Rational. software

Team Development and Model Management with IBM Rational Software Architect

Kim Letkeman
Senior Manager, Rational Model Driven Development Platform
kletkema@ca.ibm.com

IBM Rational Software Development Conference UK 2007

What keeps me **Rational**?

AM7

IBM

Agenda

- Introduction
- Team Development
- Model Driven Development
- Enterprise Model Management and Partitioning
- Demo

Agenda

- Introduction
- Team Development
- Model Driven Development
- Enterprise Model Management and Partitioning
- Demo

Why Manage your Models?

- Models are key corporate assets from which products and systems are derived.
  - *These should be managed such that they are stored safely and reliably and that every important baseline remains accessible throughout the derived product's legally mandated maintenance period.*

- Models are constructed and manipulated in distributed teams, each modeler working on a partial view of the overall system.
  - *This parallel effort should be managed and coordinated to maintain model consistency and integrity, mitigating risk when parallel versions collide upon check-in.*

Goals for RSA Model Management

- For RSA 6, our goal was team development – ClearCase and CVS integrations specifically
  - We had to be able to merge multiple generations of the same model when parallel check-ins were detected
  - We concentrated on merging by element identity, which allows detection of move and reorder changes in addition to the usual additions, changes and deletions
- For RSA 7, our goals expanded to include the work flows inherent in Model Driven Development
  - We added structural merging (fusion) for MDD transformations
  - Fusion also addresses the need to transition smoothly from ad-hoc modeling to formal version-controlled modeling

*Note: the recommended version of RSx for team and model driven development is **7.0.0.3** or above*

Terminology

- **Repository** – permanent storage for artifacts, e.g. models and code
  - For work in progress within a distributed team, a controlled-access, version-able repository is essential. ClearCase is an enterprise-class solution. CVS is a solution for smaller groups.
  - For enterprise-wide governance and sharing with query-able metadata, RAM is an enterprise-class solution.
  - This discussion concentrates on the former set of use cases.
- **Compare and Merge** – technology that allows two or three models to be compared by *element identity*, detecting all delta types (add, change delete, move, reorder), and that allows these changes to be accepted or rejected.
- **Fusion** – technology that allows one model to be combined into another by structure … that is by *qualified name* … fusion is *identity-aware* as of version 7.0.0.4 and will first align elements by ID if present

Terminology

- **Identity** – a unique moniker for an element inside a model … an element ID *must* be unique within all models that may interact within an organization … the ID for an element is *permanent* and *immutable*
- **Name** – a moniker by which an element is generally known … an element name is not generally required to be unique, although UML validation will fail with certain duplicate names
- **Qualified Name** – the full path to an element by the names of all of its ancestors and itself
- **Delta / Difference** – any change between two elements that have the same identity (compare) or qualified name (fusion) in two different versions of the model … differences are directional – left minus right might produce an add delta where right minus left would produce a delete delta

Agenda

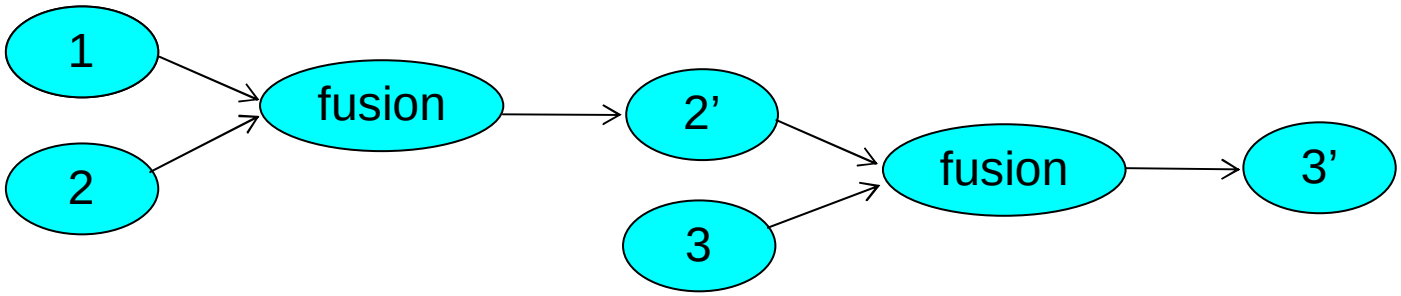
- Introduction
- Team Development
- Model Driven Development
- Enterprise Model Management and Partitioning
- Demo

Team Development

- When developers or modelers collaborate towards a common goal, sharing their work product or artifacts and modifying them *in parallel,* we call this *team development* … the key concept here being parallel changes
- Productivity suffers when the coordination required to protect the integrity of these assets impedes the smooth flow of changes into the source files
- Tooling should protect the artifacts from accidental corruption or deletion while providing automated coordination of parallel development, thus relieving the team from this burden and increasing productivity
- RSA directly supports team development through ClearCase and CVS
- RSA 7 added features to directly support ad hoc modeling during the transition to more formal team development practices

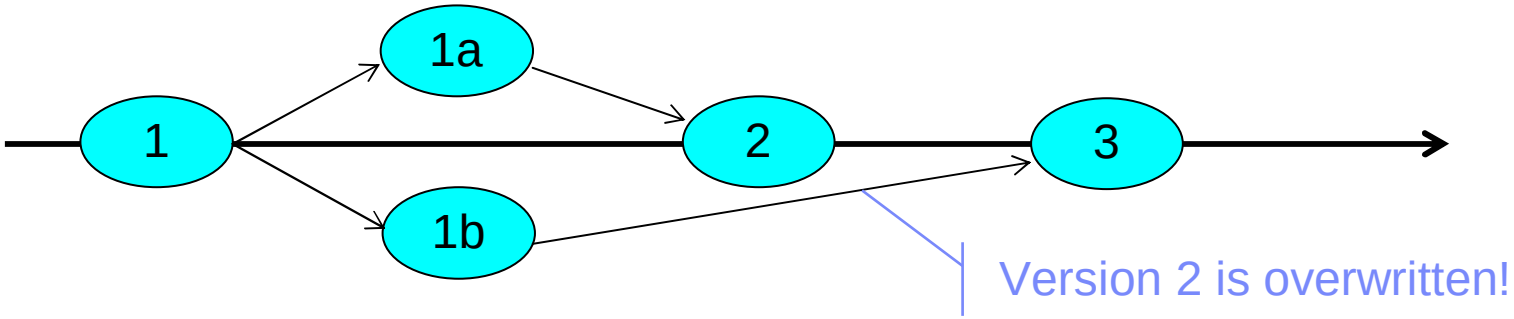
Use Case – Ad Hoc Modeling

- During the early stages of a project, it is not uncommon for several modelers to create similar models that must eventually be combined to form an enterprise model going forward
- Chief characteristic is the lack of any formal version control and models that were created independently and *do not* share ancestry
- **High risk** of model corruption and confusion, very difficult task if done manually using drag and drop
- RSA 7.0.0.3 has first-class support for this scenario



