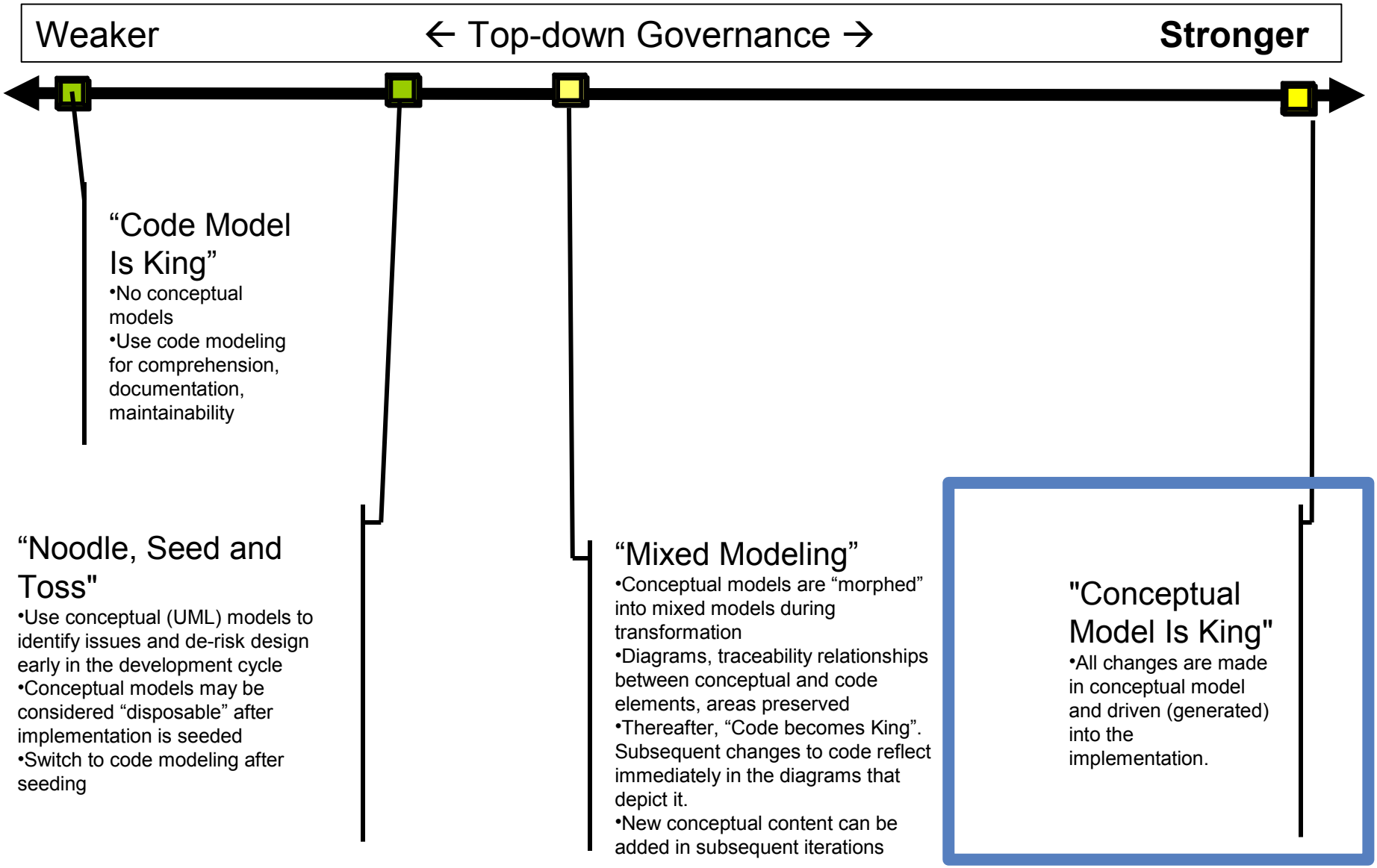
RUP

Demo: “Mixed Modeling”

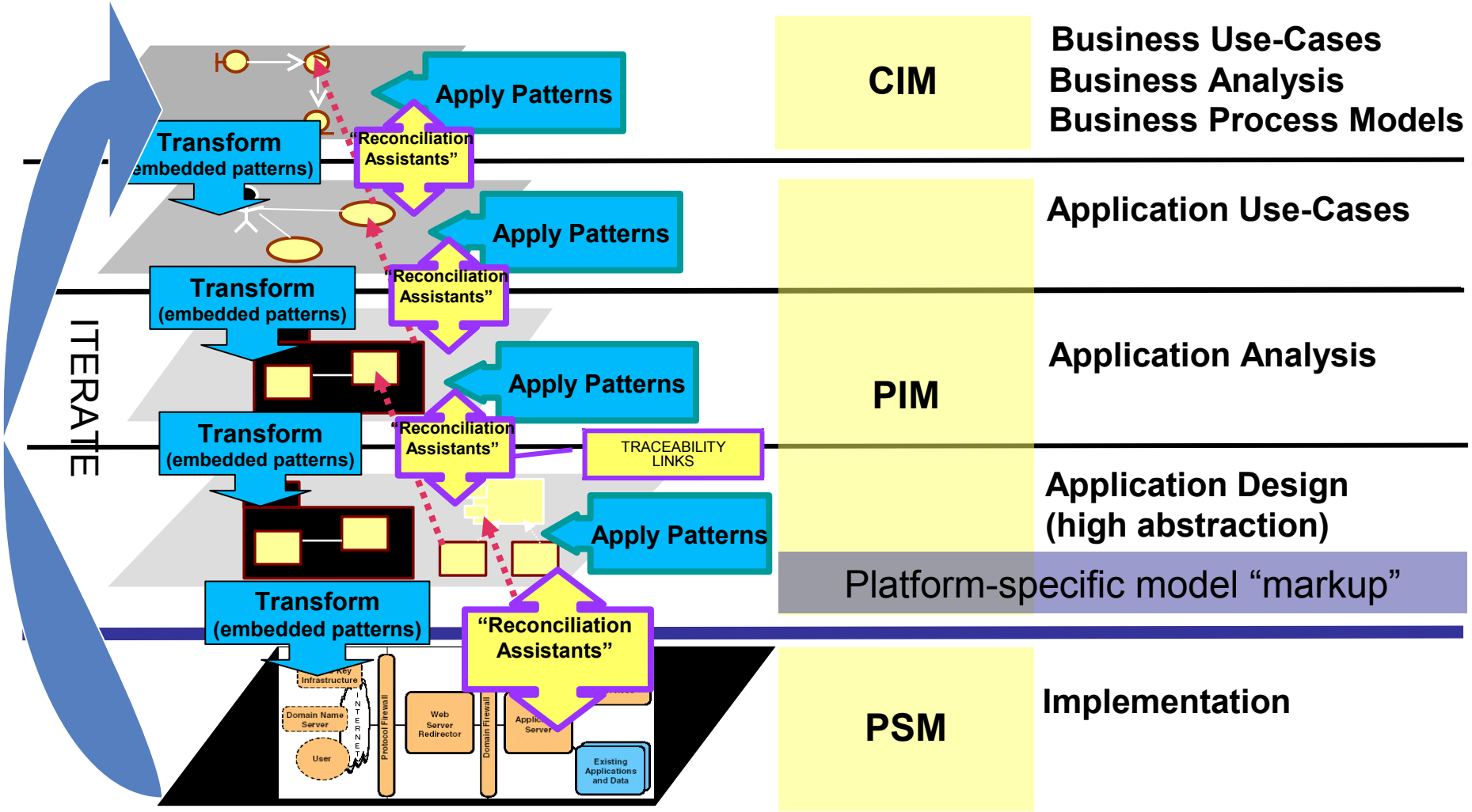
"Mixed Modeling" Value Proposition

- *"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”)

RUP

"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"

