

**Eike Stepper**

[stepper@esc-net.de](mailto:stepper@esc-net.de)  
<http://www.esc-net.de>  
<http://thegordian.blogspot.com>

**Berlin, Germany**



# Scale, Share and Store your Models with CDO

---

**Eclipse Live Webinar, April 29, 2010**

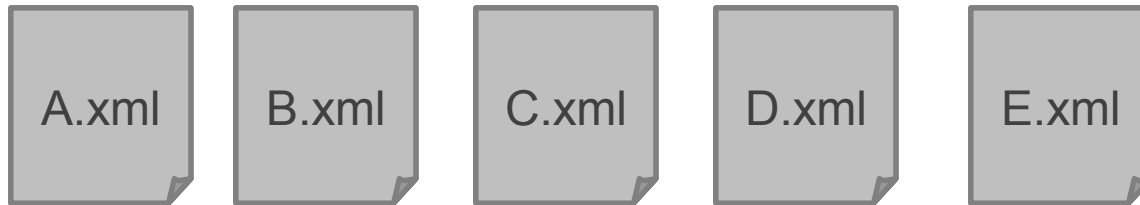
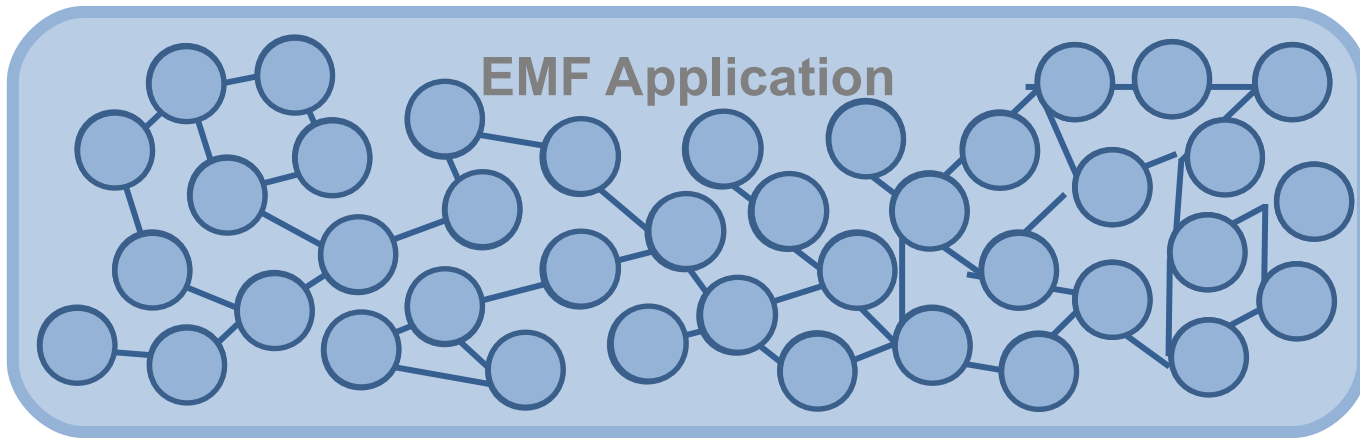
---

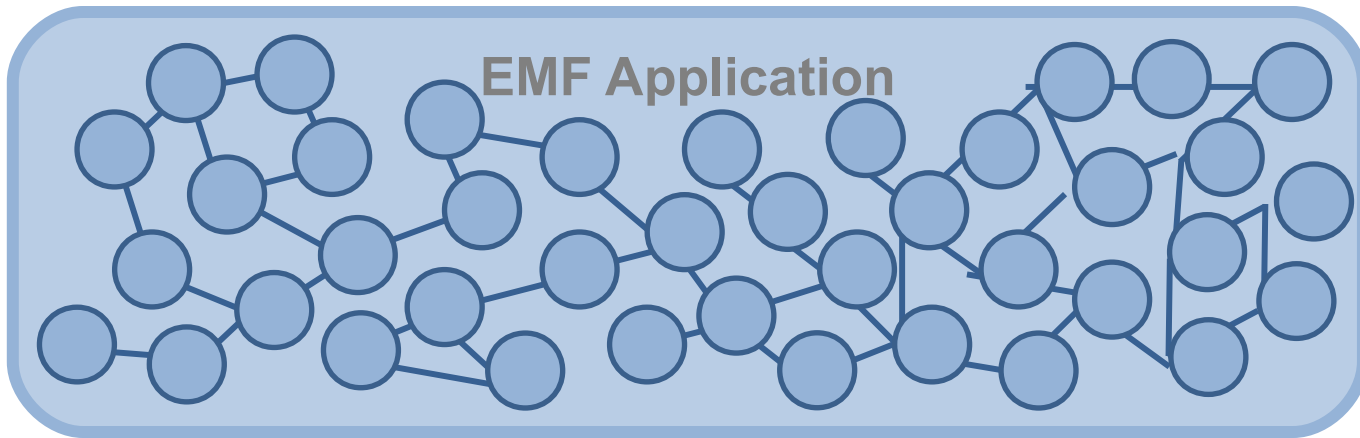


**Ed Merks**

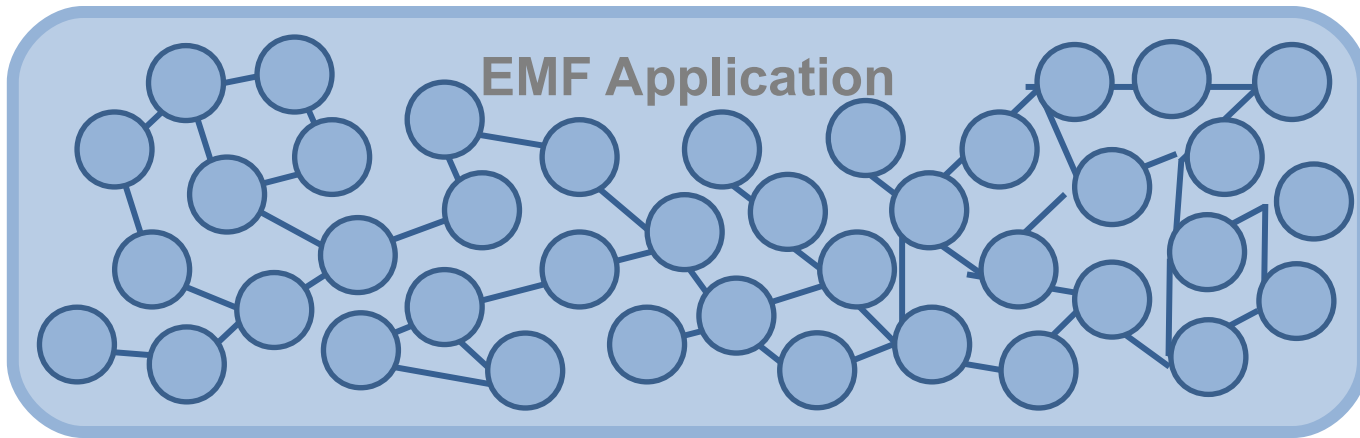
[Ed.Merks@gmail.com](mailto:Ed.Merks@gmail.com)  
<http://www.macromodeling.com>  
<http://ed-merks.blogspot.com>

**Ballantrae, Ontario, Canada**



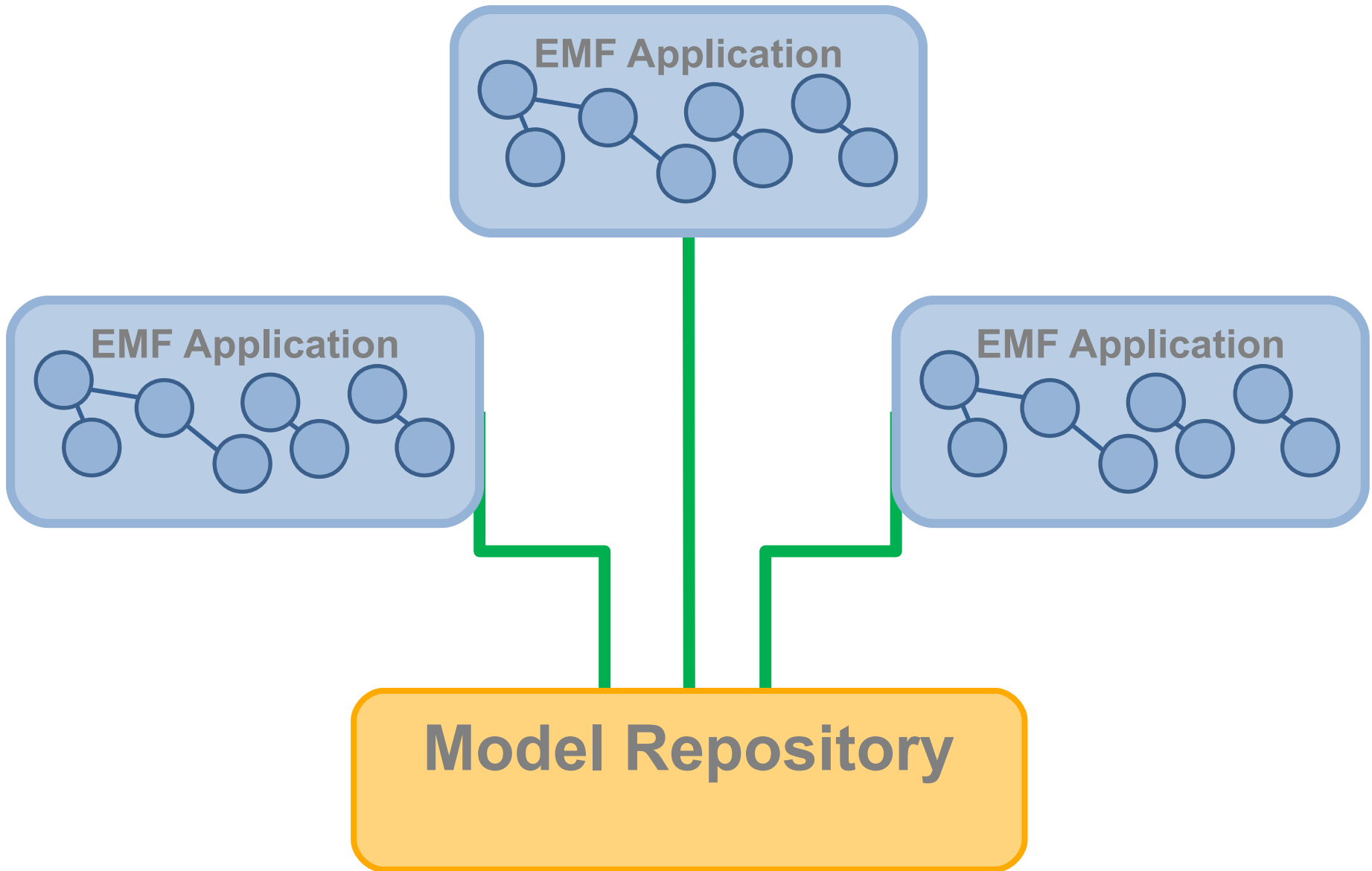


- **Huge models require lots of smaller files**
- **Partitioning must be done at design time**
- **Saving changes is not transactional safe**
- **Loading single objects is still impossible**
- **Garbage collection of objects is impossible**
- **Conflicts must be resolved in text form**