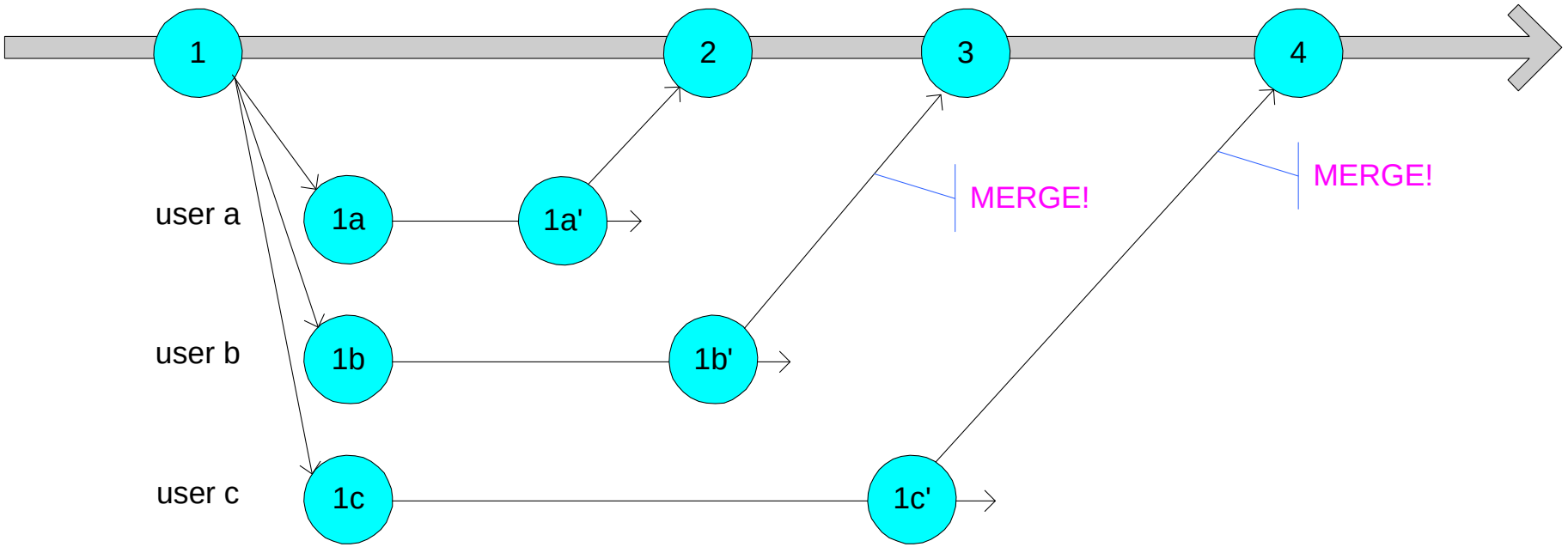
Use Case – Informal Version Control

- Modelers sometimes collaborate to build a software system with the files stored in a folder system. Copies are sent to practitioners, which means that models share ancestry and can thus use merge instead of fusion. One person is designated to periodically merge the work manually.
- Chief characteristic is *parallel development* without direct tool support
- **High risk** of accidental erasure of artifacts if any procedural mistakes are made



Use Case – Formal Version Control

- ClearCase and CVS automatically detect the need for a merge!
- Typical behavior in mature projects

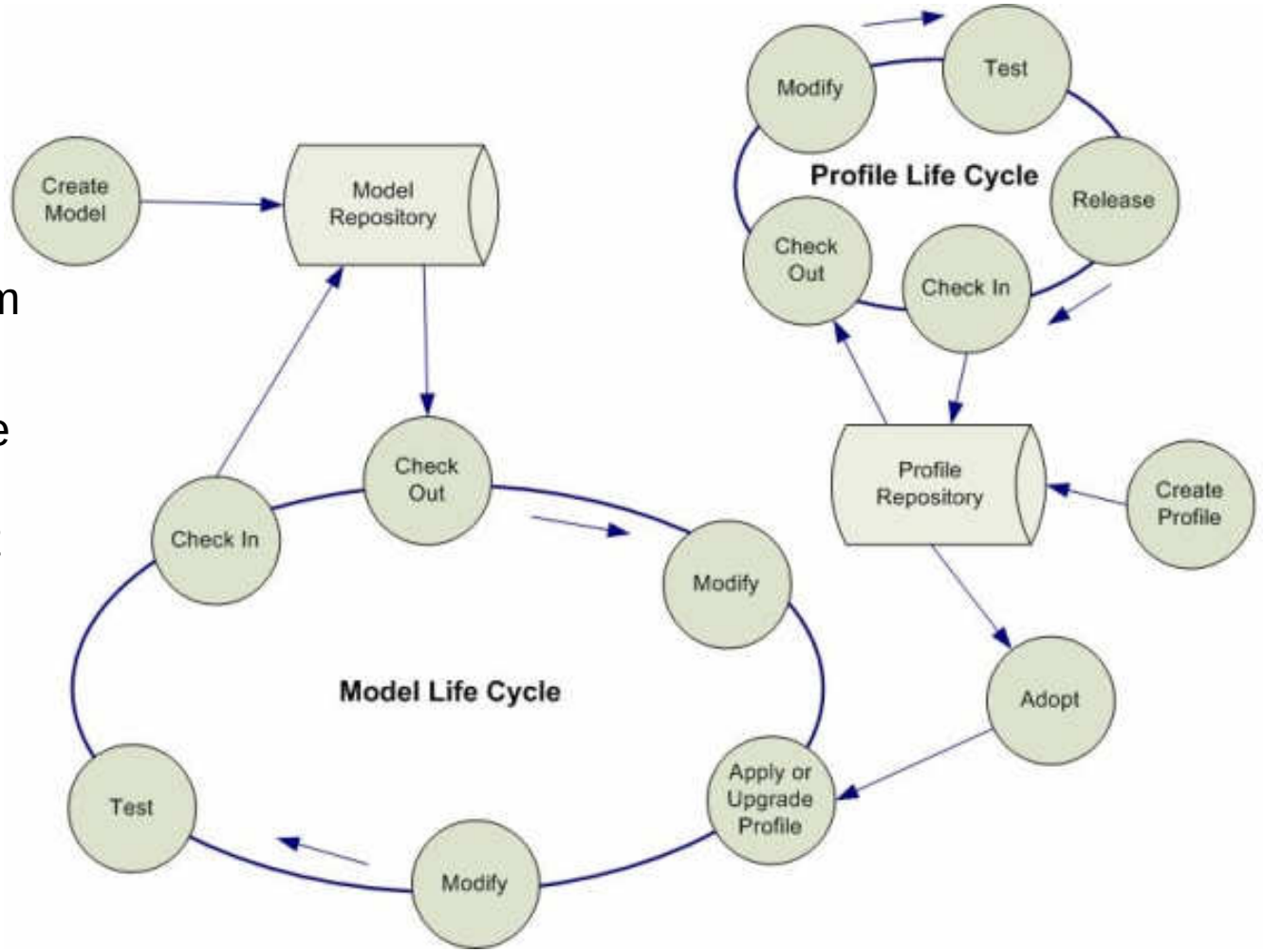


Use Case – Model Management with Custom Profiles

- Extending the UML meta-model with a custom profile is a common way to adapt a model to a specific domain
- A custom profile *changes a model's meta-model once applied;* mistakes in backward compatibility can be fatal to previous generations of a model
- A custom profile's management life cycle should be separated from the model's with access to the design-time profile limited to those with meta-model expertise
- Timing of profile migrations is up to individuals, which can create a merge session with different versions of the meta-model being compared
- Since different versions of a meta-model cannot be compared, RSA migrates all in-memory models to the latest version of the custom profile (meta-model) automatically, allowing the merge to be completed