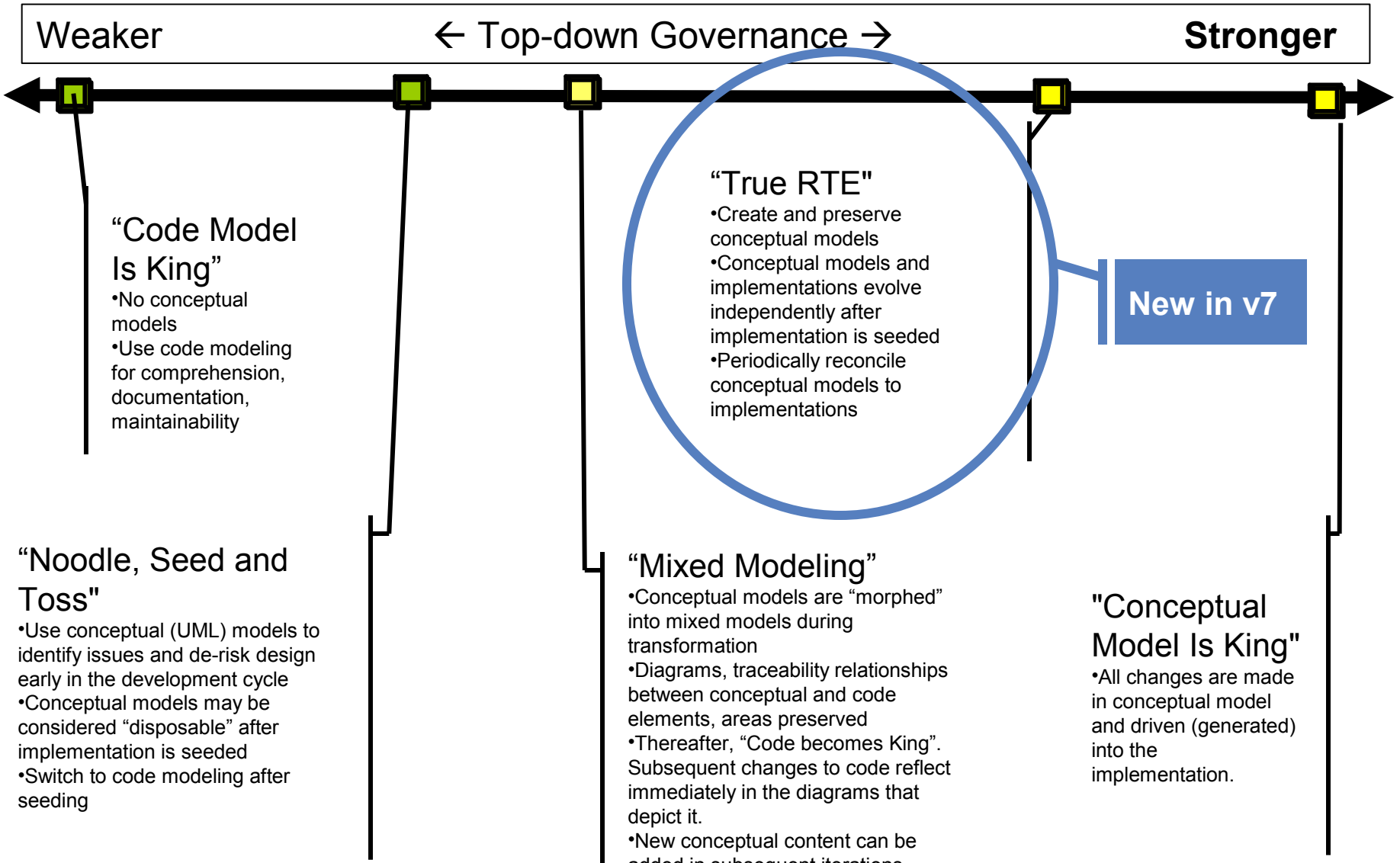
Agenda

- *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)*

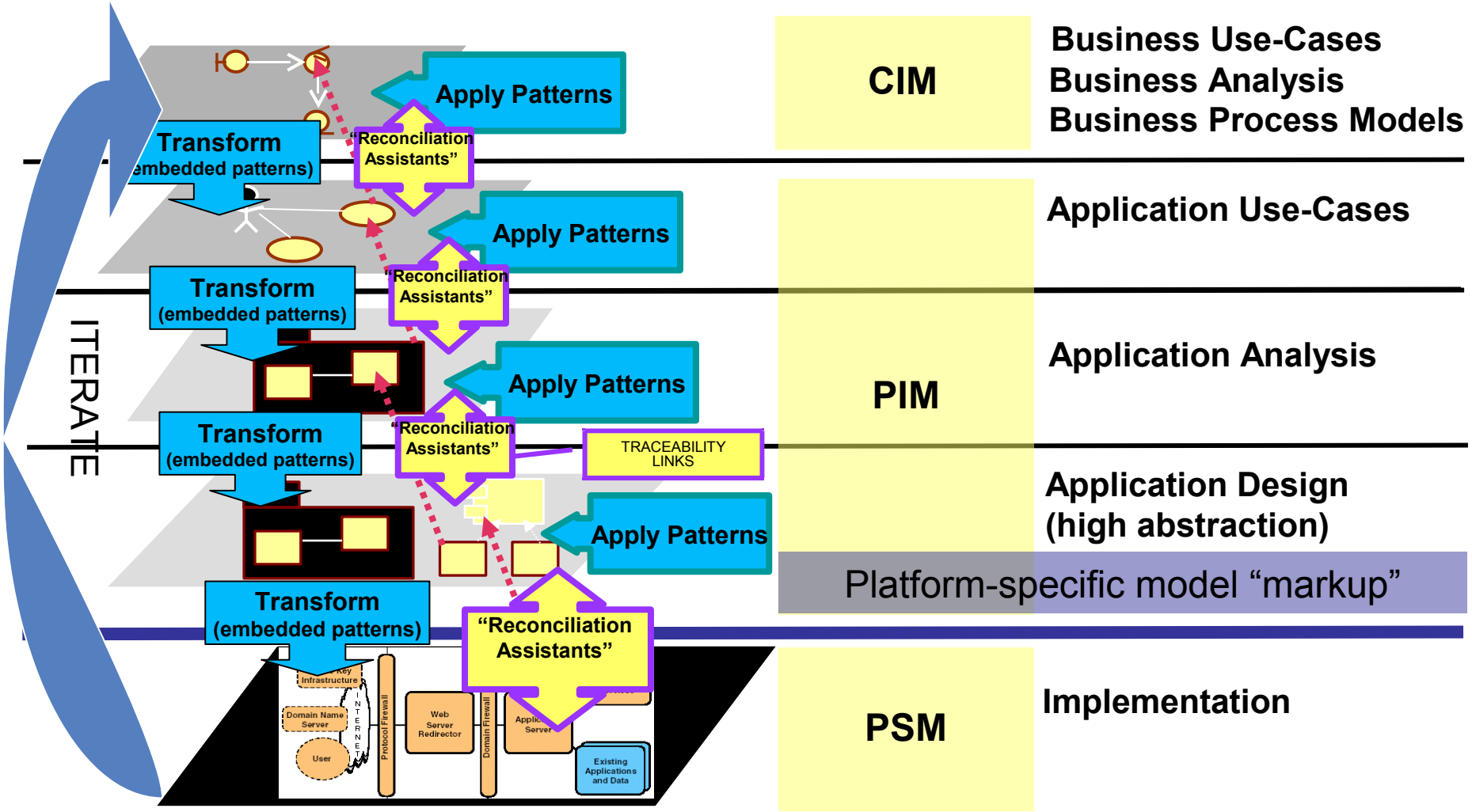
Key Elements of the MDD Vision

- ***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



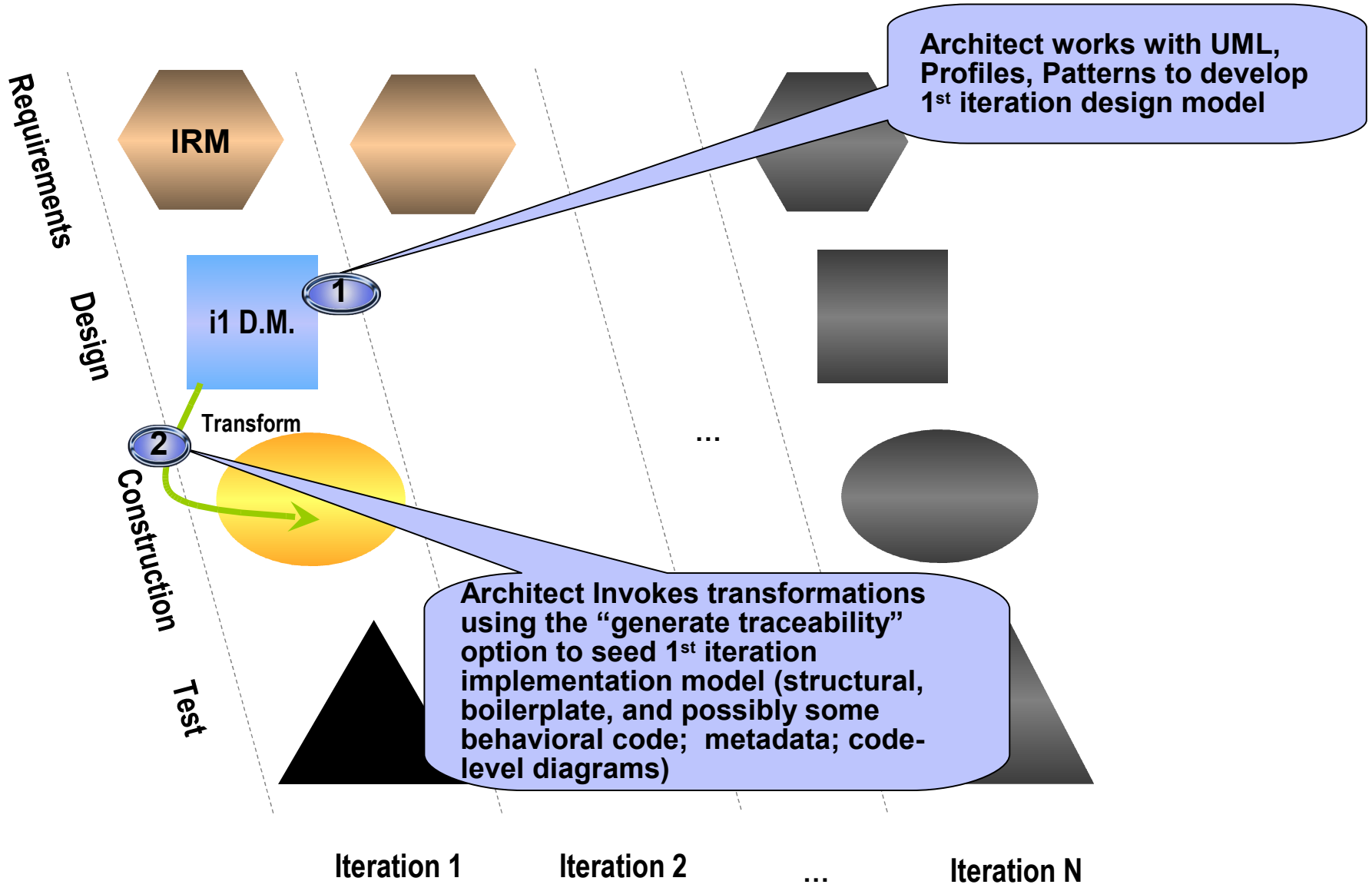
“True RTE”



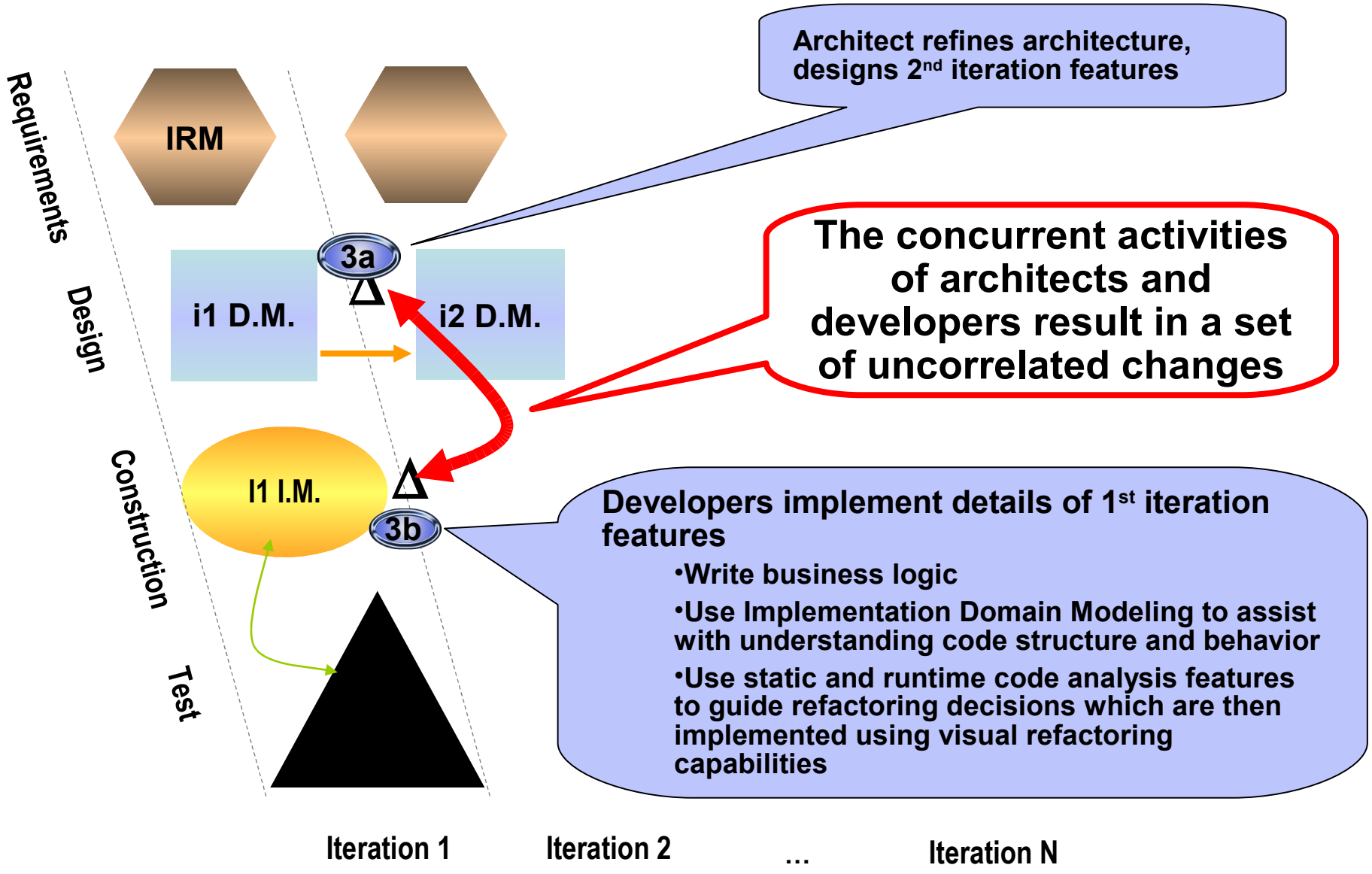
Code modeling (e.g. RAD “UML Visual Editor”, VS2005 “Class Designer”)

RUP

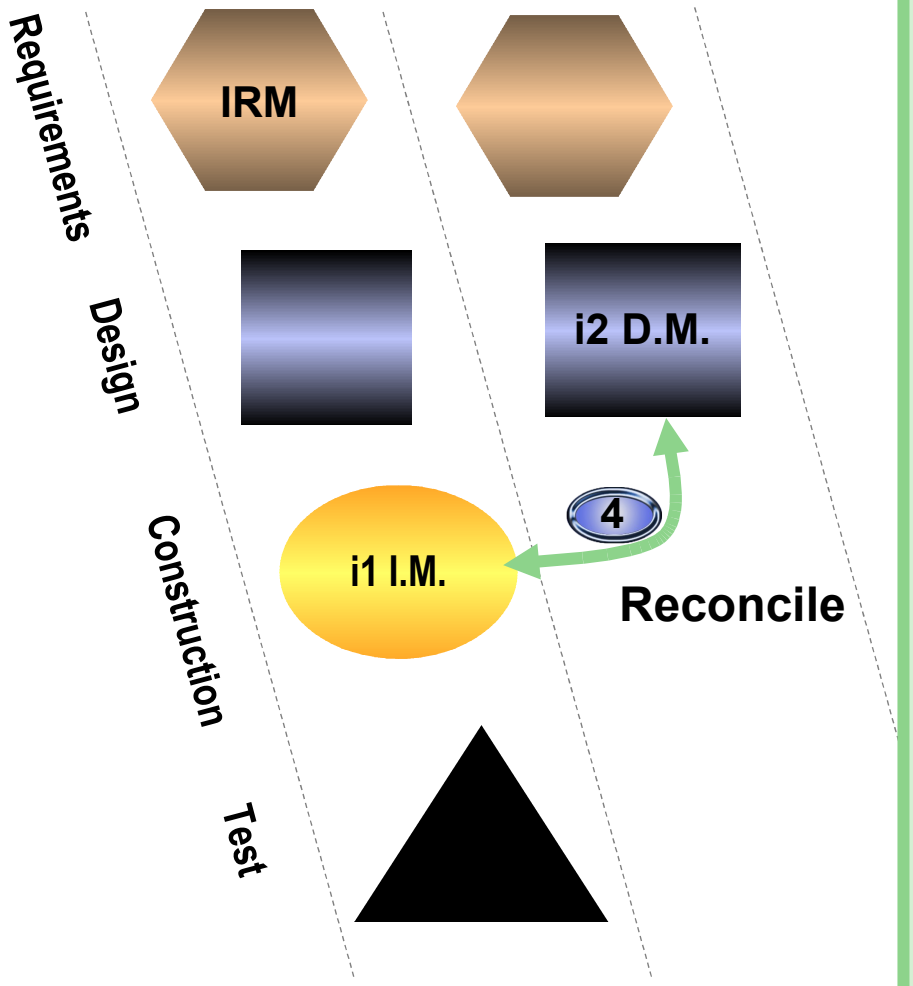
"True RTE" (1,2)


Architect works with UML, Profiles, Patterns to develop 1st iteration design model
Requirements
Design
Construction
Test
IRM
i1 D.M.
1
Transform
2
...
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)
Iteration 1
Iteration 2
...
Iteration N

"True RTE" (3)


Architect refines architecture, designs 2nd iteration features
IRM
Requirements
Design
Construction
Test
3a
i1 D.M.
i2 D.M.
The concurrent activities of architects and developers result in a set of uncorrelated changes
I1 I.M.
3b
Developers implement details of 1st iteration features
•Write business logic
•Use Implementation Domain Modeling to assist with understanding code structure and behavior
•Use static and runtime code analysis features to guide refactoring decisions which are then implemented using visual refactoring capabilities
Iteration 1
Iteration 2
…
Iteration N