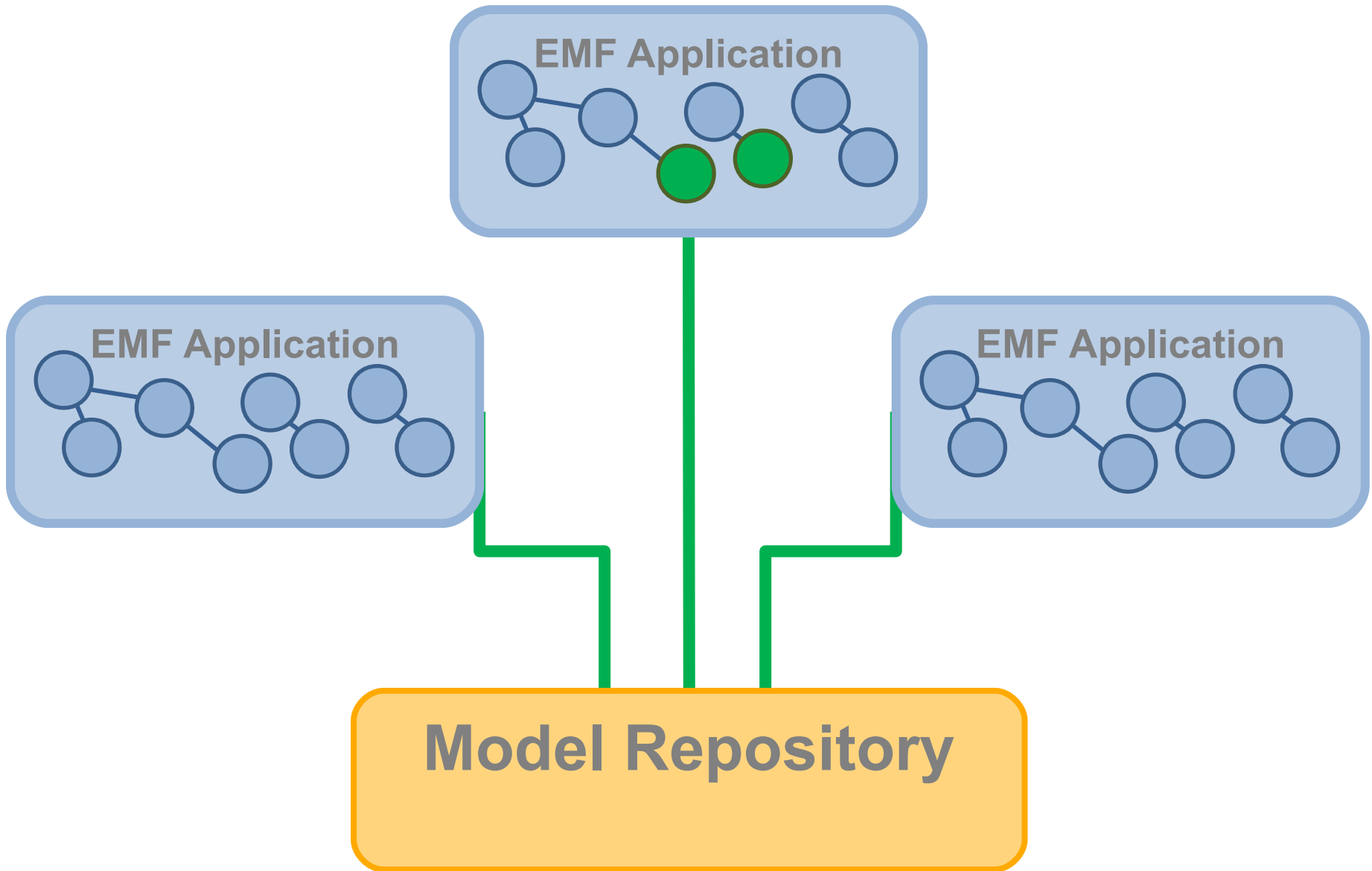


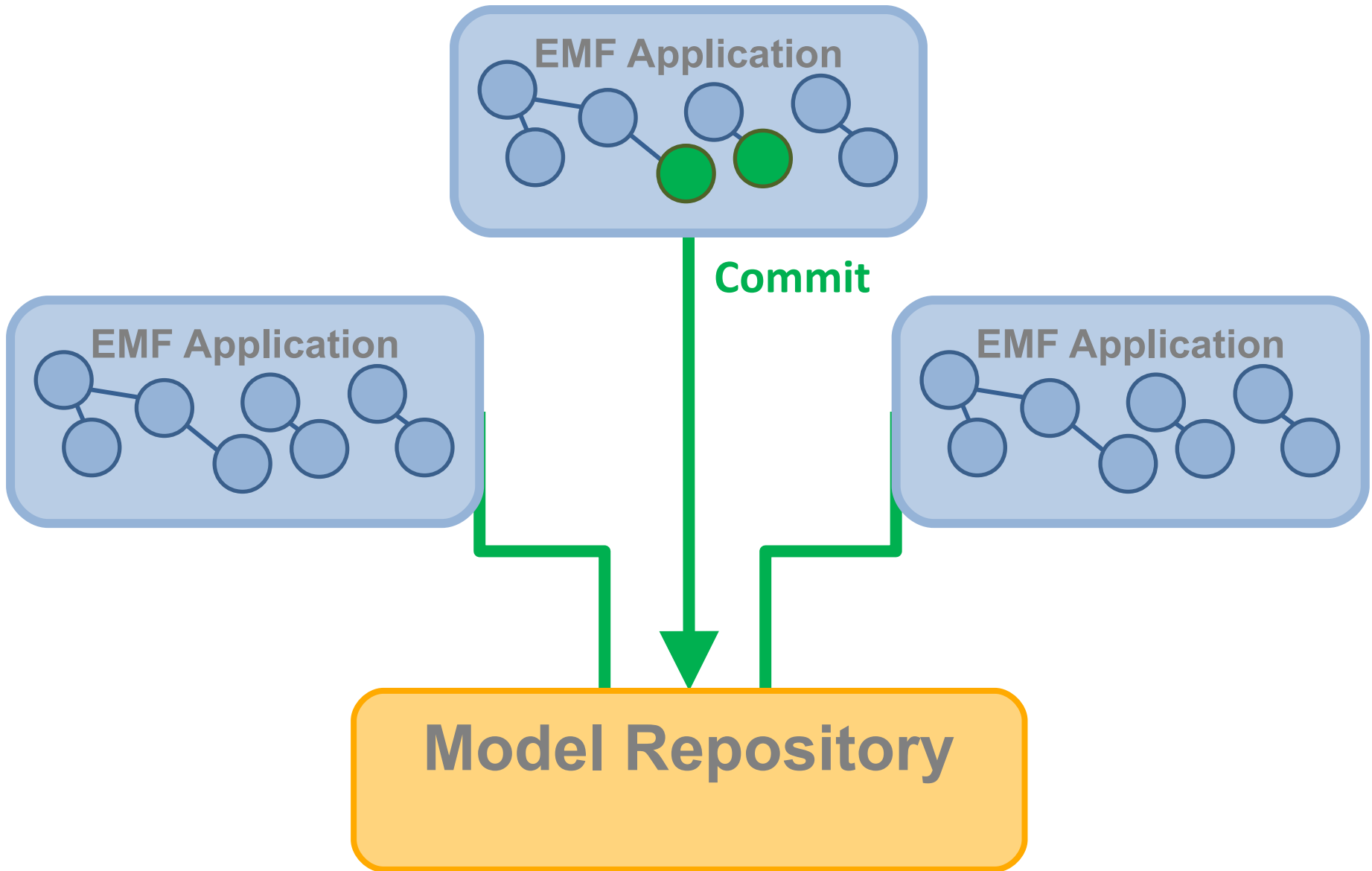
**Does not scale well**

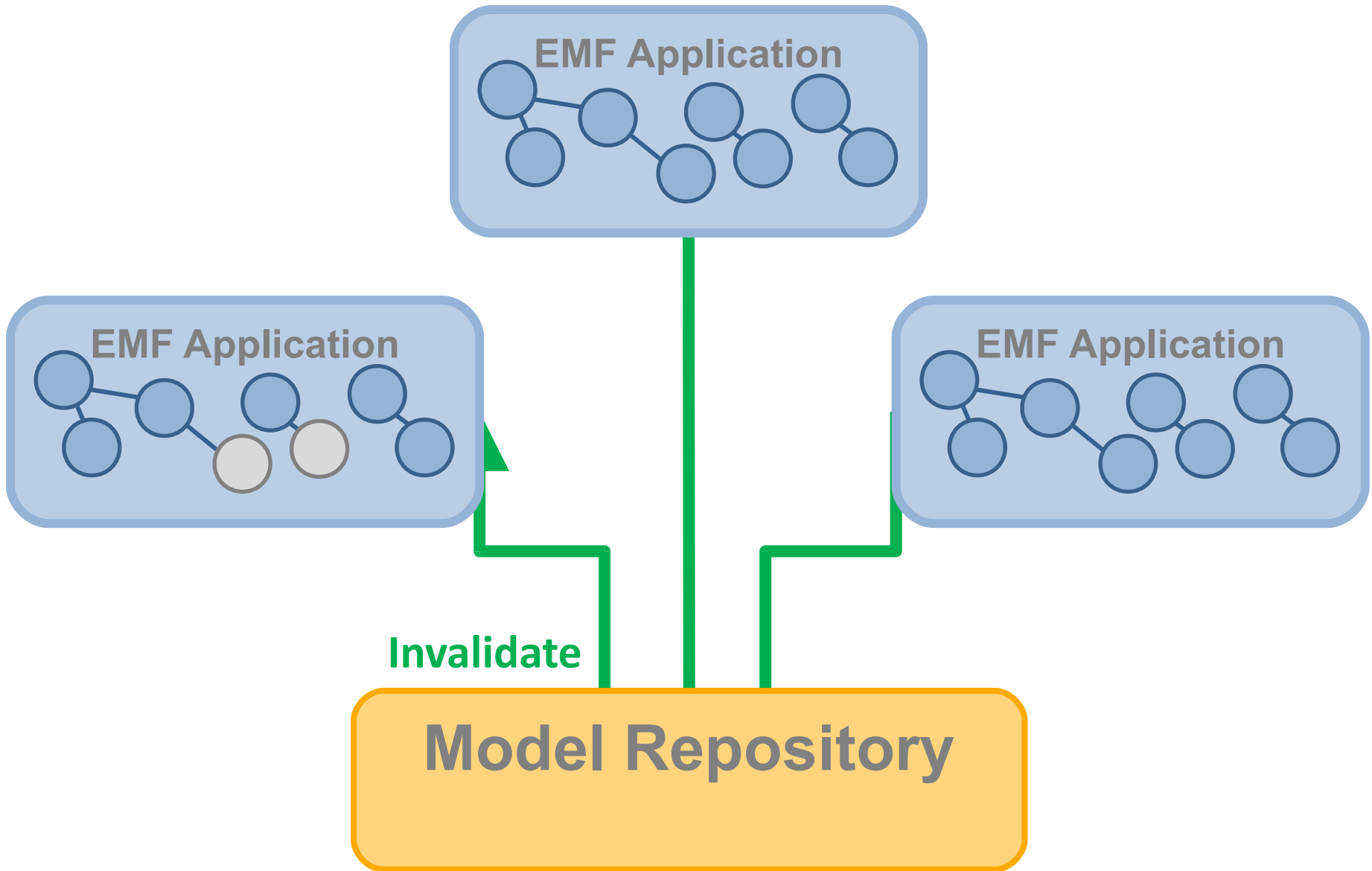