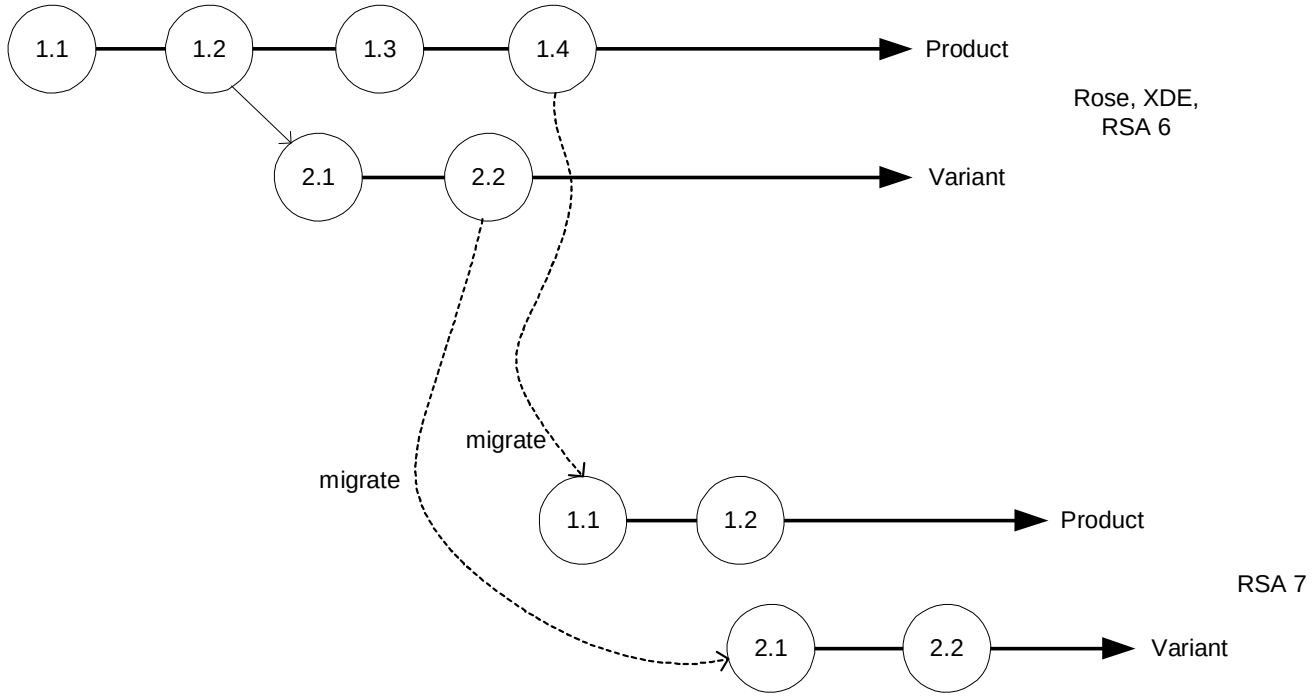
Custom Profile and Model Life Cycle Interaction

- It is *very important* that these life-cycles be separated.
- Profiles should be stored separately from models.
- Failure to follow these guidelines can result in models that cannot be opened.
- More information is available in Part 6 of the compare merge article series.



Use Case – Multi-stream Model Upgrades

- A model or a set of models exist in multiple variants, so that derivative products can be modeled and maintained
- All of these models are upgraded at some point, from Rose or XDE to RSA, or from RSA 6 to RSA 7

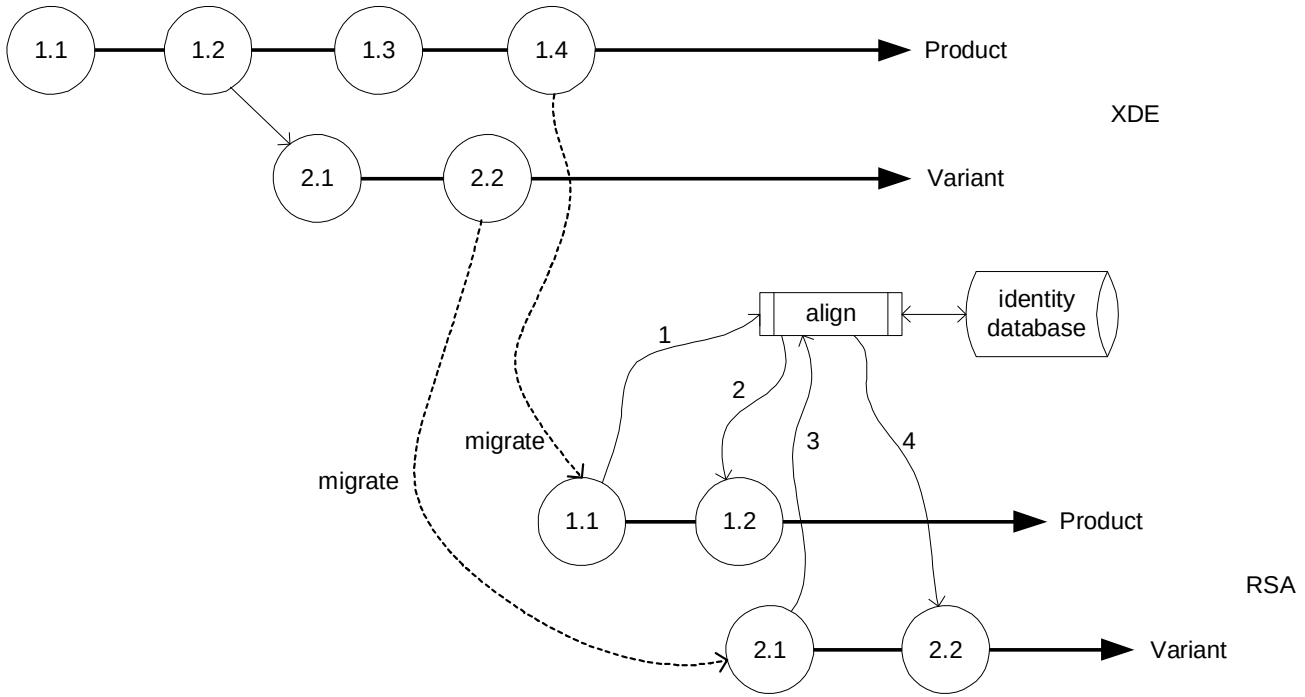


Use Case – Multi-stream Model Upgrades

- Upgrading from XDE or Rose to RSA, or from RSA 6 to RSA 7 changes UML and notational meta-models
- These meta-model changes require the creation of many new elements that did not exist in previous UML generations – we've seen 10,000 per model
- These new elements have new identifiers and the same new elements are created in each stream; each have a new unique identifier – *but these will not match during a compare!*
- RSA 7 includes technology to realign identities in a series of models, effectively recreating the element ancestry relationships from stream to stream ad infinitum
- After running the model alignment tool between two such streams, any two models will have the same identifiers for any element that is intended to be the same, thus superfluous differences are removed