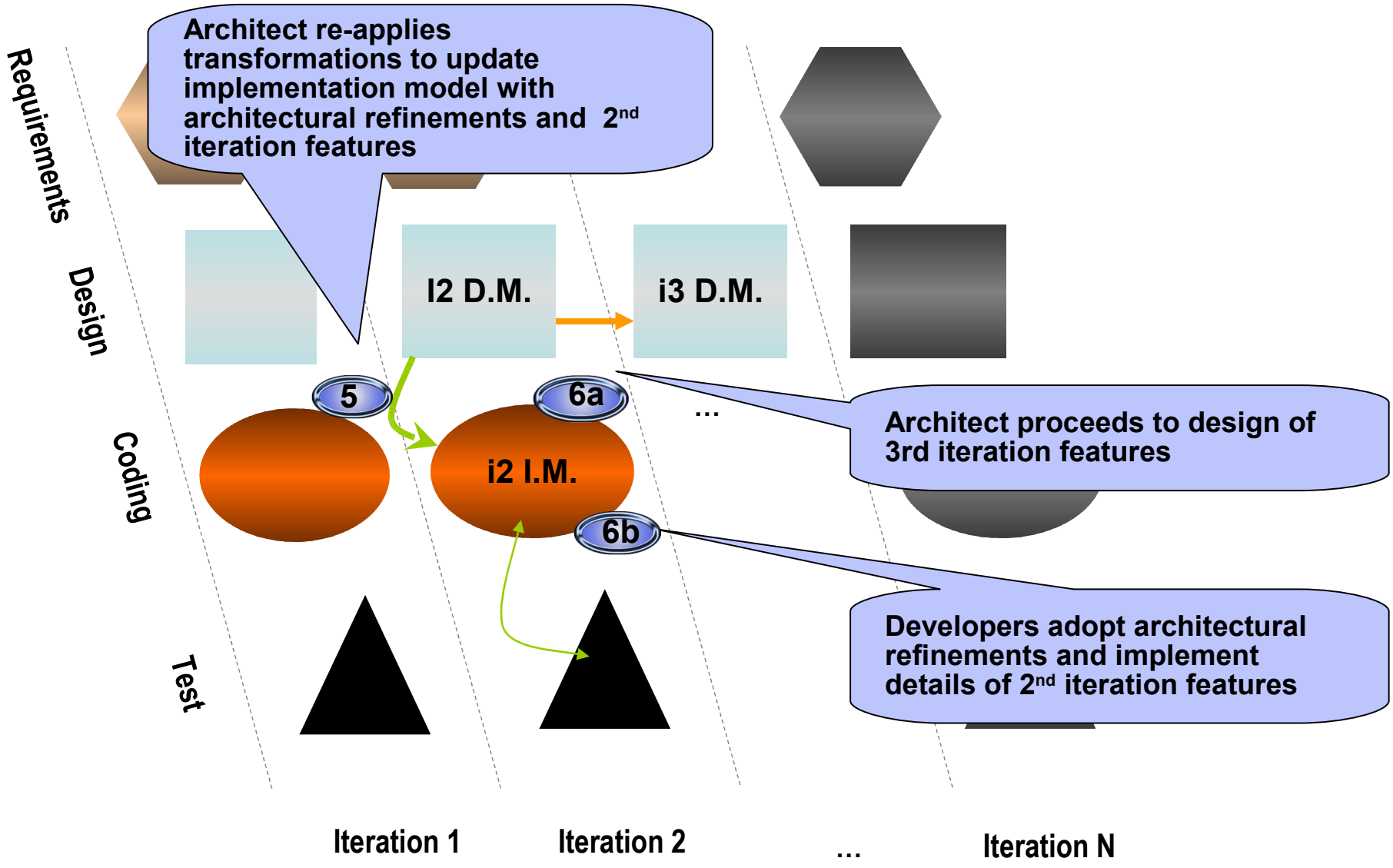
"True RTE" (4)



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)


Architect re-applies transformations to update implementation model with architectural refinements and 2nd iteration features
Requirements
Design
Coding
Test
I2 D.M.
i3 D.M.
5
6a
i2 I.M.
6b
...
Architect proceeds to design of 3rd iteration features
Developers adopt architectural refinements and implement details of 2nd iteration features
Iteration 1
Iteration 2
...
Iteration N

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 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."

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 Modeling in v7

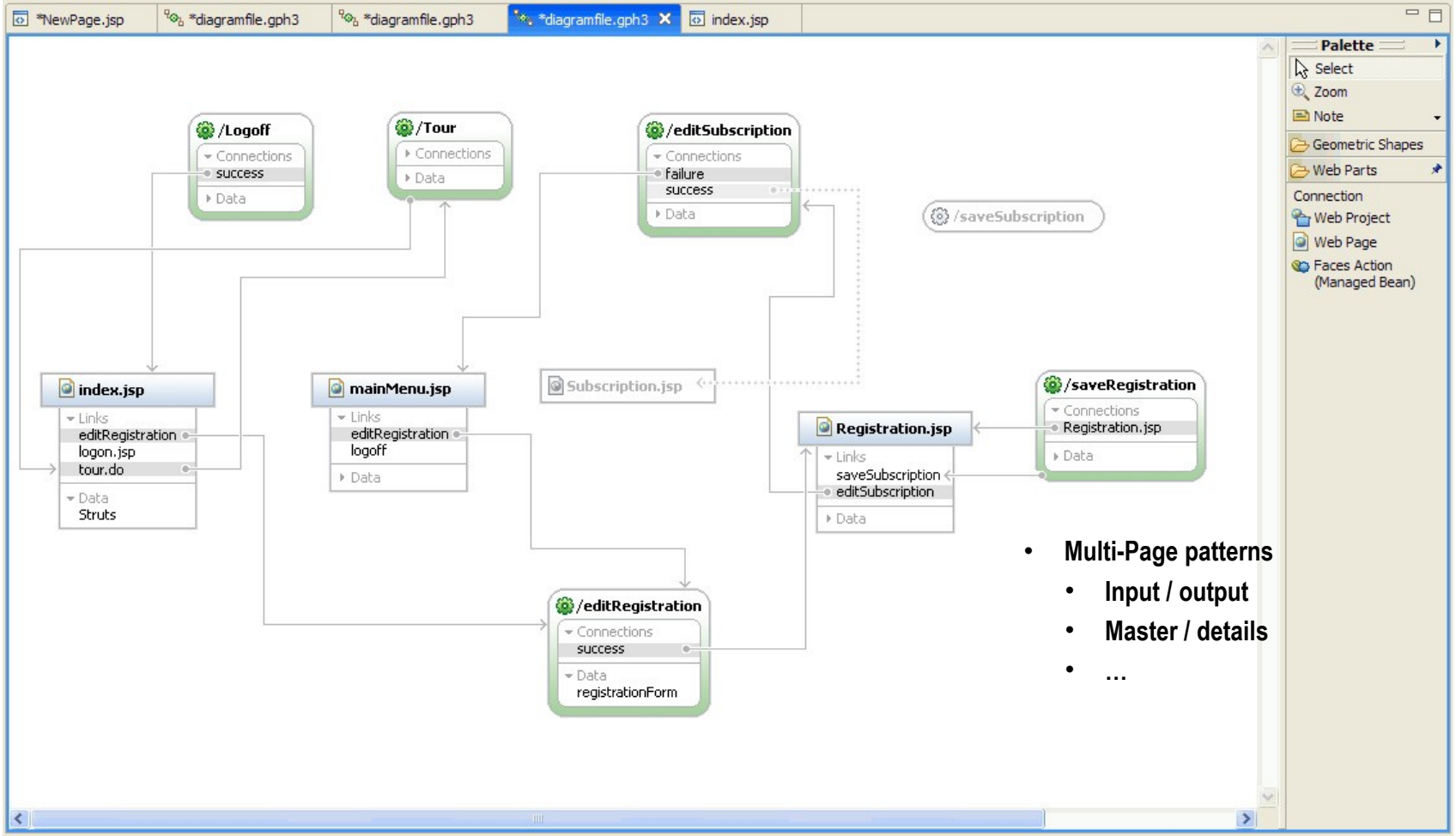
- *Java 5 support (annotations, generics, enums, static import, varargs…)*
- *Operation signature preference (UML*
- *Ability to depict classes in ex*
- *Java developer tools available from diag*
- *Inline editing of java fields and methods (with optional invocation of refactoring)*
- *Many additional 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

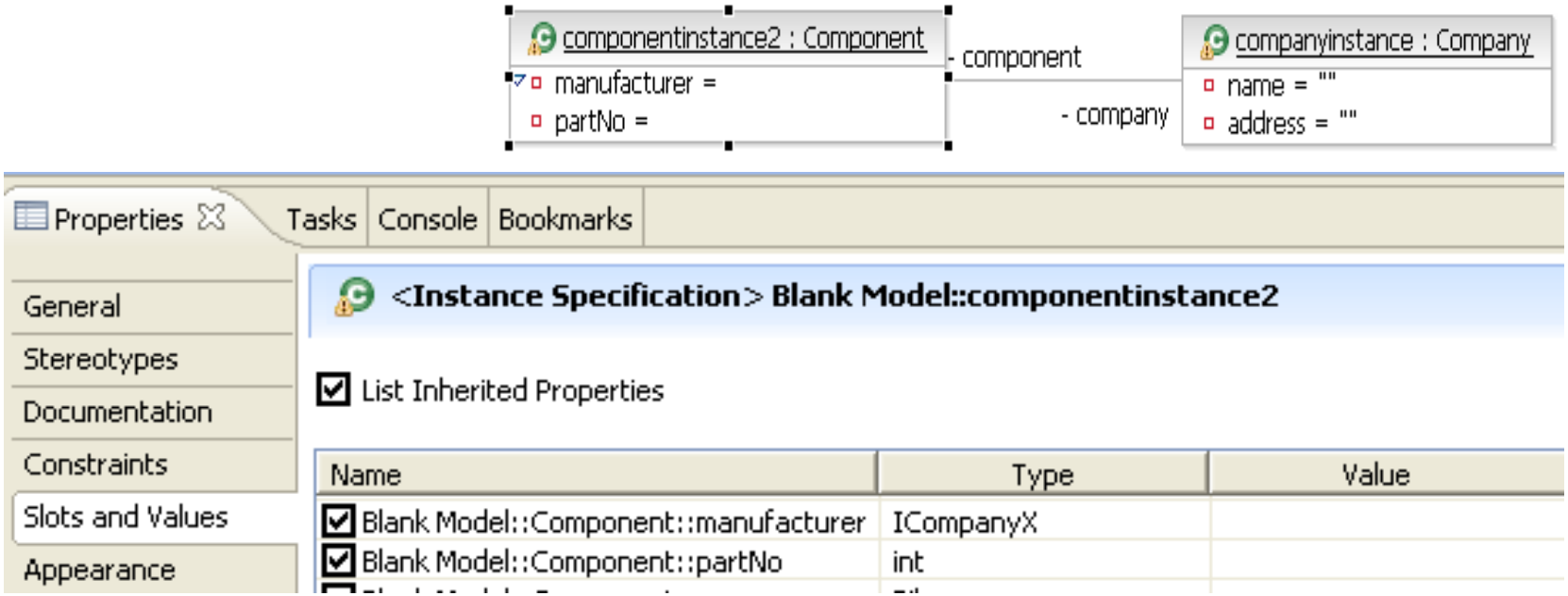
Visualize Data and Service consumption



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

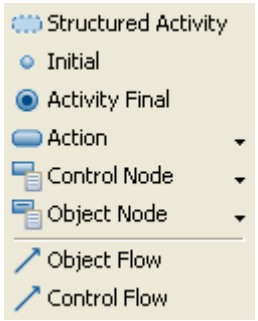
UML Modeling in v7 (1 of 2)

- *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!**

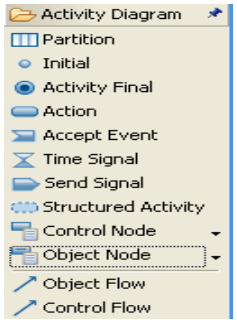


Activity Diagrams

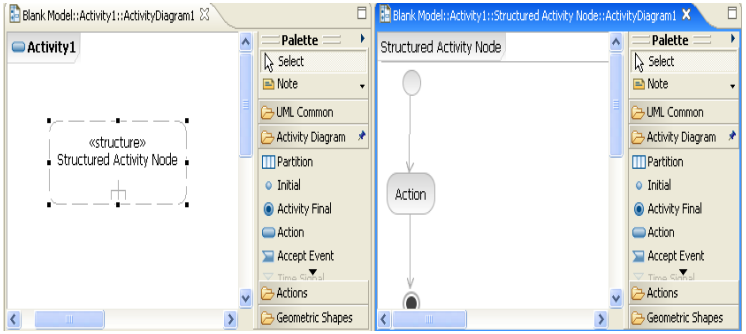
- *Additional element (action) types (~ intermediate level support)*
- *Sub-diagram support on structured activity nodes*
- *Partition shown on activity*
- *Multi-select*
- *Operation shown on call node*



6.0

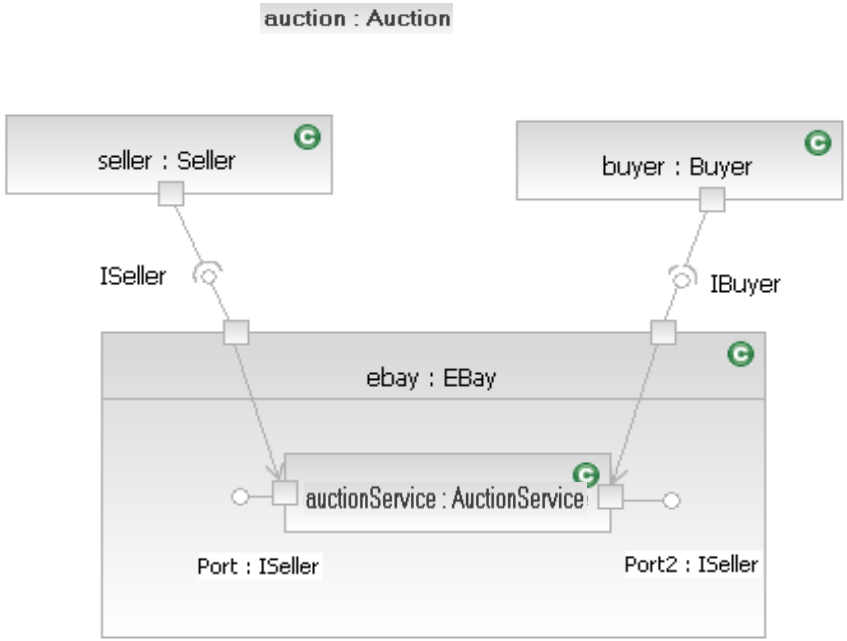


7.0



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**

Team Modeling (UML)