**Not suitable for multi-user**

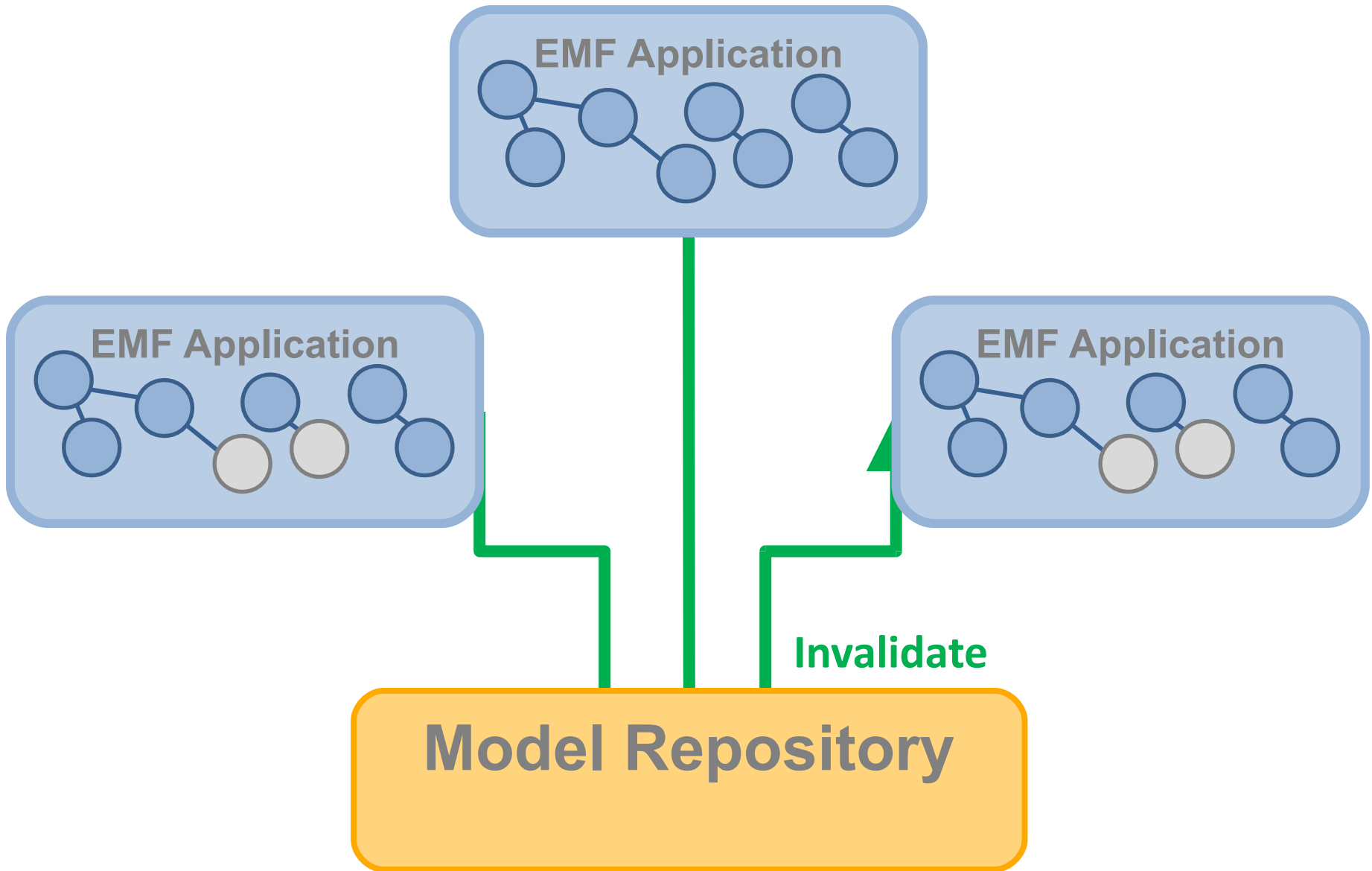


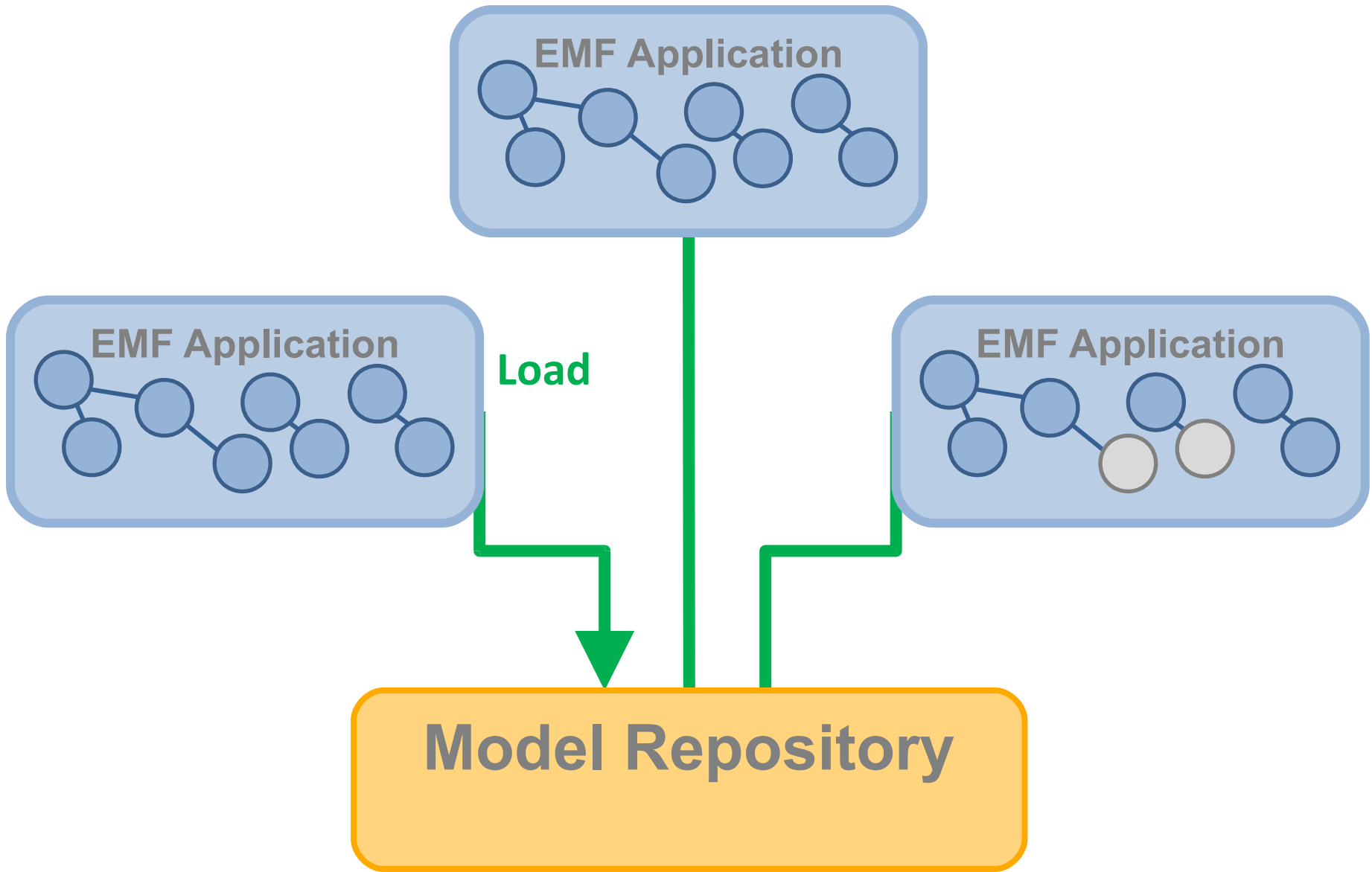