Use Case – Multi-stream Model Upgrades

- The model alignment tool workflow looks like:



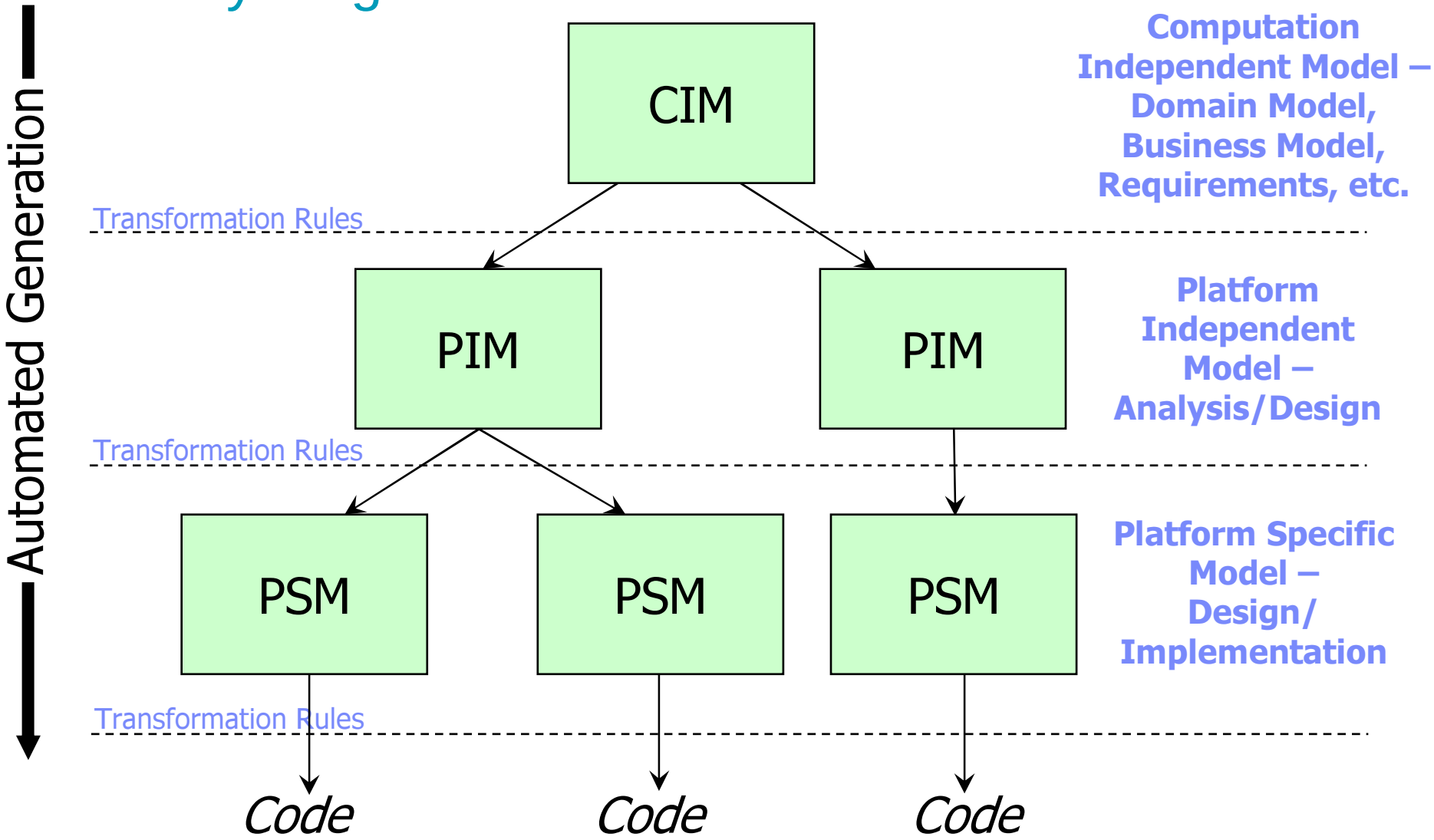
Agenda

- Introduction
- Team Development
- Model Driven Development
- Enterprise Model Management and Partitioning
- Demo

Model-Driven Architecture

- A modeling approach whereby the system is specified in a platform independent way, and thus is separated from the idiosyncrasies of any specific implementation platform.
- Most importantly, a language and vendor neutral architecture is created.
- MDA is formally defined by the OMG.
- We refer to the use of models as first-class artifacts to generate solutions as *Model-Driven Development*, or *MDD*