- **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

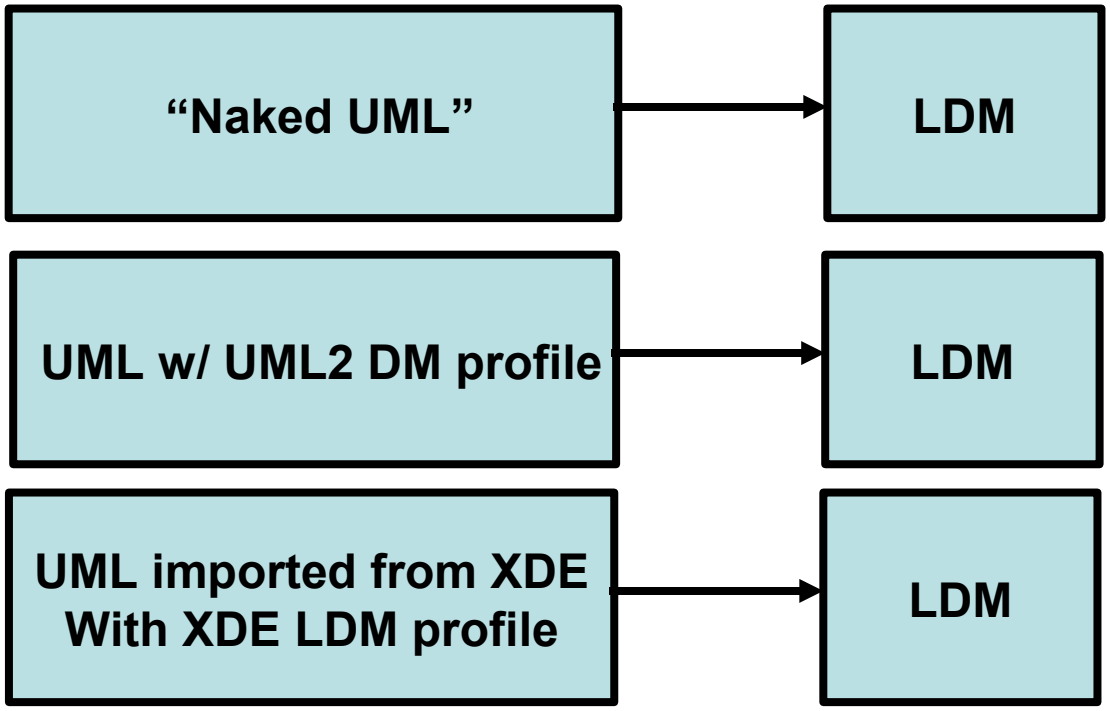
- **Business value**
  - Simply supporting UML modeling and RTE is no longer enough
  - MDD must return more repeatability
  - **MDD must return more automation of repetitive development tasks**
  - **MDD must bridge and integrate domains (business domains, activity domains, problem domains, solution 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

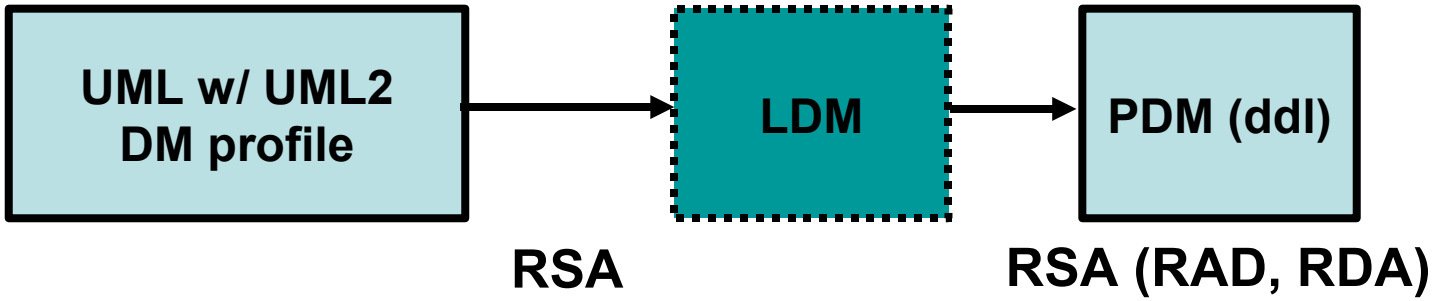
- *WSDL improved and productized*
- *XSD improved and productized*
- *Java updated (Java 5)*
- *Java inverse / reconciliation*
- *C++ updated (fuller support, non-destructive re-apply, inverse / reconciliation)*
- *EGL*