# Modify

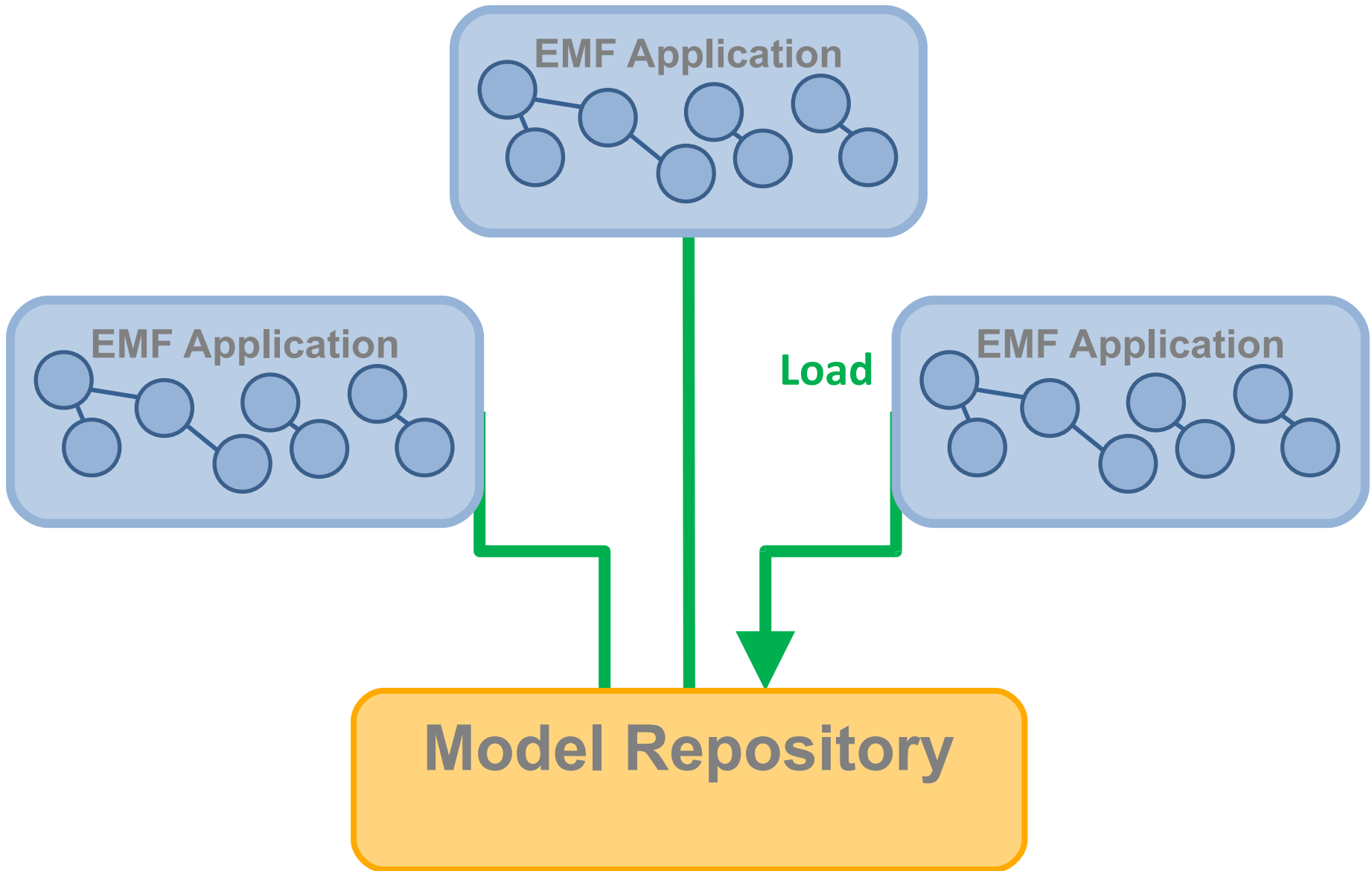


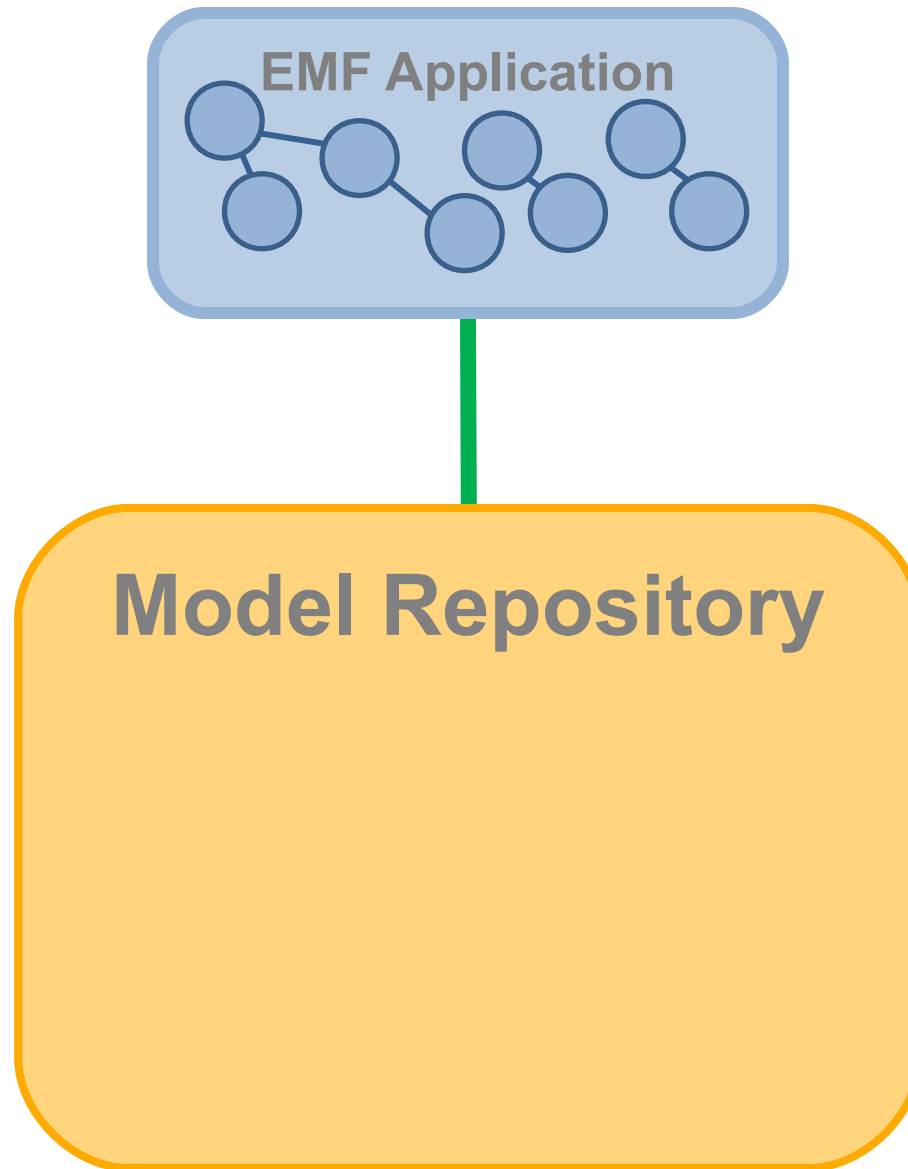


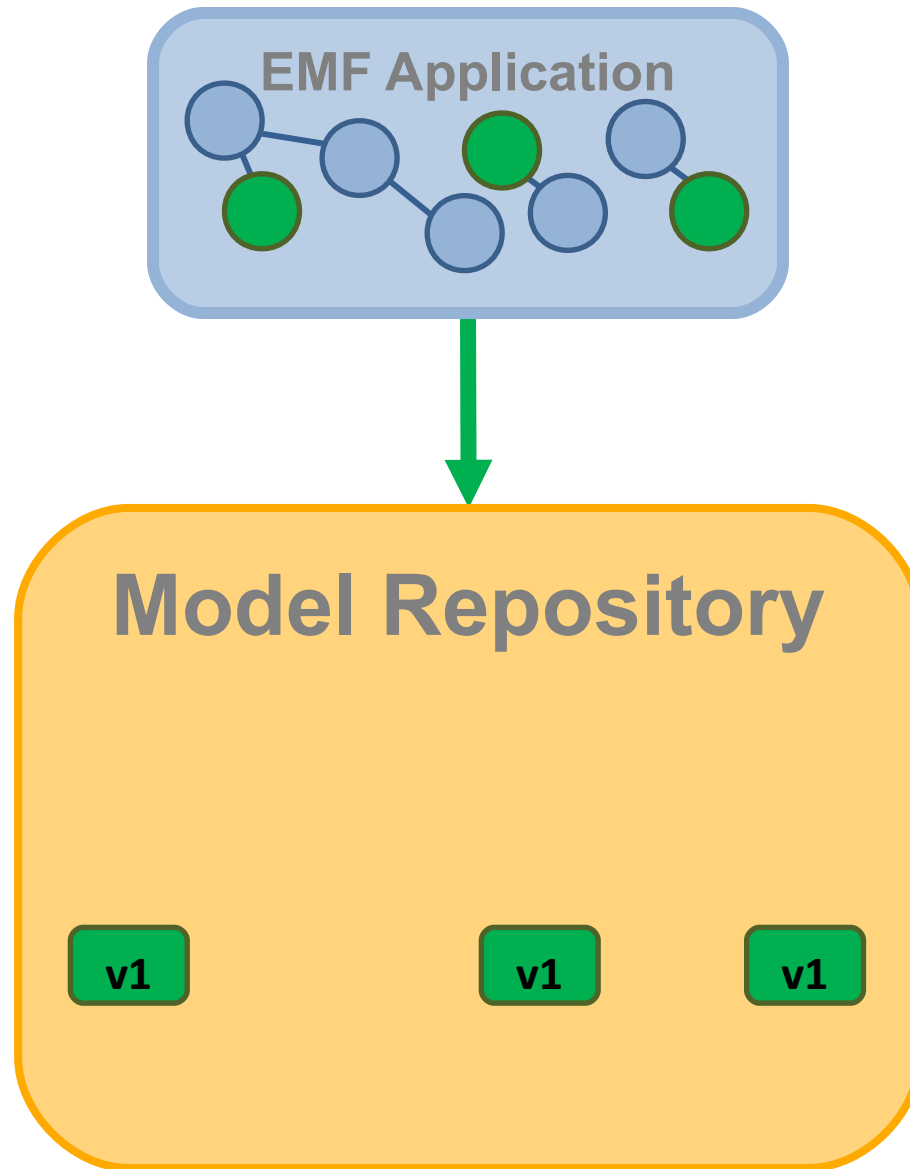