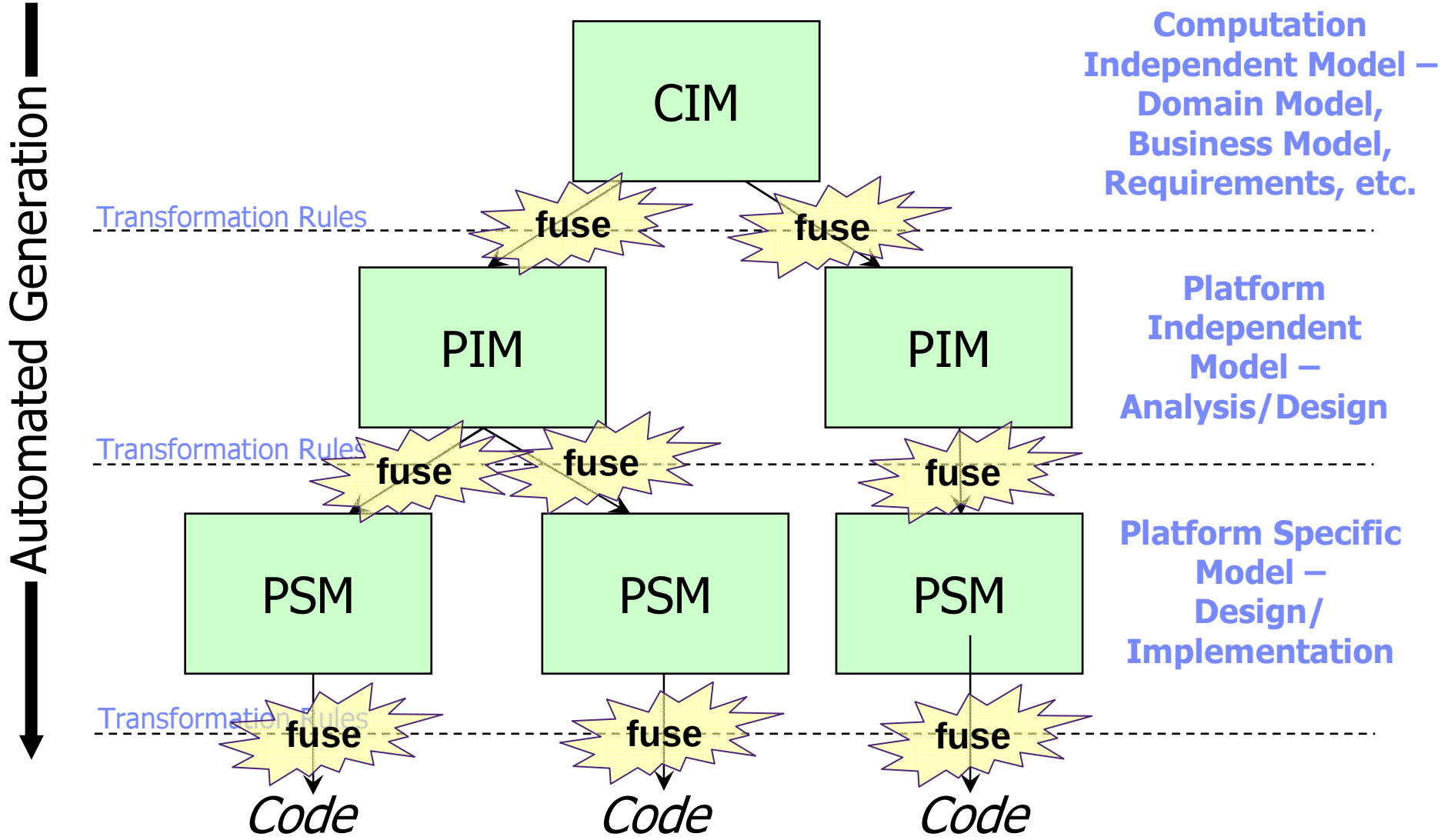
MDA Layering



Model Driven Team Development

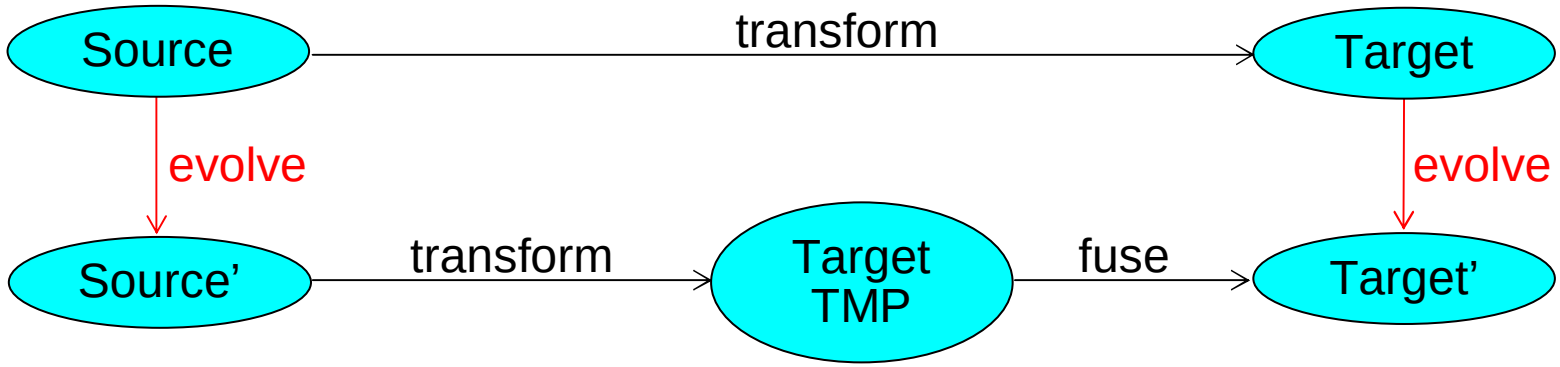
- Each transition between layers in the MDA stack is a candidate for MDD with a transformation
- The more abstract layer is designated the source model for a transformation, while the more concrete layer is designated the target model
- When a transformation runs, it may replace the target model …
  - … but the preferred approach is to fuse the generated intermediate model into the target model, thus capturing structural changes while preserving any existing evolution of the target model

MDD with Fusion



Fusion at the Transformation Target

- At every transformation stage, you can iterate. Change the source model, and you must transform again.
- But the target model may already have custom content from its own iteration or evolution.
- There is therefore a **fusion** step at the end of each transformation step, which must preserve the custom content in the transformation target