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

Data-Oriented Application Architects

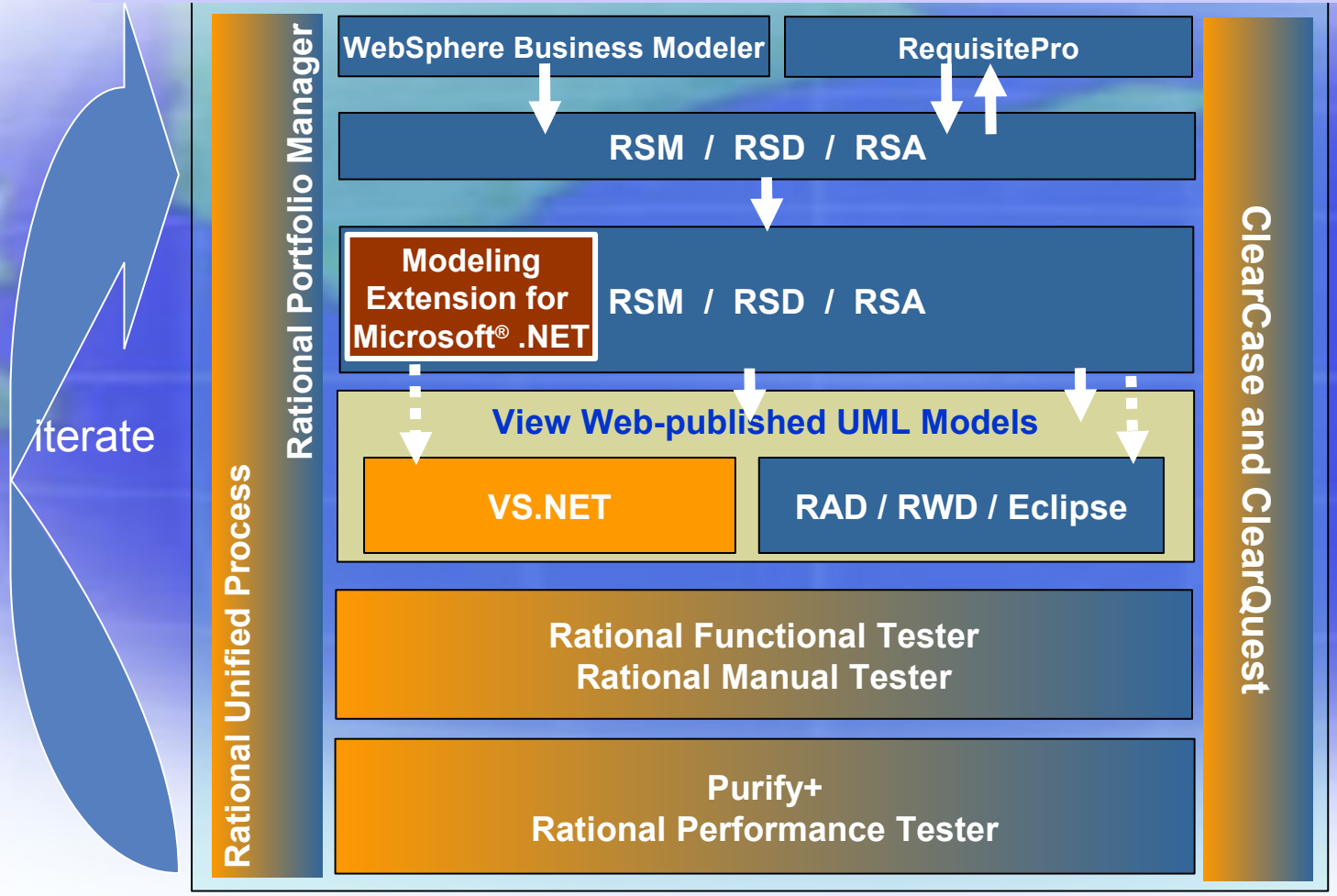


- Transformations
- Shell-sharing by RSx and RDA

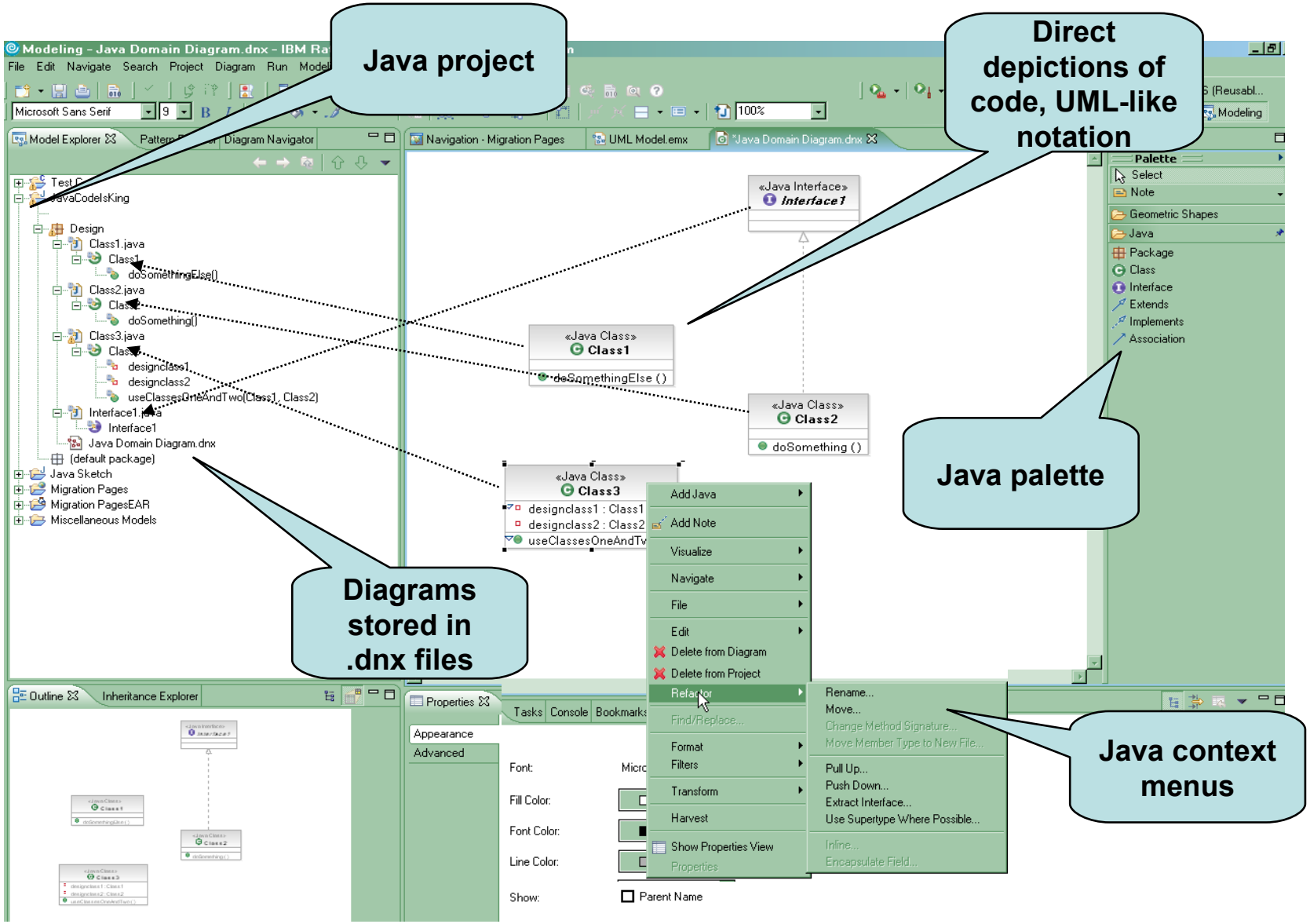


Heterogeneous Development With the IBM Rational SDP

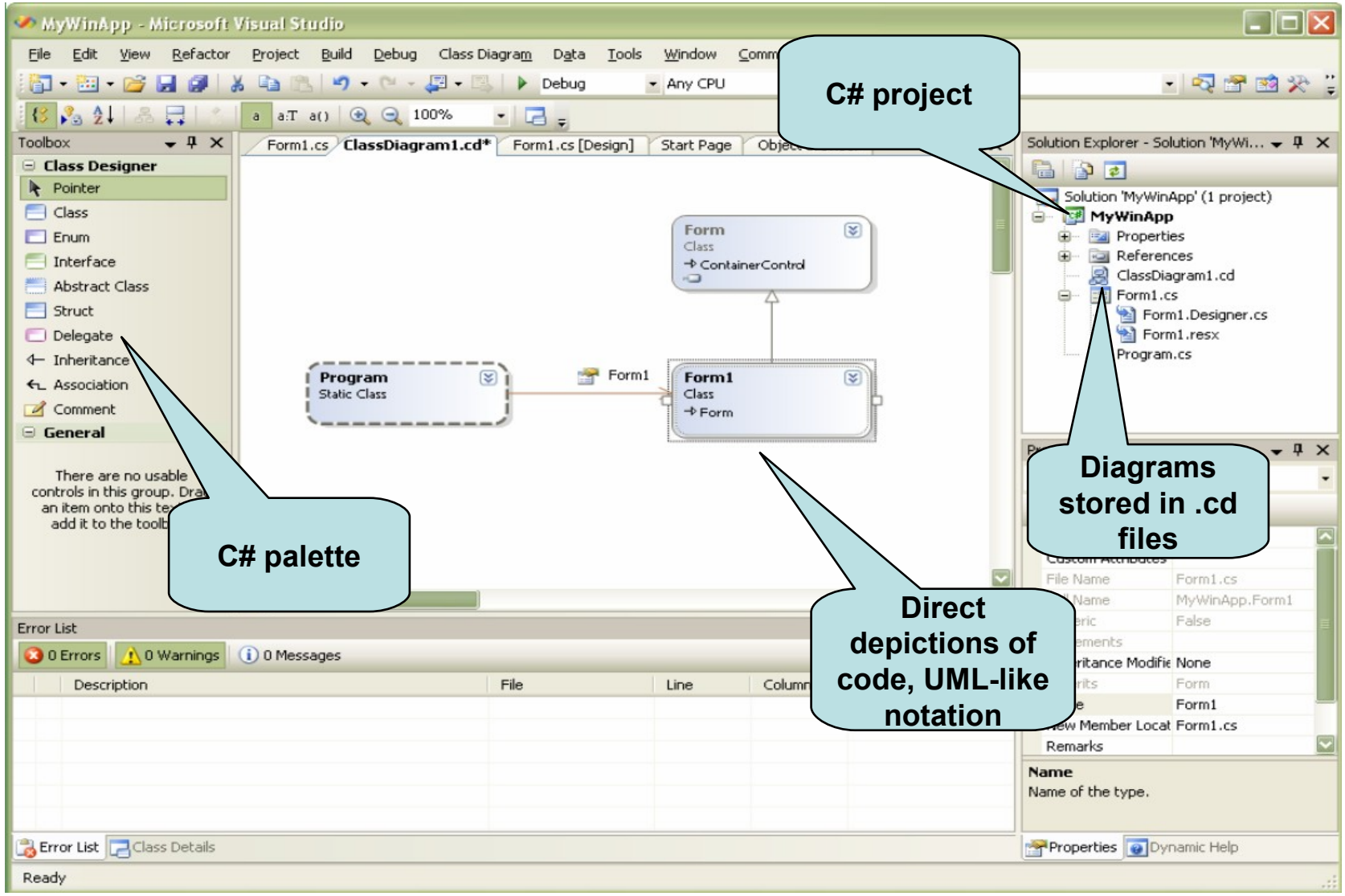
Learn, deploy, and maintain a complete IT governance solution consisting of an open standards-based set of integrated lifecycle tools, with consistent workflows and single-source product support, enabling development that can target a mix of platforms including Enterprise Java and .NET



“Code Model Is King”: Rational Application Developer



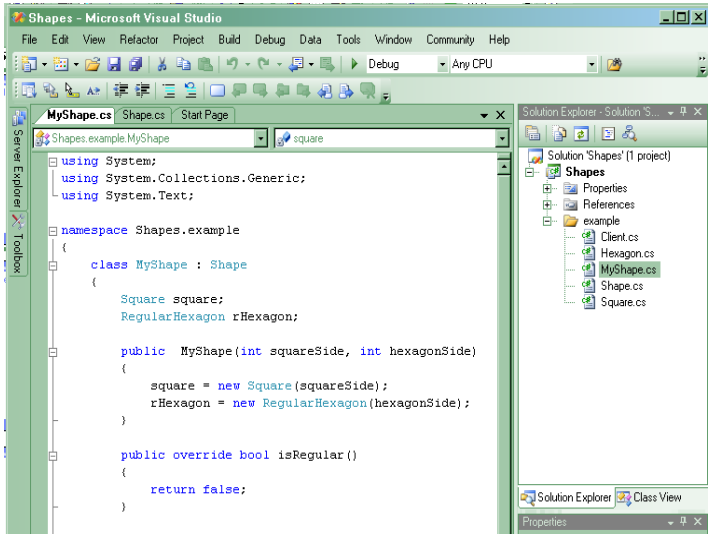
“Code Model Is King”: Visual Studio

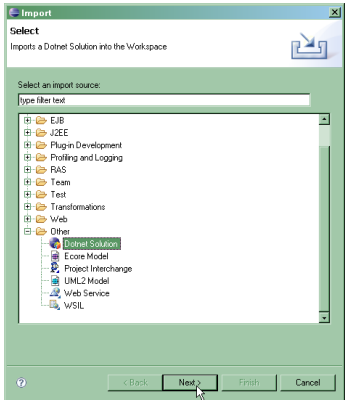


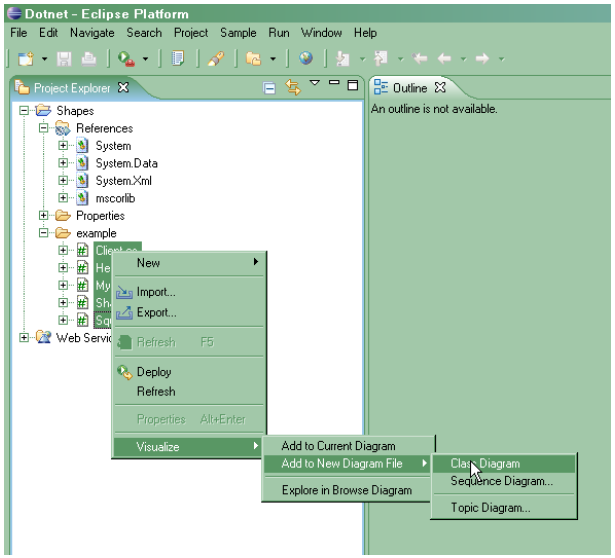
Model Driven Development for Microsoft® .NET

- 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

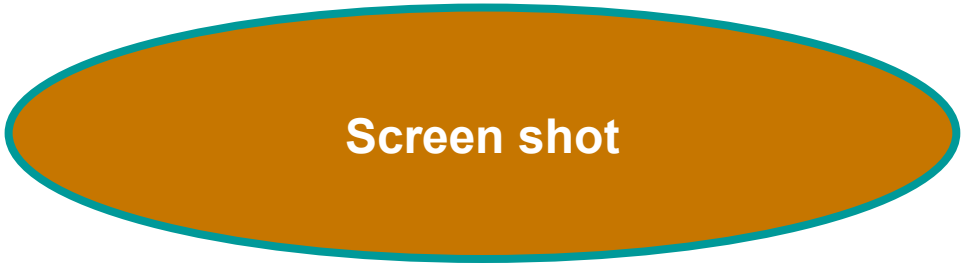
Importing and Visualizing a Solution



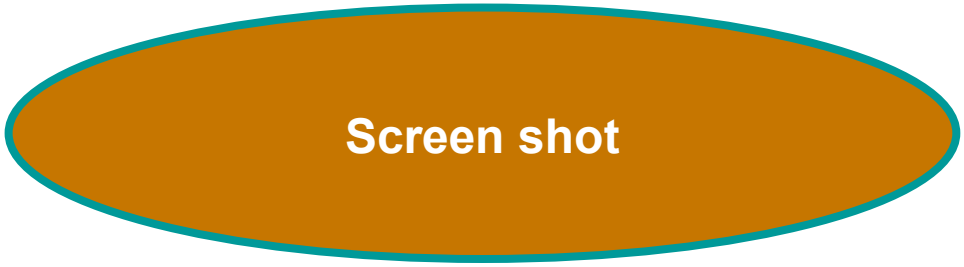




“Mixed Modeling”: RSx with .NET Extension



“True RTE”: RSx with .NET Extension



Demo:

MDD for .NET

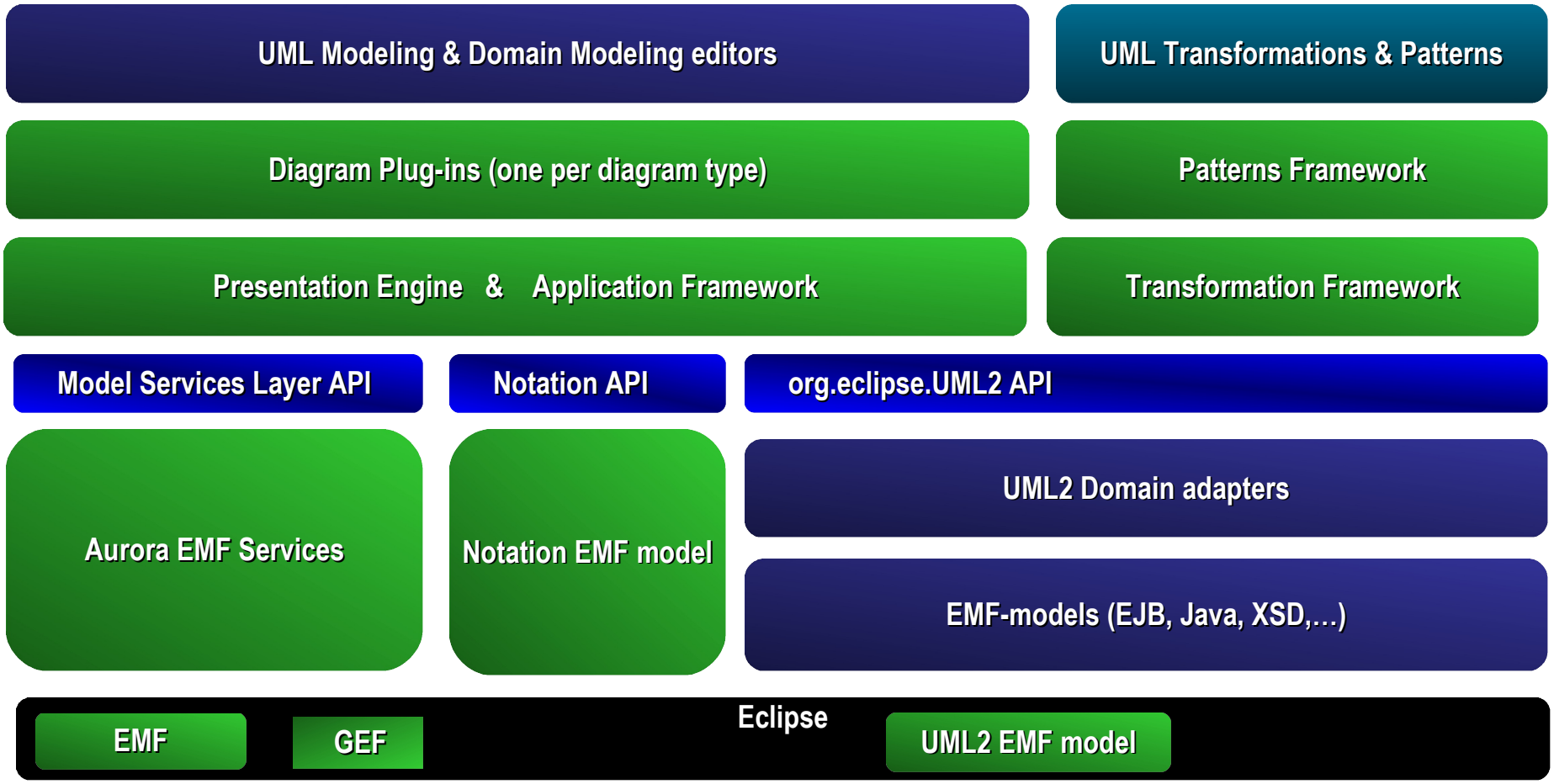
Key Elements of the MDD Vision

- **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

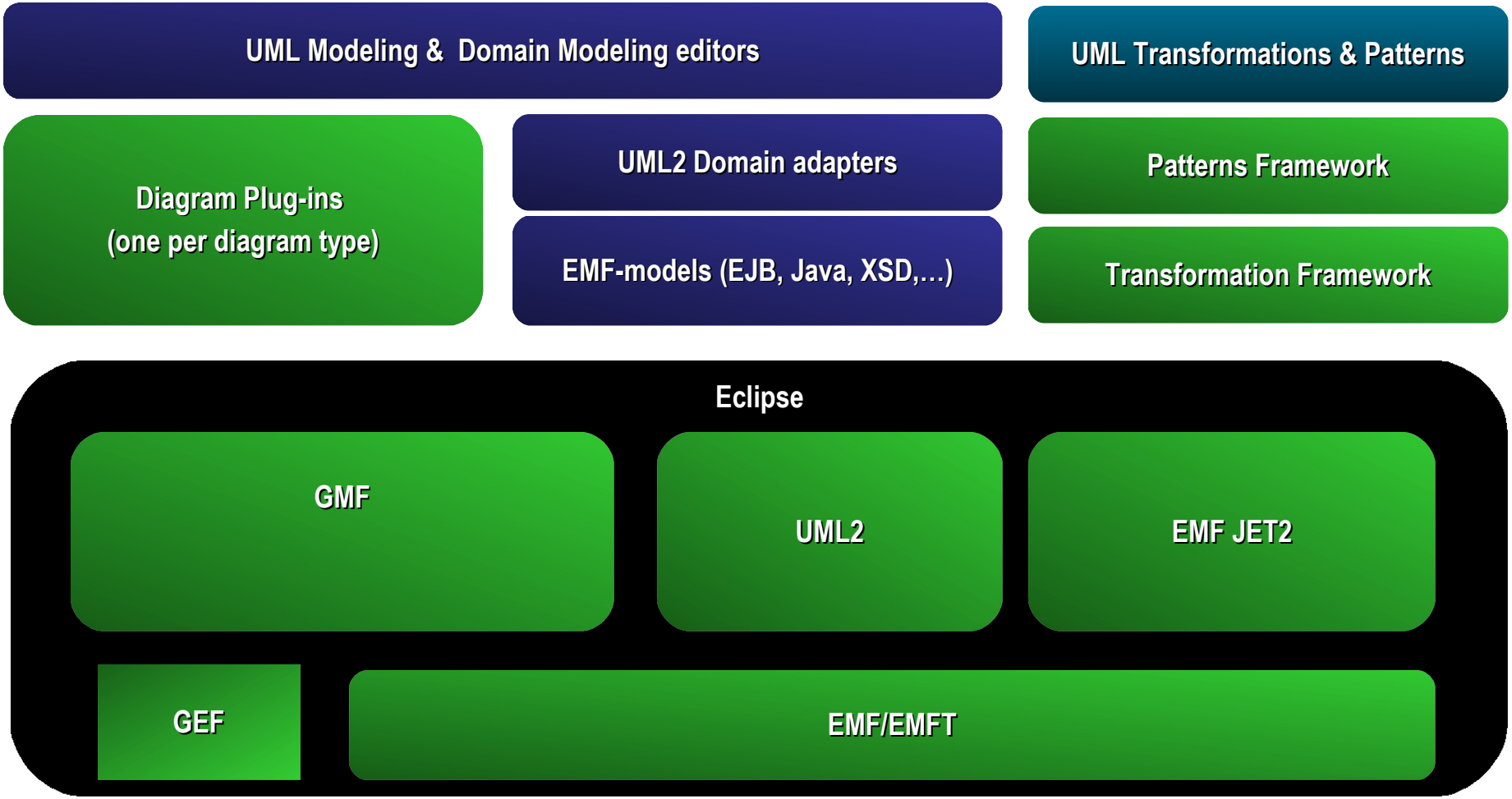
- **Flexibility**
  - Support multiple MDD “theories of operations”, corresponding to multiple development governance philosophies

- **Extensibility**

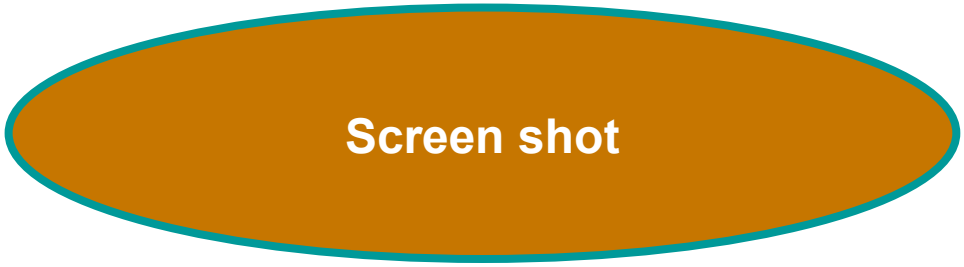
High-level 6.0 architecture



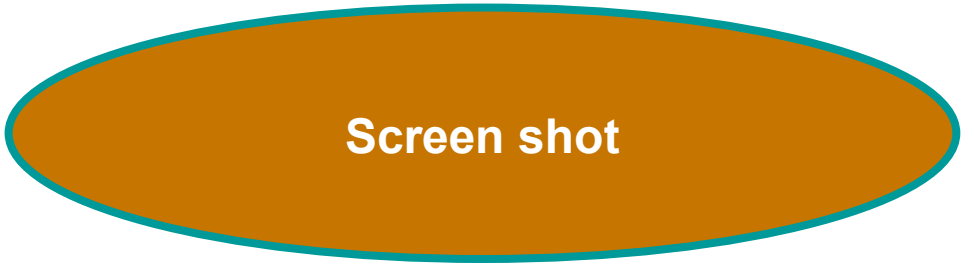
High-level 7.0 architecture



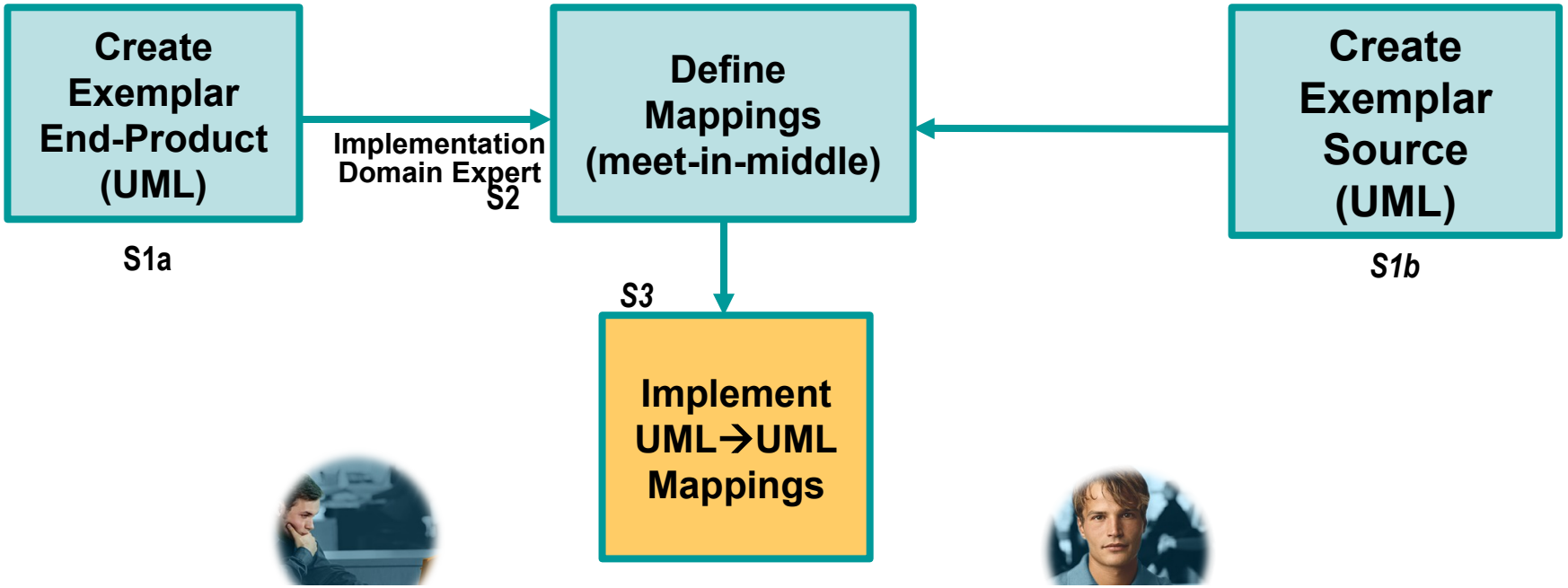
Model-to-Model Transform Authoring



Exemplar-Driven Authoring for JET2 Model-Text (Code)

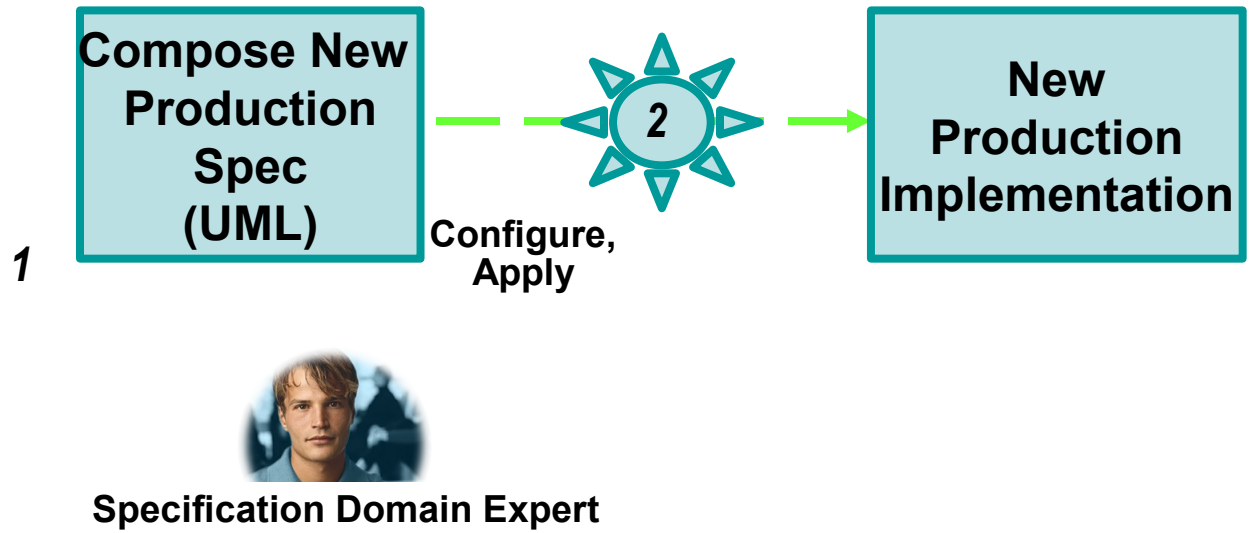


Author UML → UML

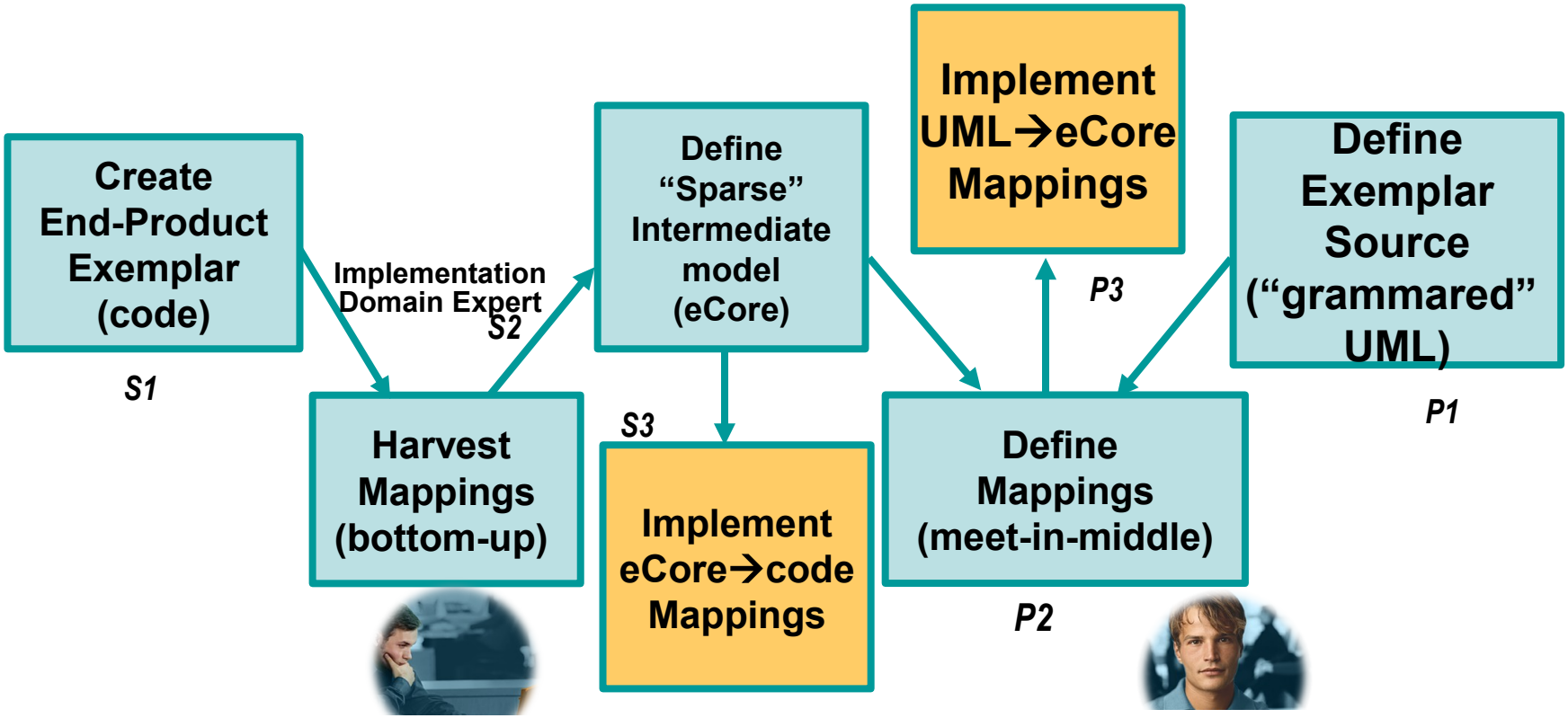


Who	←----- Implementation Domain Expert -----→		←---------- Specification Domain Expert ----------→
Mappings		*Sophisticated (business value)*	
Programming Models		Graphical Mapping Tools, Java	
Domains	*UML*	*UML*	*UML*