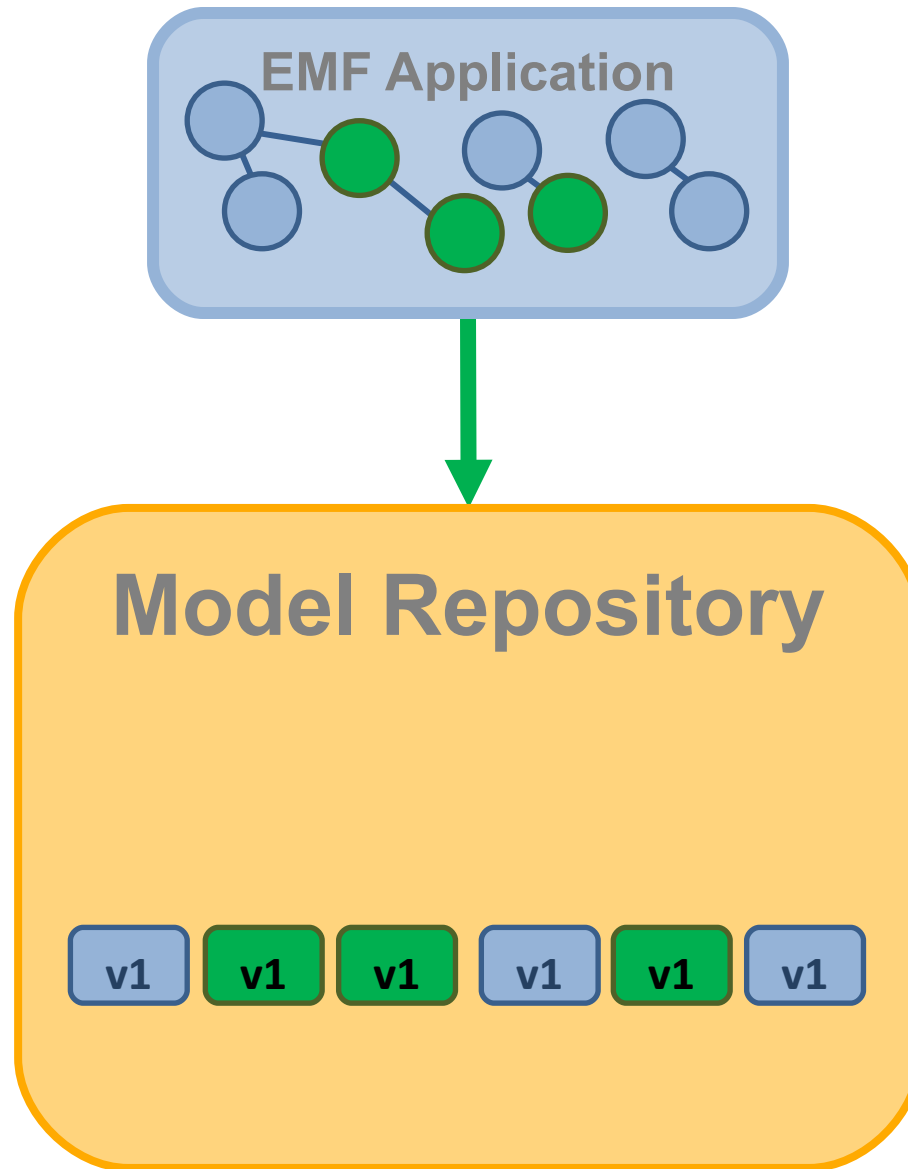


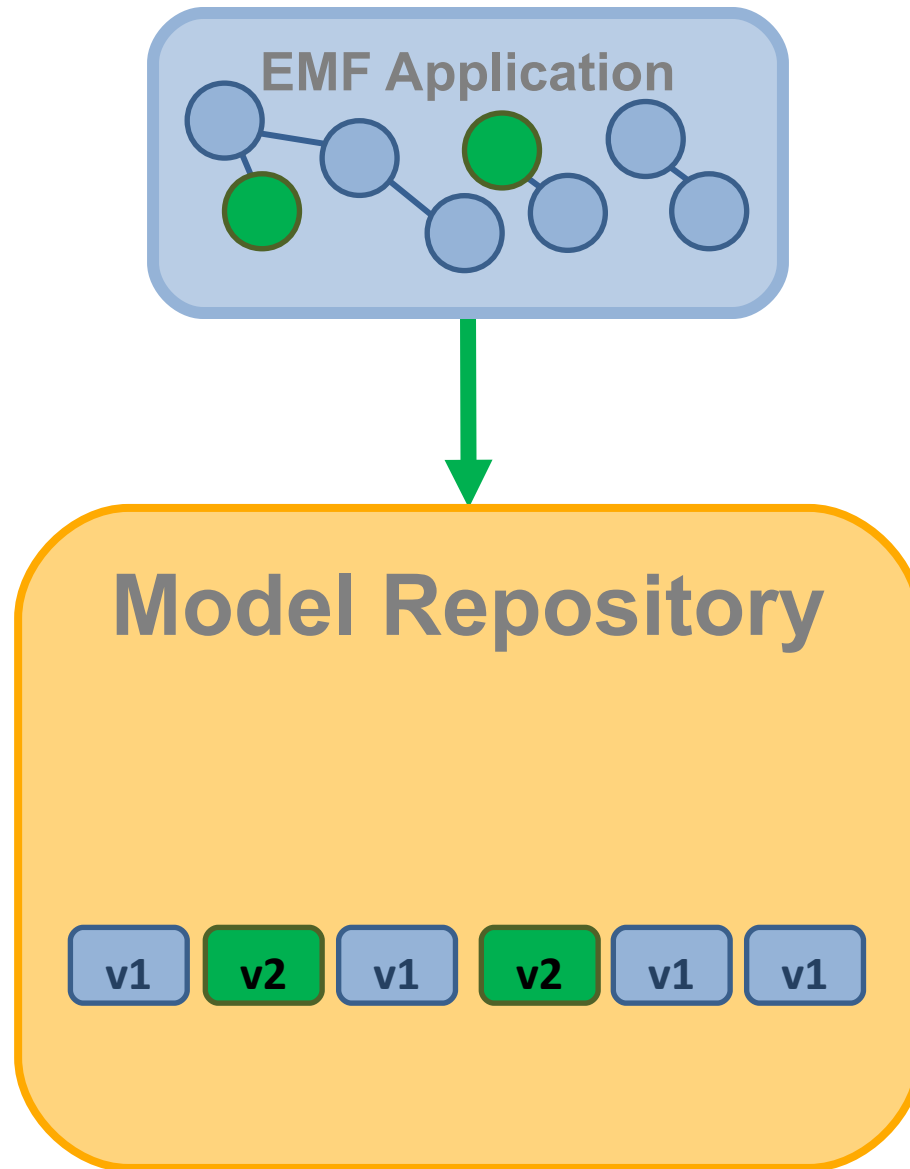


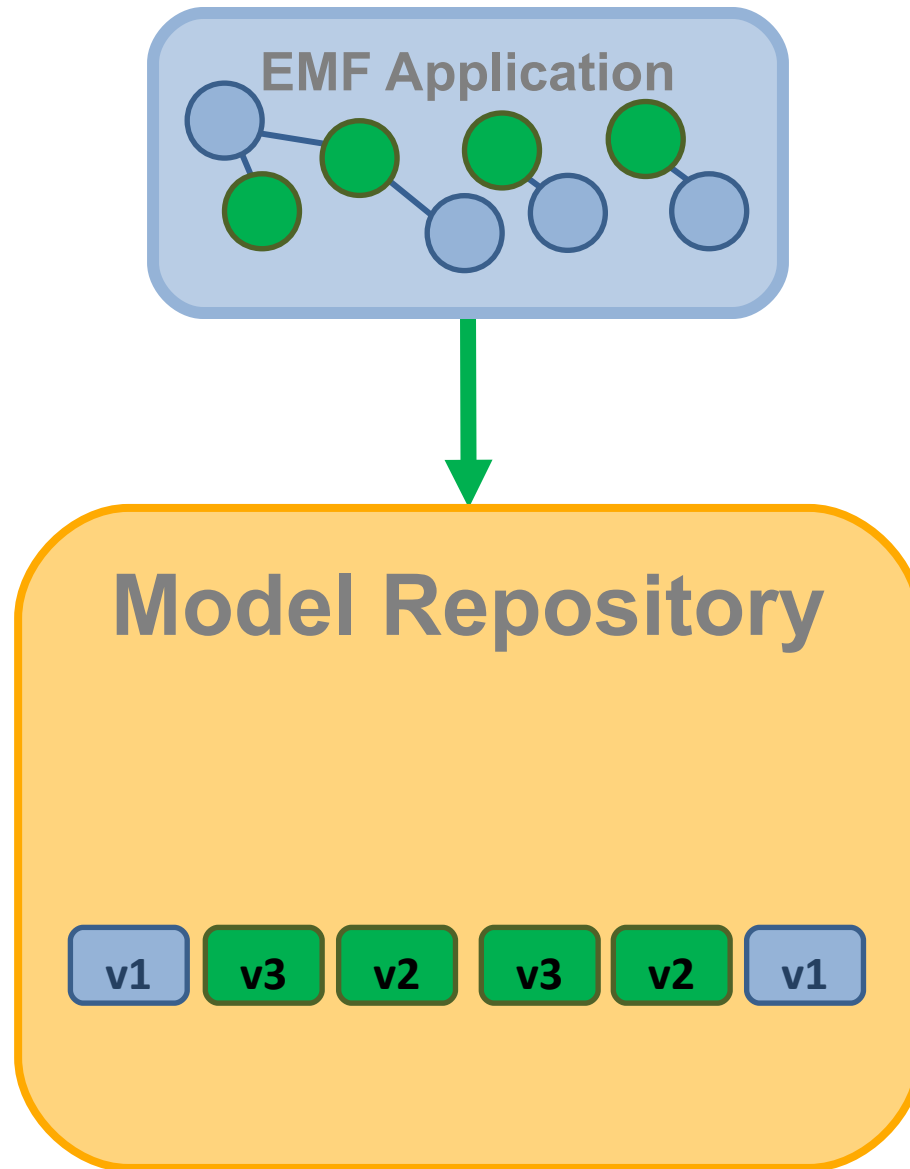