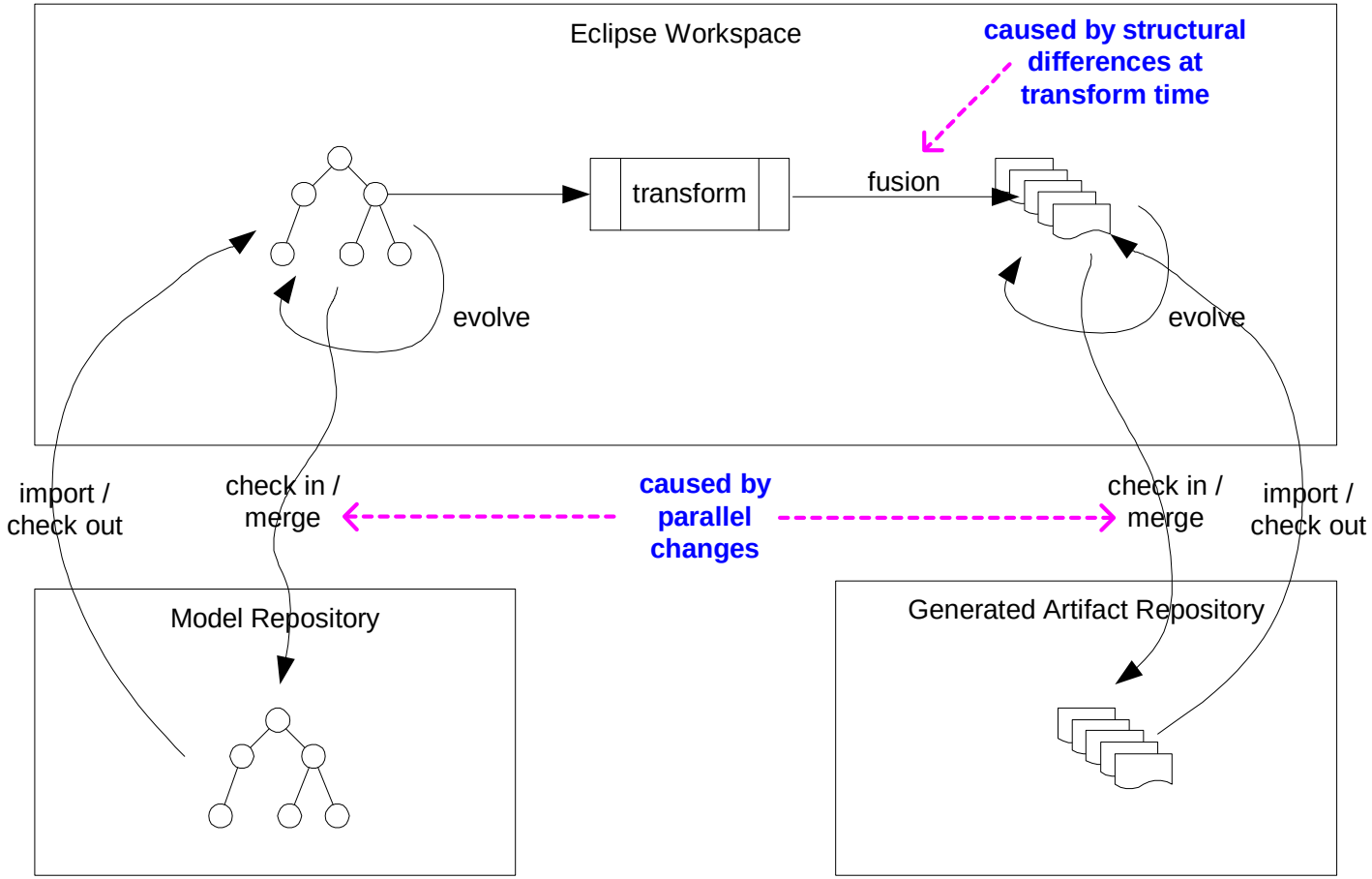


Model Driven Team Development

- Transformation is a form of evolution with both source and target models in a workspace
- The target model must be writable – e.g. checked out of ClearCase
  - Note that ClearCase will automatically check out any target files it needs to, so one can simply run the transformation and let Eclipse and ClearCase handle the check-outs
- Once the transformation is completed, target files can be checked in again with a common comment or description, in ClearCase UCM they will also be members of one common activity whose description indicates that a transformation took place

Model Driven Team Development

- *Artifact flow:*



Agenda

- Introduction
- Team Development
- Model Driven Development
- Enterprise Model Management and Partitioning
- Demo

Enterprise Model Organization

- Development teams within an enterprise can be:
  - Isolated from one another
  - Tightly structured, perhaps in a hierarchy
  - Loosely structured, perhaps in a mesh
  - … any combination of the above
- Model structure can be influenced by it team structure – for example, separation of component teams into silos may force the same organization for component models
- The derived system's organization will have to align with the team organization (or vice versa), and corporate governance requirements may directly drive team and therefore repository and model structure
- It is useful to thoroughly understand team and system architecture and organization before creating model repositories and partitioning models

Repositories with First-Class Integration to RSA

- CVS
  - Very good for small, isolated and distributed teams
  - Becomes more difficult with heavy use of hierarchical streams and branches
- ClearCase
  - LT version can handle small teams
  - Enterprise class servers with multi-site support for globally distributed development
  - CCRC for WAN based model management
  - Extremely flexible and powerful for multi-stream branching and merging
  - Surprisingly easy to use for practitioners performing day-to-day branching and merging

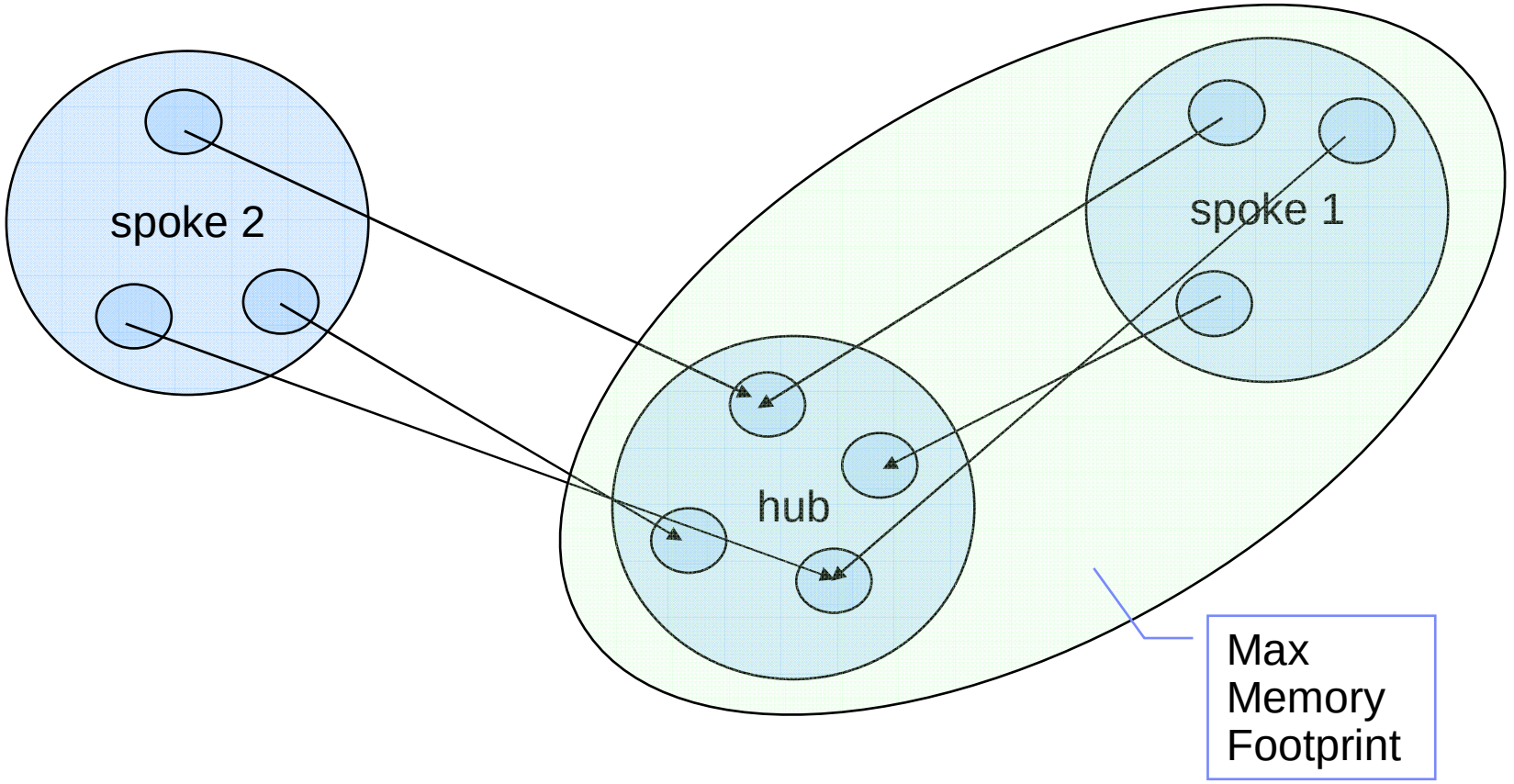
Model Partitioning

- Models need to be sized to the workstations used to manipulate them. Where 512MB RAM is the norm, models needs to maintain a very small memory footprint, as there is very little real memory available for the application
- There are several partitioning options that can reduce memory footprint, but note that it is always best to increase RAM in workstations if at all possible … at today's people costs, the break-even ROI for these upgrades is very short … days or weeks.
- Models can be *partitioned*, where one model becomes several, each referencing elements inside one or more other models.
- Models can be *fragmented*, where a model remains intact at the logical level, but is physically broken into many small chunks.
- Each has pros and cons, and each should be used appropriately to avoid model corruption or poor performance.