Apply UML → UML

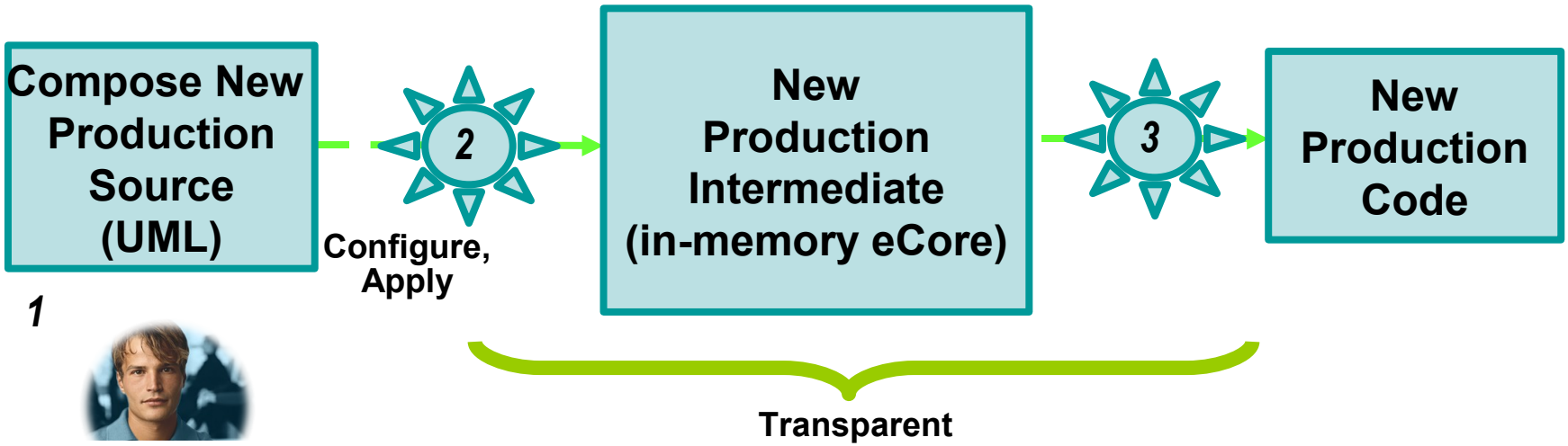


Author UML "Grammared" model → UML "Intermediate" model → code



Who	←-------- Implementation Domain Expert ----------→←---------- Spec. Domain Expert --------→
Mappings	*Simple / 1:1* — *Sophisticated (business value)*
Programming Models	Exemplar Authoring, JET2, Java — Graphical Mapping Tools, Java "light"
Domains	• Code — *eCore / XML Java* — *eCore, UML, Java* — *UML*

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



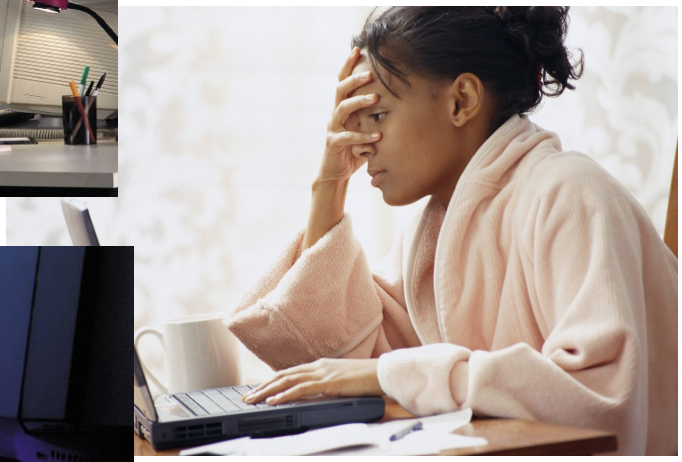
Agenda

- *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"

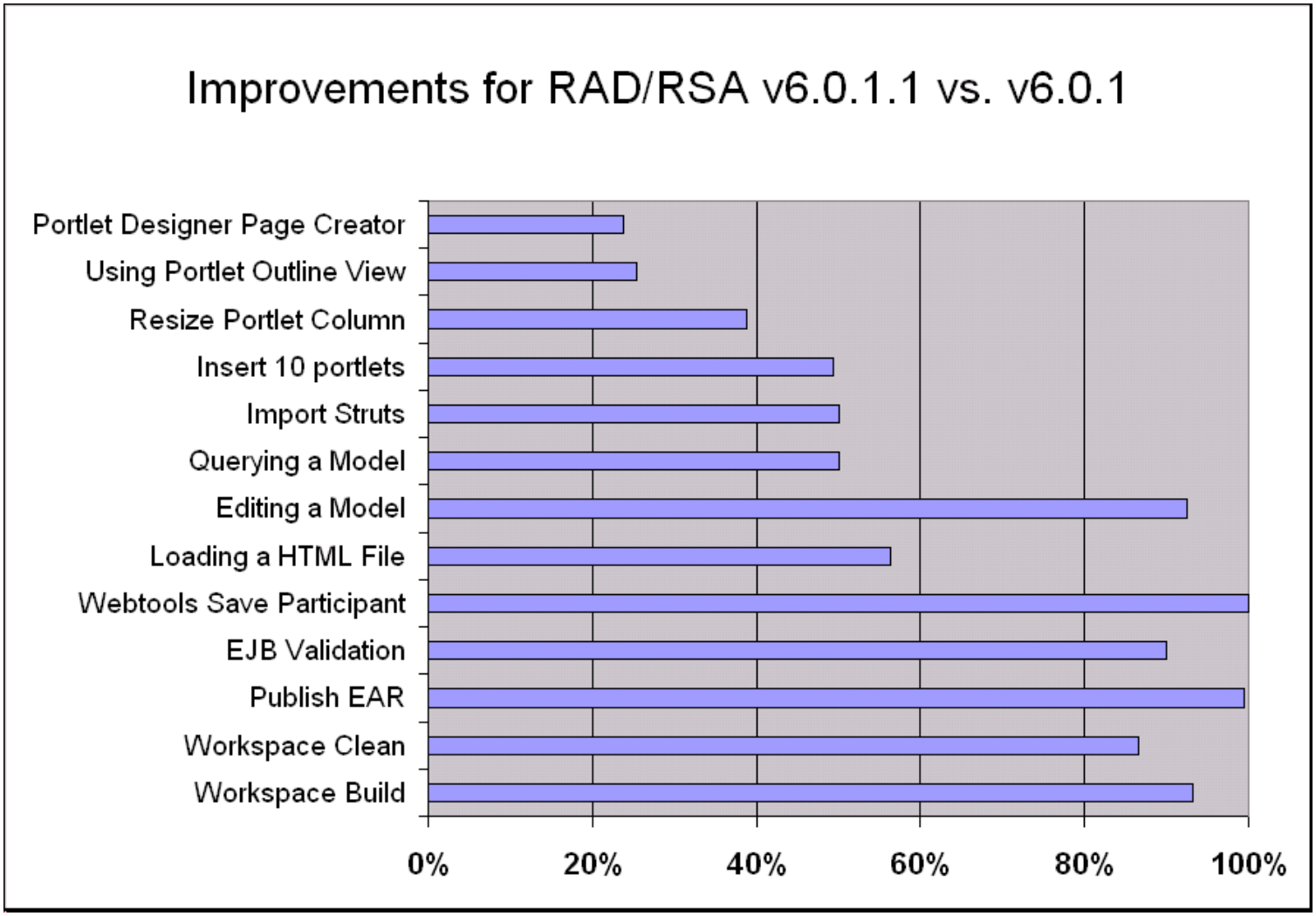
"Work in my existing Eclipse environment"

"Needs too much RAM"

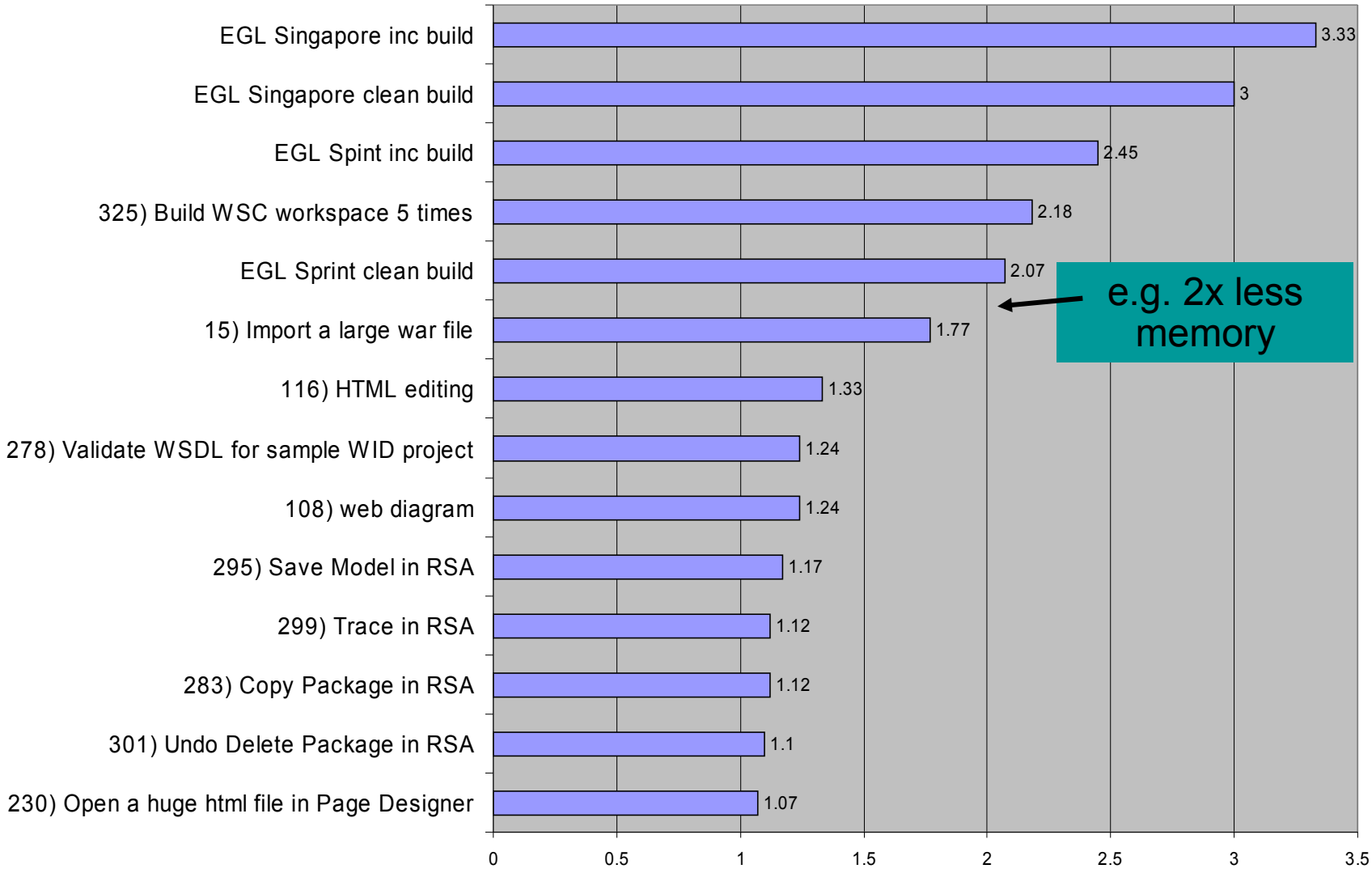
"Smaller on-disk footprint"

"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)

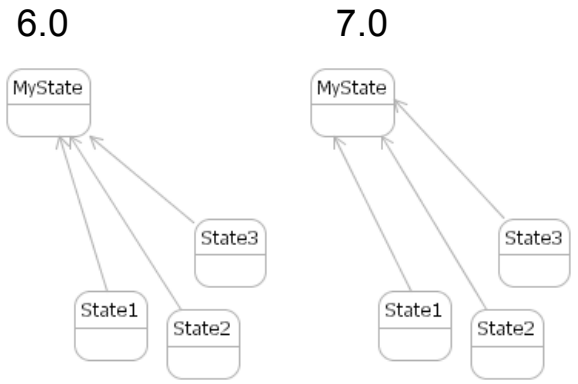


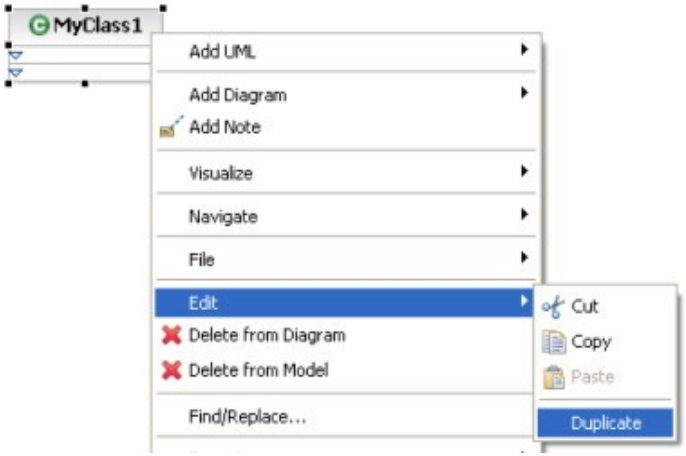
Performance Improvements in V6.0.1.1 (Space)

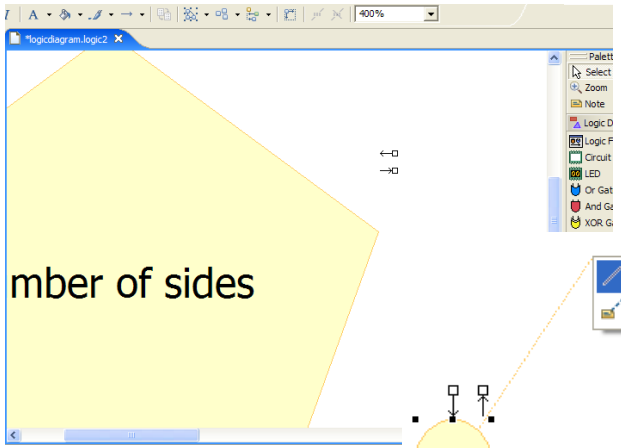


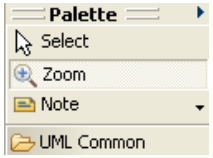
General Modeling Usability

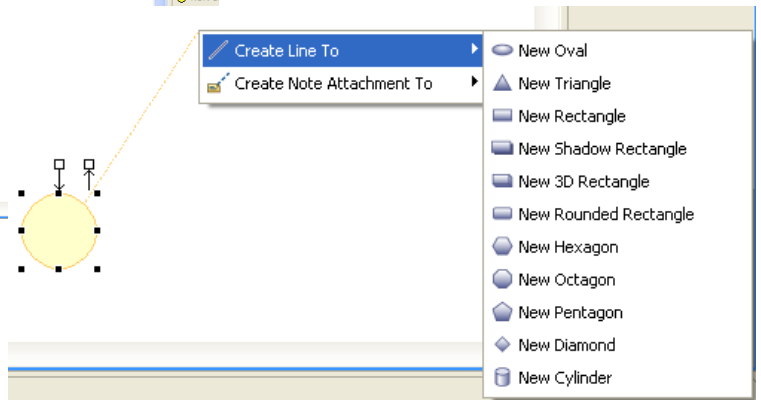
- *Improved relationship anchor support*
- *"Change Metatype" refactoring action*
- *Zoom tool, animated zoom, animated arrange*
- *"Duplicate element" action when drawing*
- *Connector assistants for Notes and Geo Shapes*