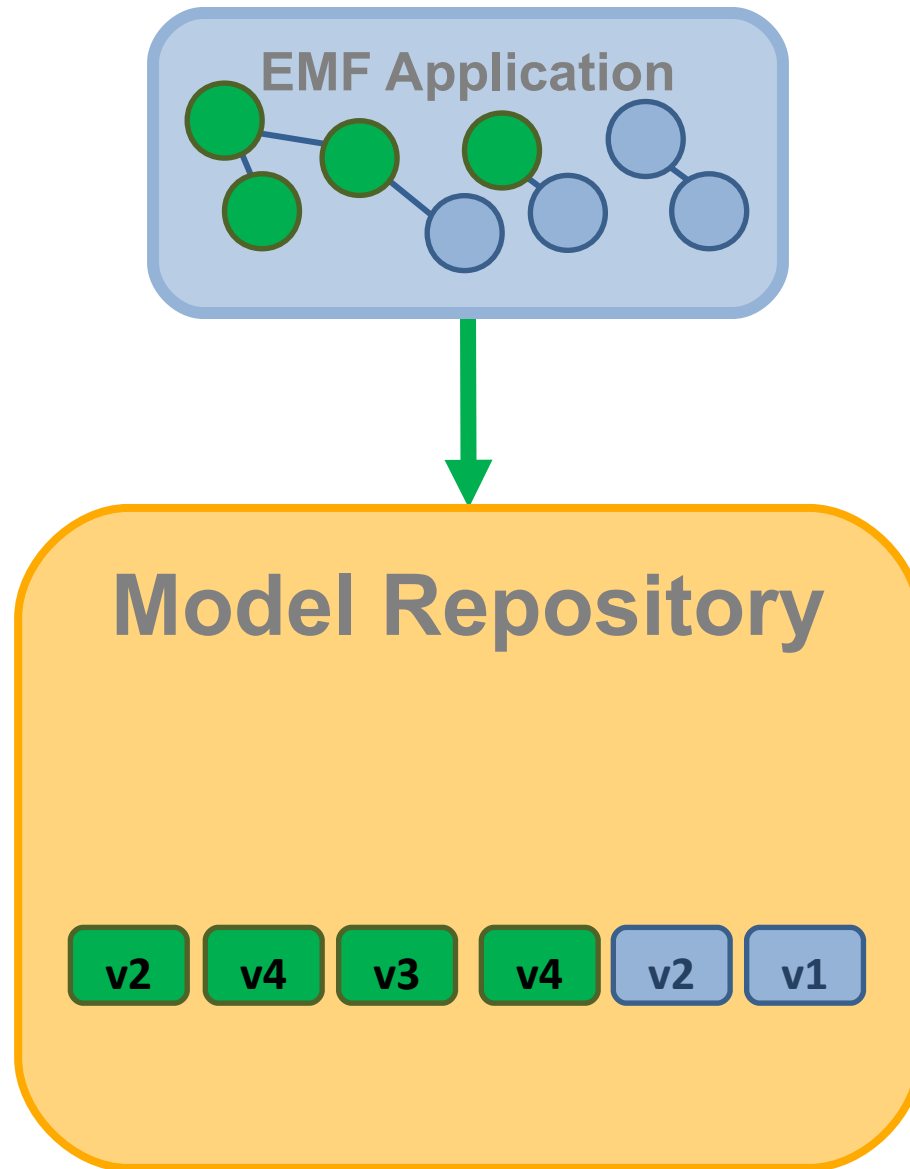


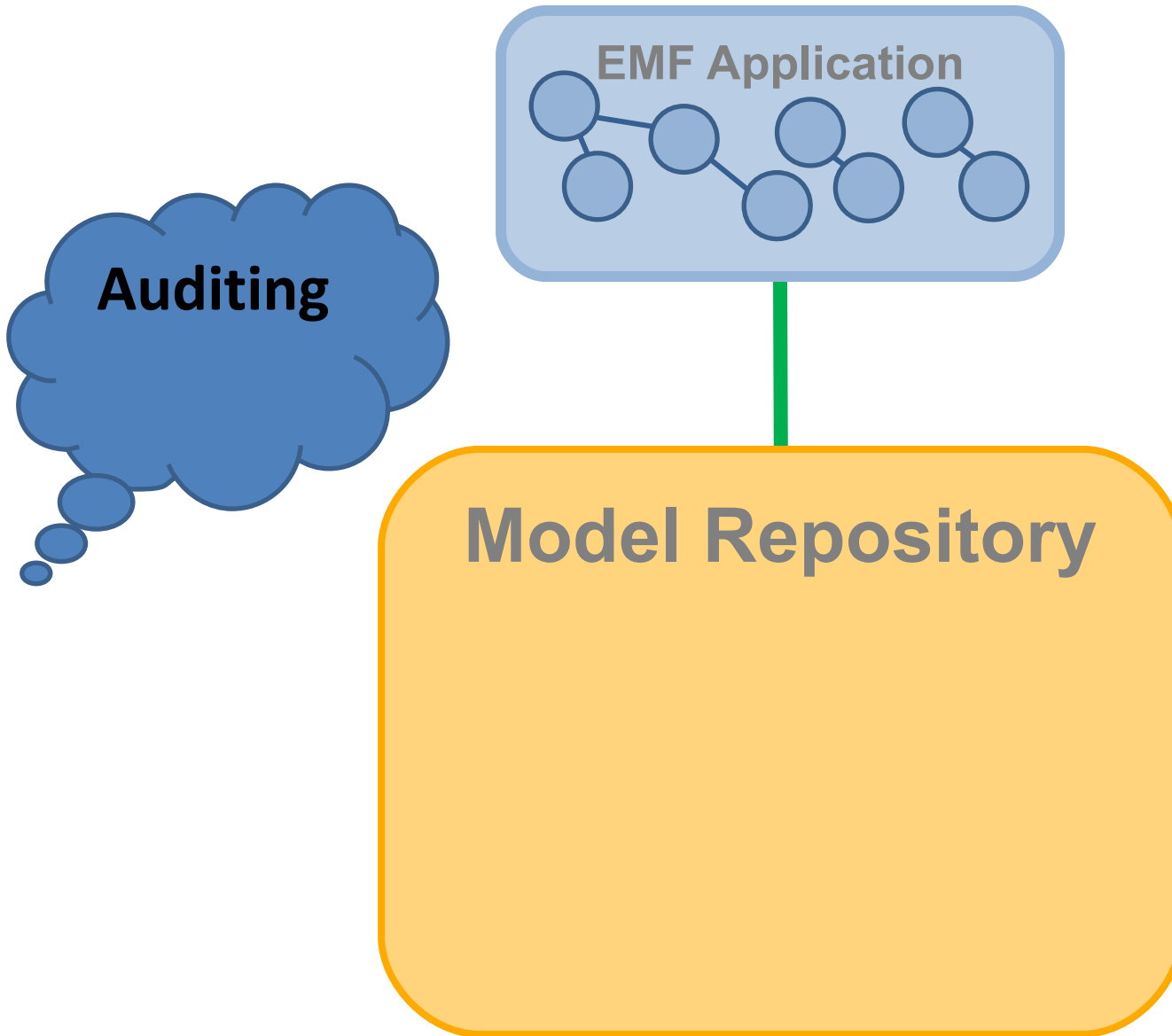


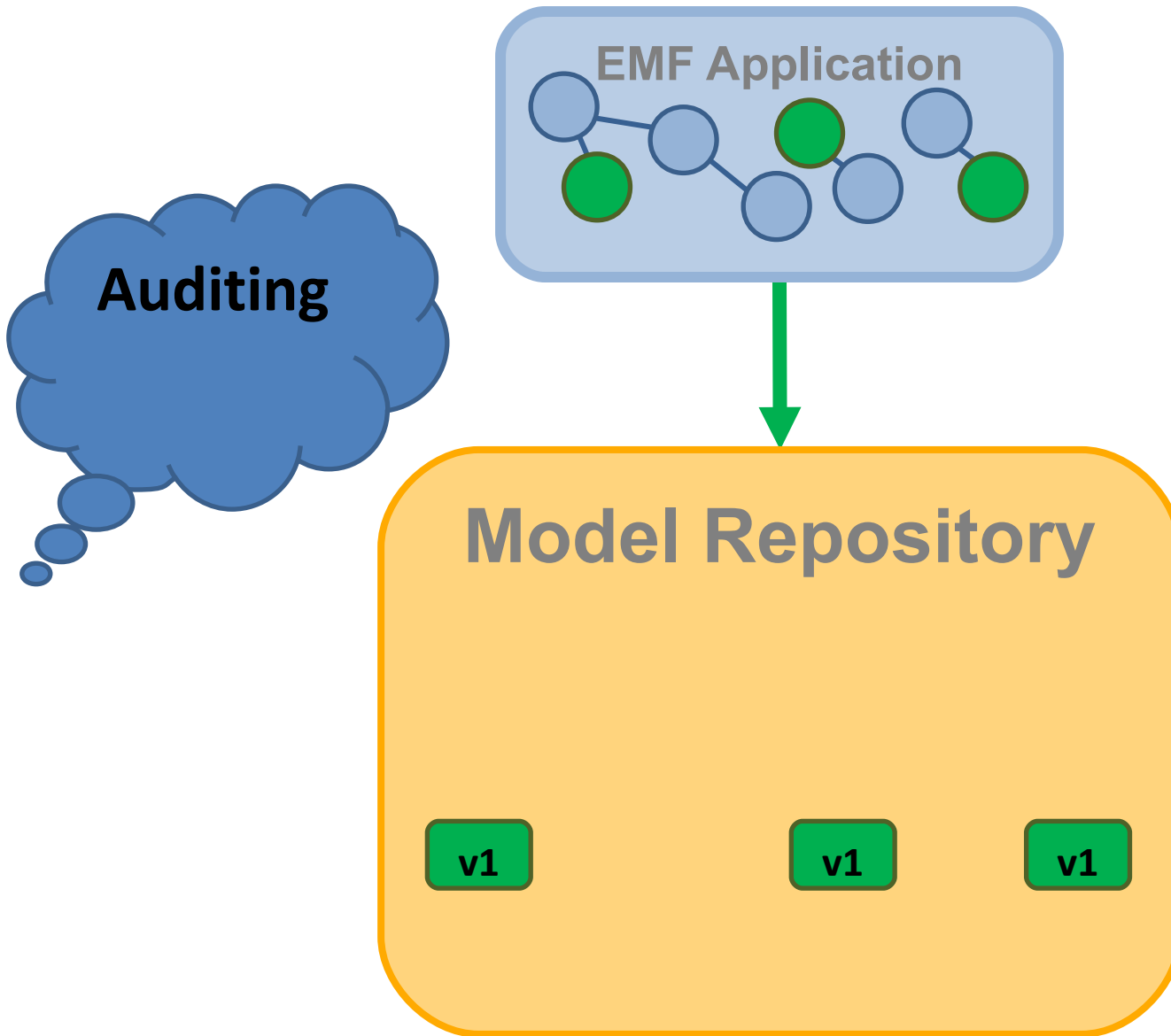


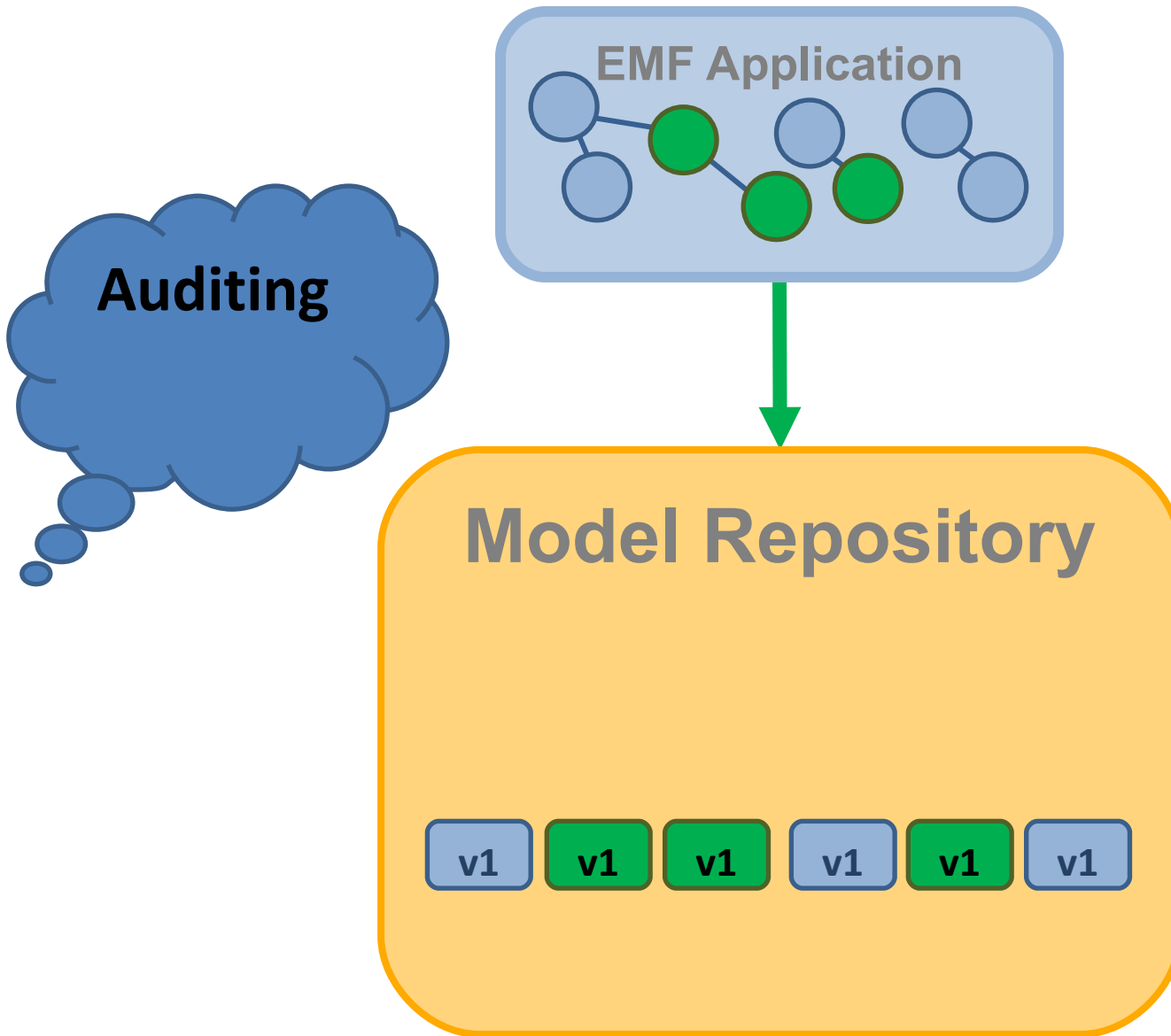