Partitioning Option 1 – Separate Models

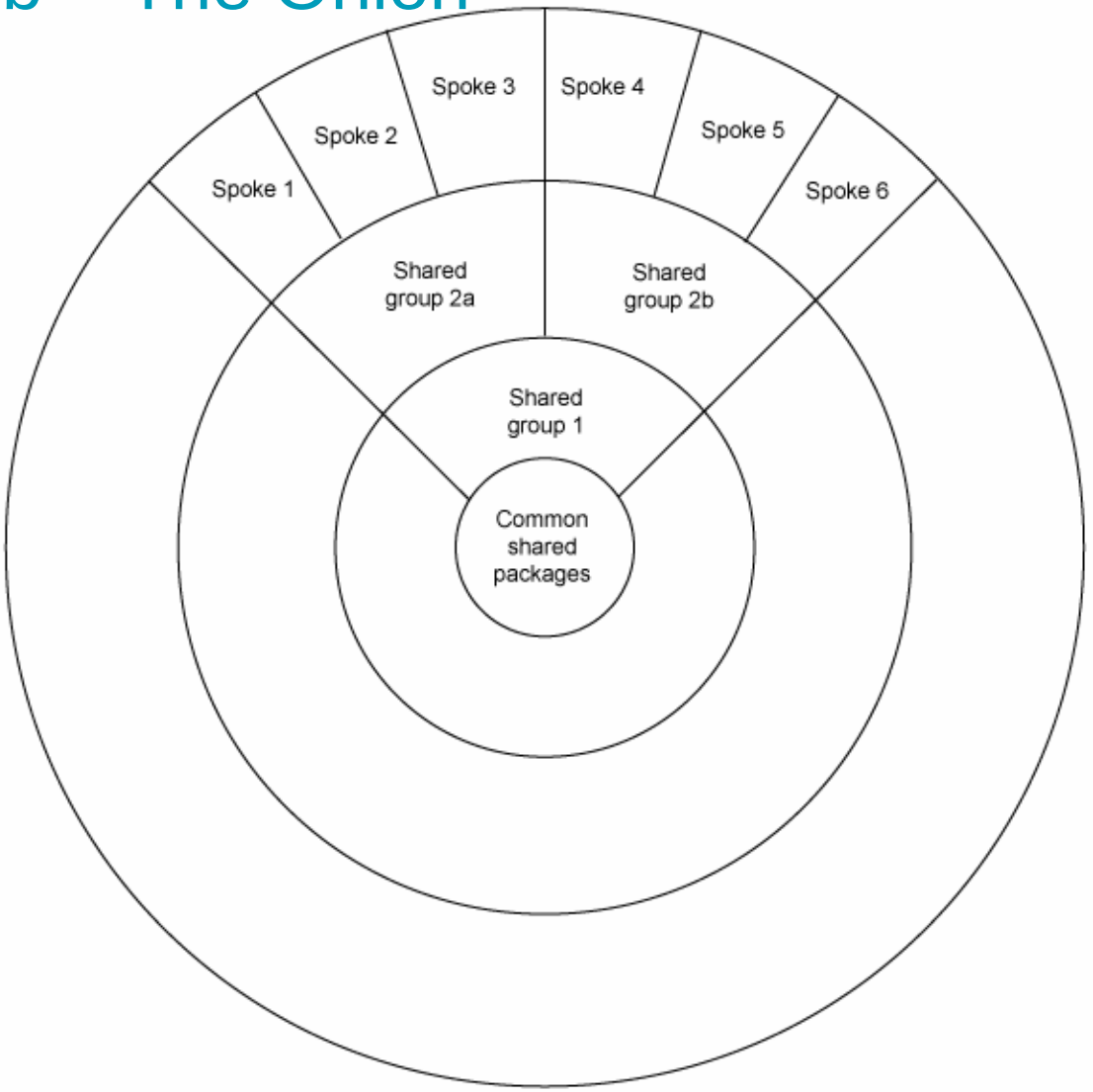
- Shared packages occupy one or more “hub” models at the center of a virtual “wheel”
- Related diagrams occupy individual “spokes” fanning out from the center, with their references to shared elements pointing inwards to the hub models and never sideways to other spokes
- Advantages include:
  - High context for merges, allows full model integrity protection
  - Small memory footprint with one spoke and one or more hub models opened
  - Separation allows for strong ownership, minimizing complex merges
  - Fast repository operations for a small number of large files
- Disadvantages include:
  - Requires discipline to maintain appropriate separation between spokes
  - Complete hierarchy requires a mapping model with shortcuts

Hub and Spoke Partitioning



Partitioning Option 1b – The Onion

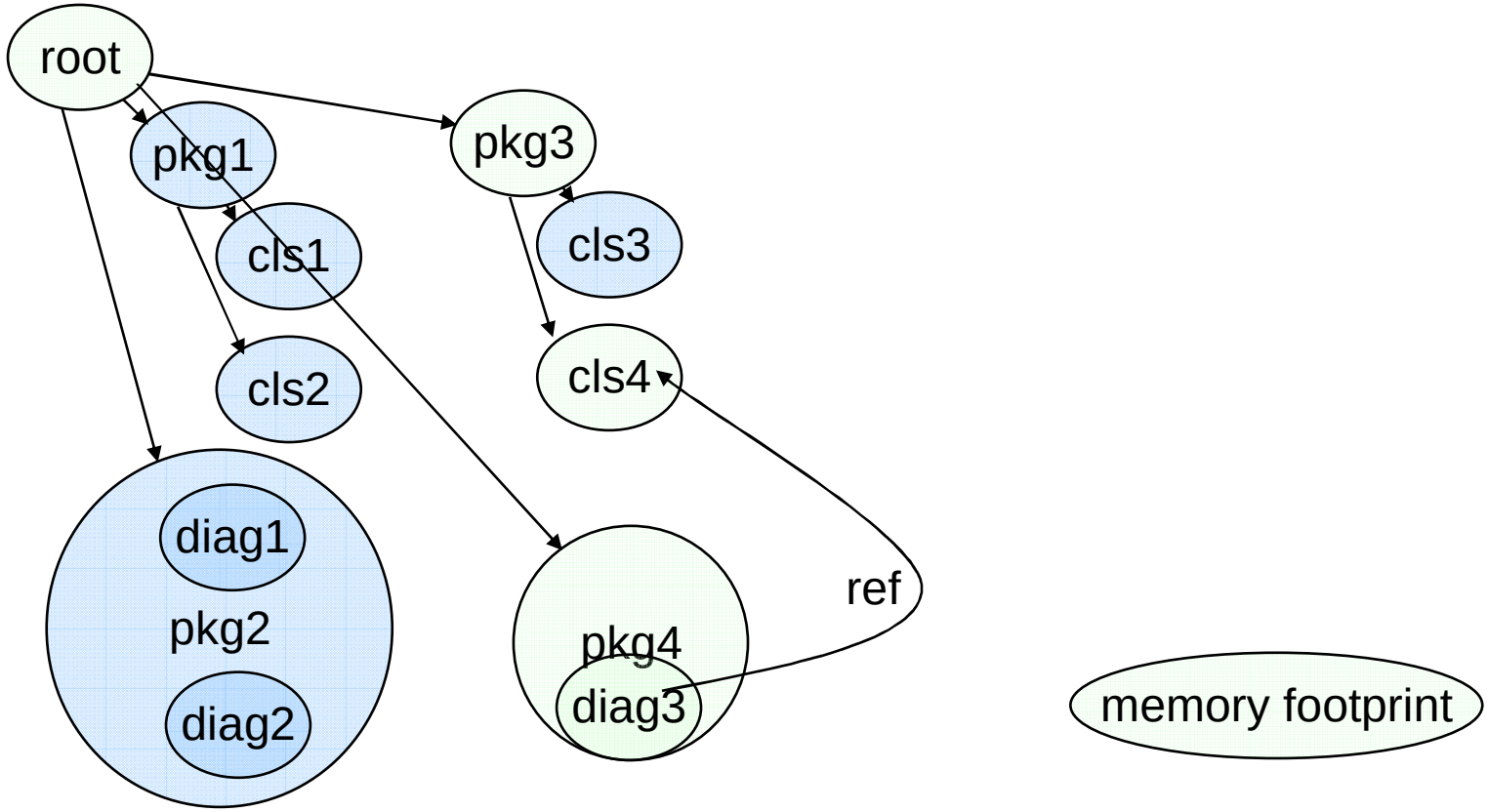
- An advanced form of hub and spoke partitioning entails an *onion* layering where each layer moving outward provides more specialization and less sharing
- Complex, but a powerful form of reuse