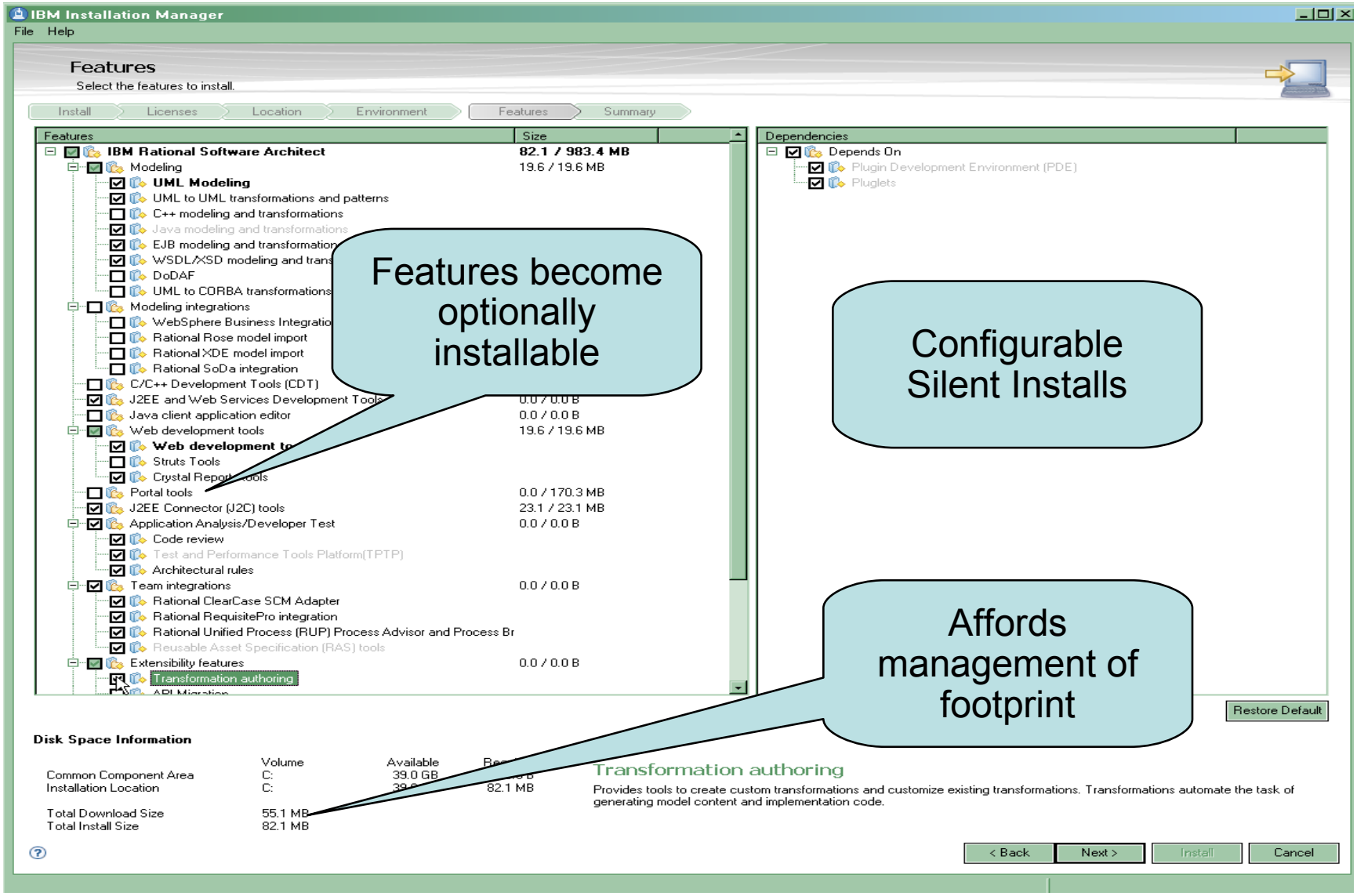




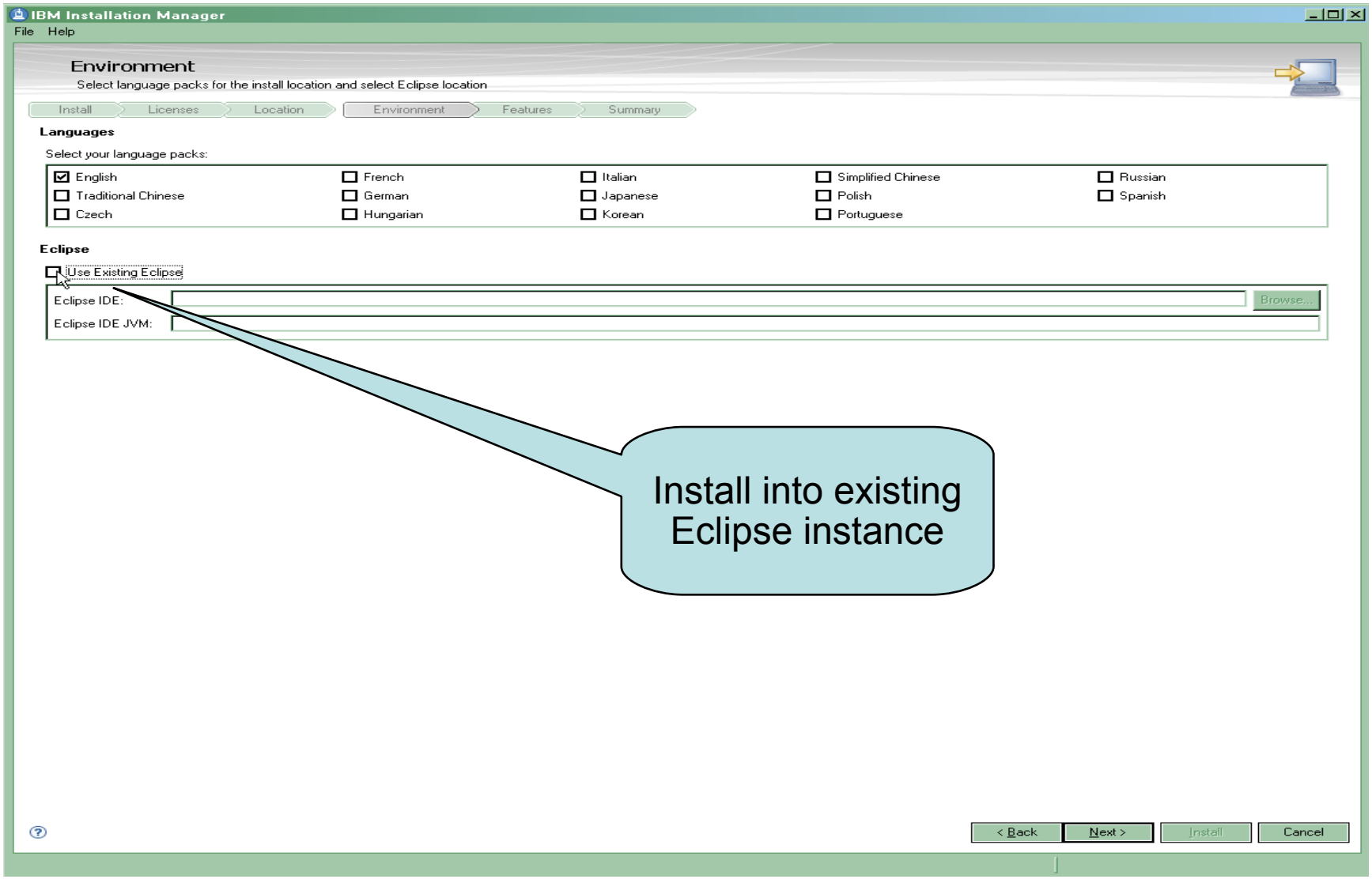




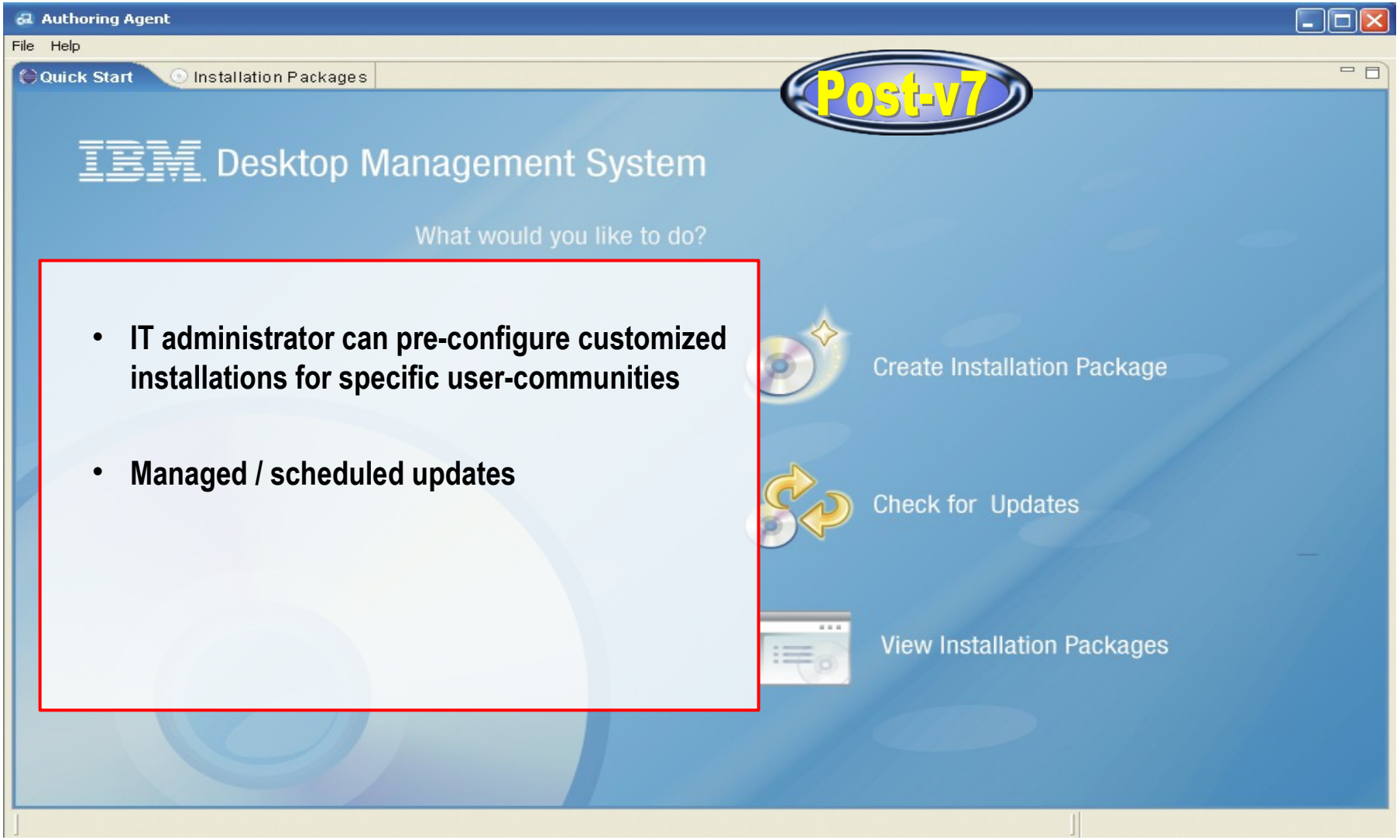
Product Installation and Deployment (1)



Product Installation and Deployment (2)

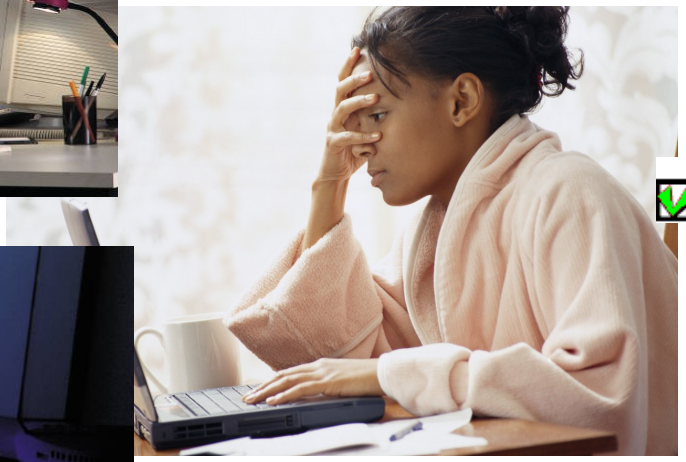


Product Installation and Deployment (3)



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

We Are Listening



"Improve performance"

"Automate product deployment"

Post-v7

"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