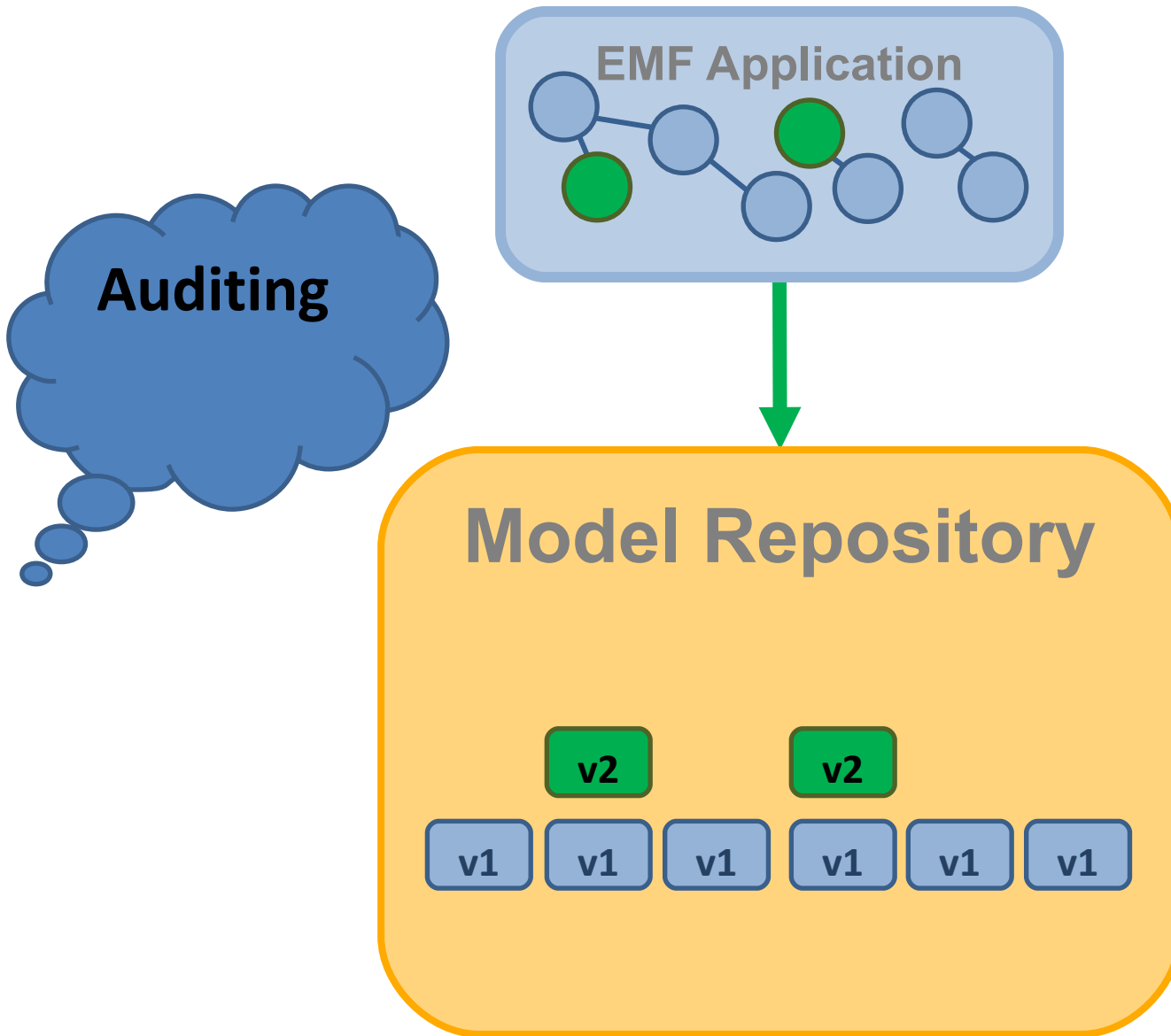


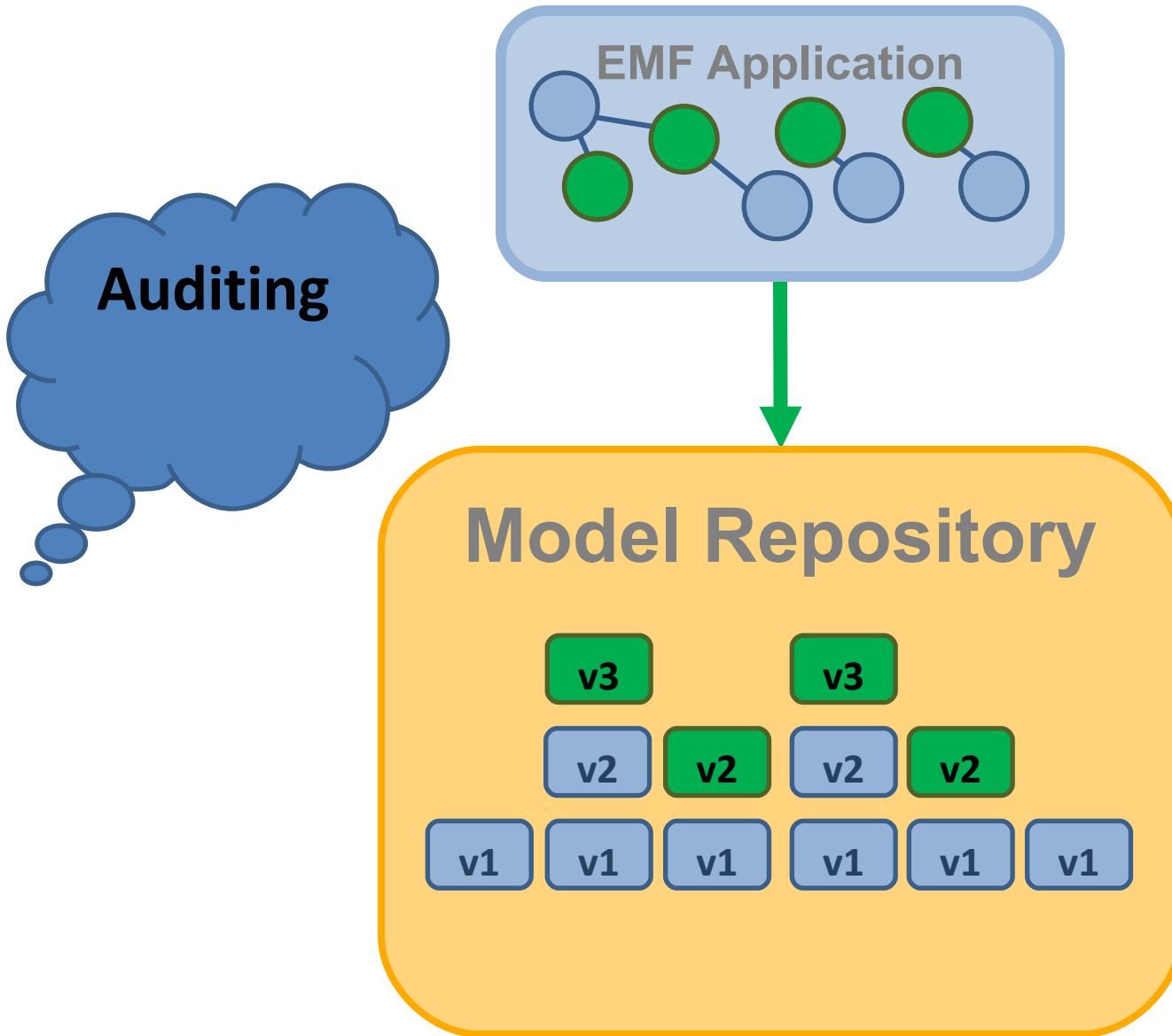


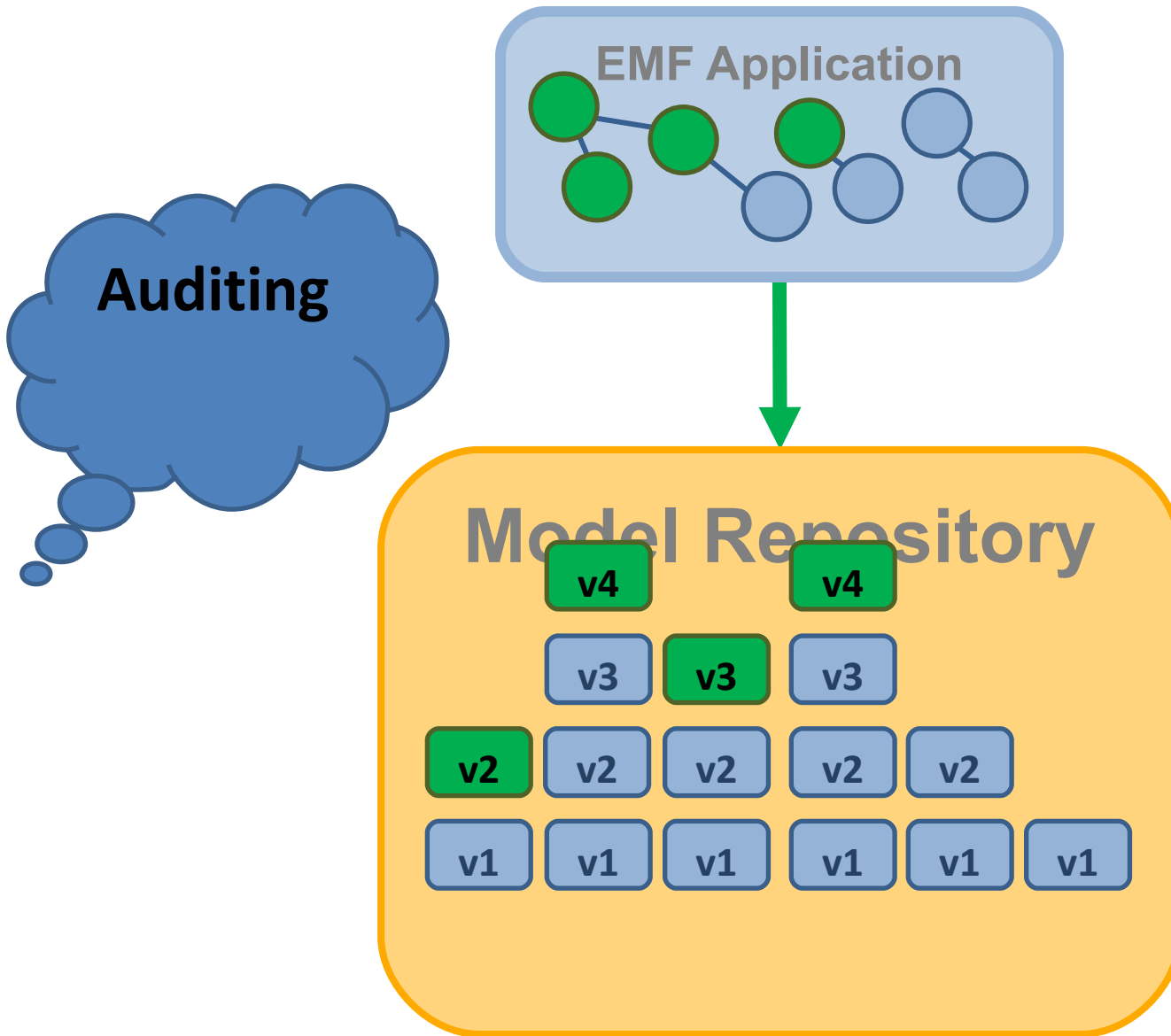