Partitioning Option 2 – Fragmentation

- Break the model into tiny fragments down to classifier element
- Advantages include:
  - Very small memory footprint with only a small number of fragments in memory at any one time
  - Fast repository operations when shared dynamic views on a single integration stream are used
  - Complete hierarchy is visible I the model explorer
- Disadvantages include:
  - Low context for merges, model integrity protection is mitigated in UCM and snapshot views
  - No obvious separation into files could lead to spaghetti fragments with many interconnections around the model
  - Slow repository operations for logical model merging in CVS (complete model is built thrice)

Fragmentation Memory Footprint



Enterprise Best Practices for Merging

- ClearCase and ClearQuest with UCM
  - Integrated solution handles multiple components and streams
  - Models reside in a VOB (versioned object base)
  - Models are propagated up and down a hierarchy of streams using rebase and deliver commands
  - Individual modelers can create development streams on any specific integration stream, allowing them to pick up baselines and deliver new work
  - Rebase and deliver are atomic operations with rollback, providing a very safe environment for major operations
  - Development streams are private, so either snapshot or dynamic views can be used depending on local disk and backup policies
  - Global development is supported with the multi-site version of ClearCase

Small Team Best Practices for Merging

- ClearCase LT works as previous slide (but has limitations)
- CVS can work well and allows fragmentation to be used with merging because of strong logical model merge support
  - Can be slow with fragmentation because many artifacts must be loaded up to 4 times to perform a full merge
  - CVS can branch and thus support labeled releases
  - Works very well on the WAN for home workers
  - Uses optimistic locking so performs well right up until the moment of synchronization with the repository

Enterprise Best Practices for Merge Avoidance

- ClearCase with a single integration stream and dynamic views
- All modelers work in repository mode, simply changing the model when they want
- Frequent check-ins keep things moving
- With enforced reserved check-outs, ClearCase provides access control to fragments so that merges are completely avoided
- Even long operations like refactorings are protected by roll-backs in the event of an access failure

Reference Materials

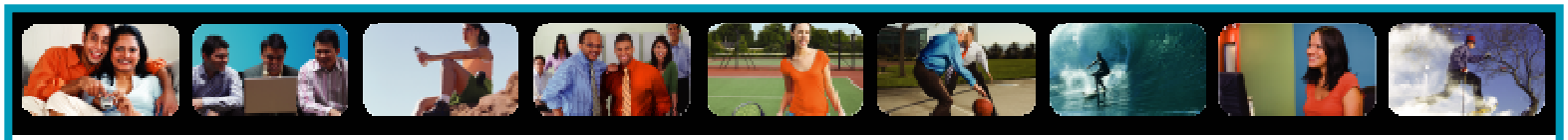
- Comparing and merging UML models in IBM Rational Software Architect – Kim Letkeman
  - Part 1 – Comparing models with local history
    - http://www-128.ibm.com/developerworks/rational/library/05/712_comp/
  - Part 2 – Merging models using "compare with each other"
    - http://www-128.ibm.com/developerworks/rational/library/05/712_comp2/
  - Part 3 – A deeper understanding of model merging
    - http://www-128.ibm.com/developerworks/rational/library/05/802_comp3/
  - Part 4 – Parallel model development with CVS
    - http://www-128.ibm.com/developerworks/rational/library/05/0809_CVS4/
  - Part 5 – Model management with IBM Rational ClearCase and IBM Rational Software Architect Version 7 and later
    - http://www.ibm.com/developerworks/rational/library/07/0703_letkeman/

Reference Materials

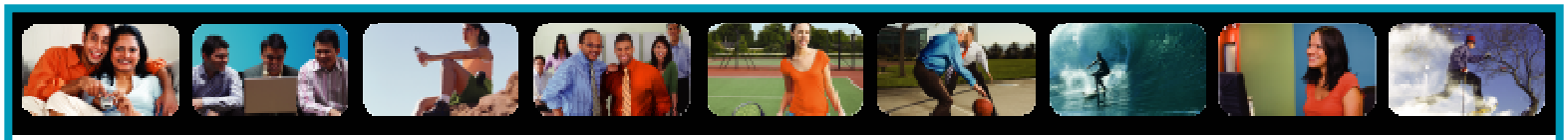
- Comparing and merging UML models in IBM Rational Software Architect – Kim Letkeman
  - Part 6 – Parallel model development with custom profiles
    - http://www-128.ibm.com/developerworks//rational/library/05/0823_Letkeman/
  - Part 7 – Ad-hoc modeling – Fusing two models with diagrams
    - http://www-128.ibm.com/developerworks/rational/library/07/0410_letkeman/
- Related Articles
  - Authoring UML profiles using Rational Software Architect and Rational Software Modeler – *Dusko Misic*
    - http://www.ibm.com/developerworks/rational/library/05/0906_dusko/
  - Model Structure Guidelines for Rational Software Modeler and Rational Software Architect – *Bill Smith*
    - http://www.ibm.com/developerworks/rational/library/04/r-3155/

Agenda

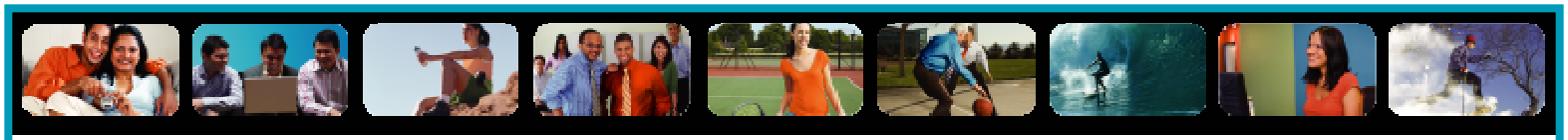
- Introduction
- Team Development
- Model Driven Development
- Enterprise Model Management and Partitioning
- Demo



Demo



Questions



Thank You

Kim Letkeman

kletkema@ca.ibm.com