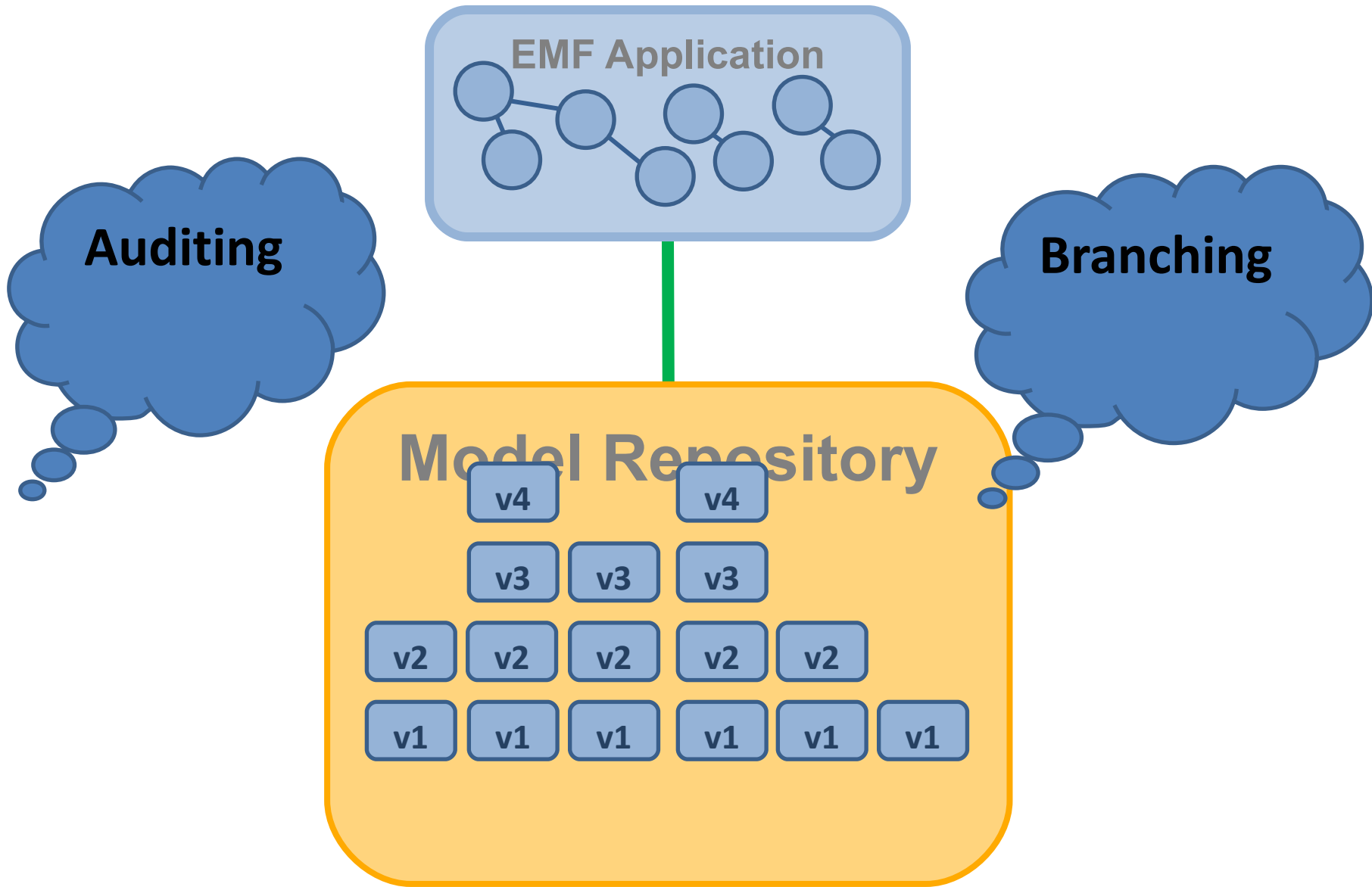


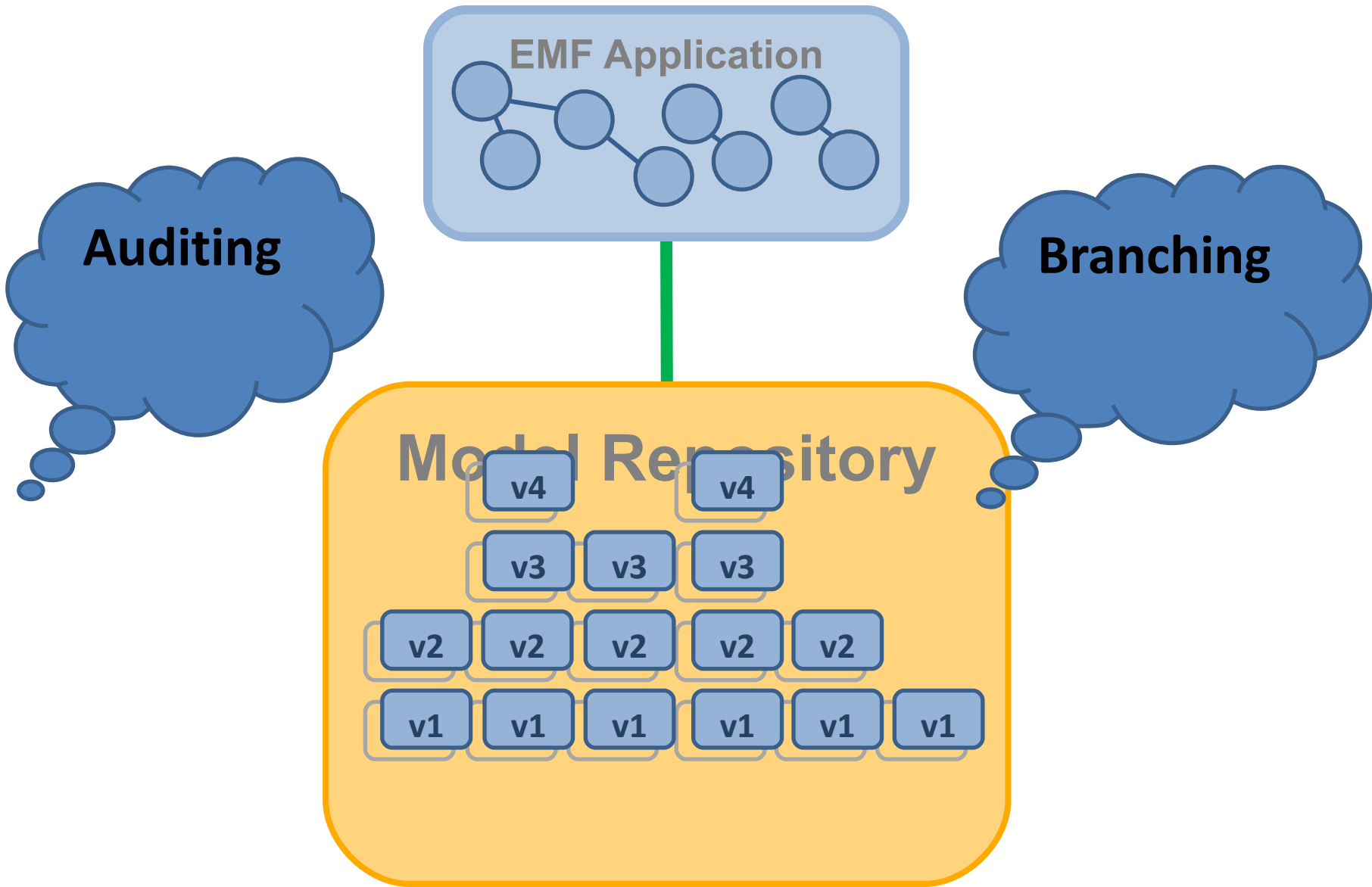


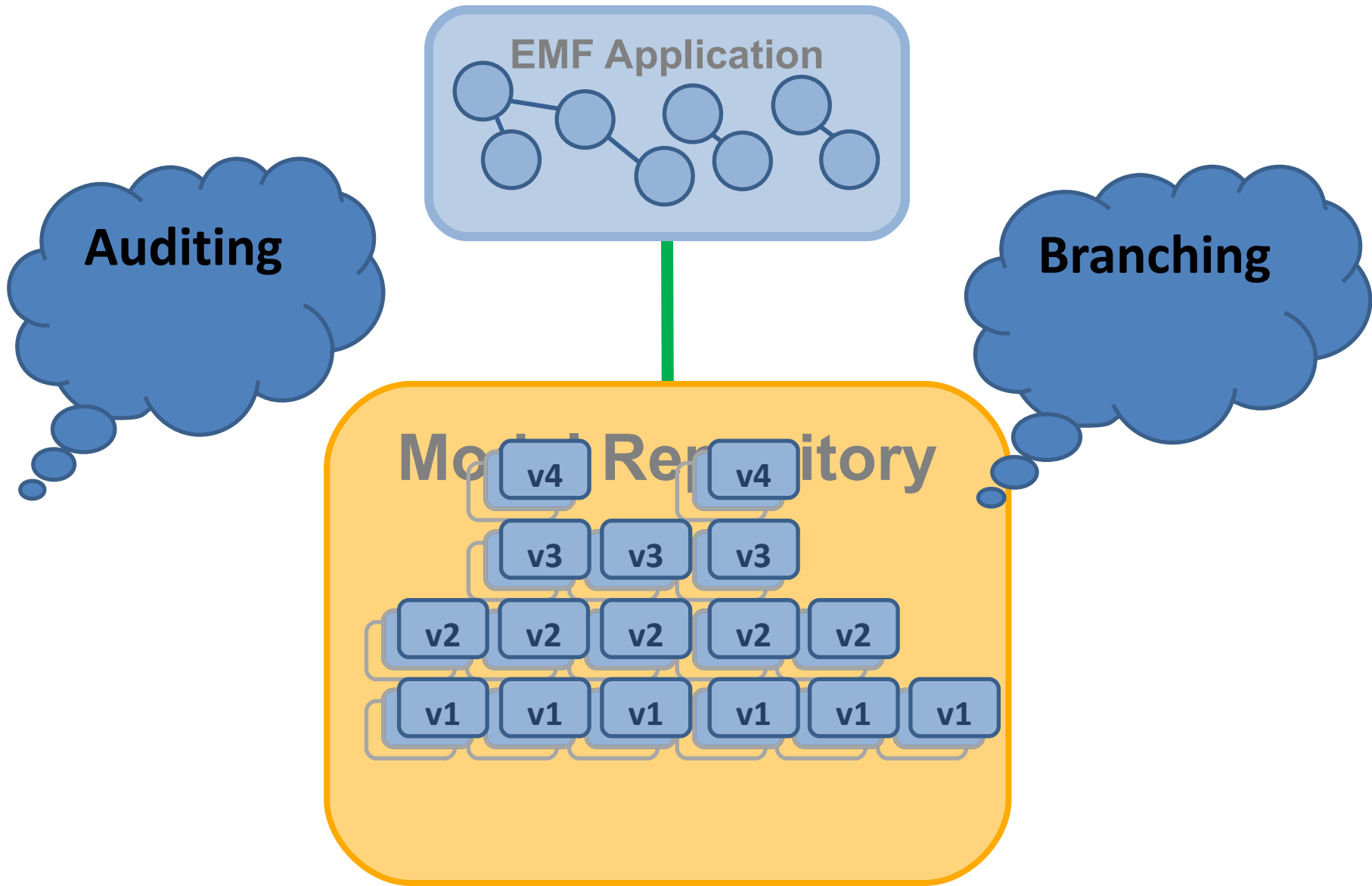


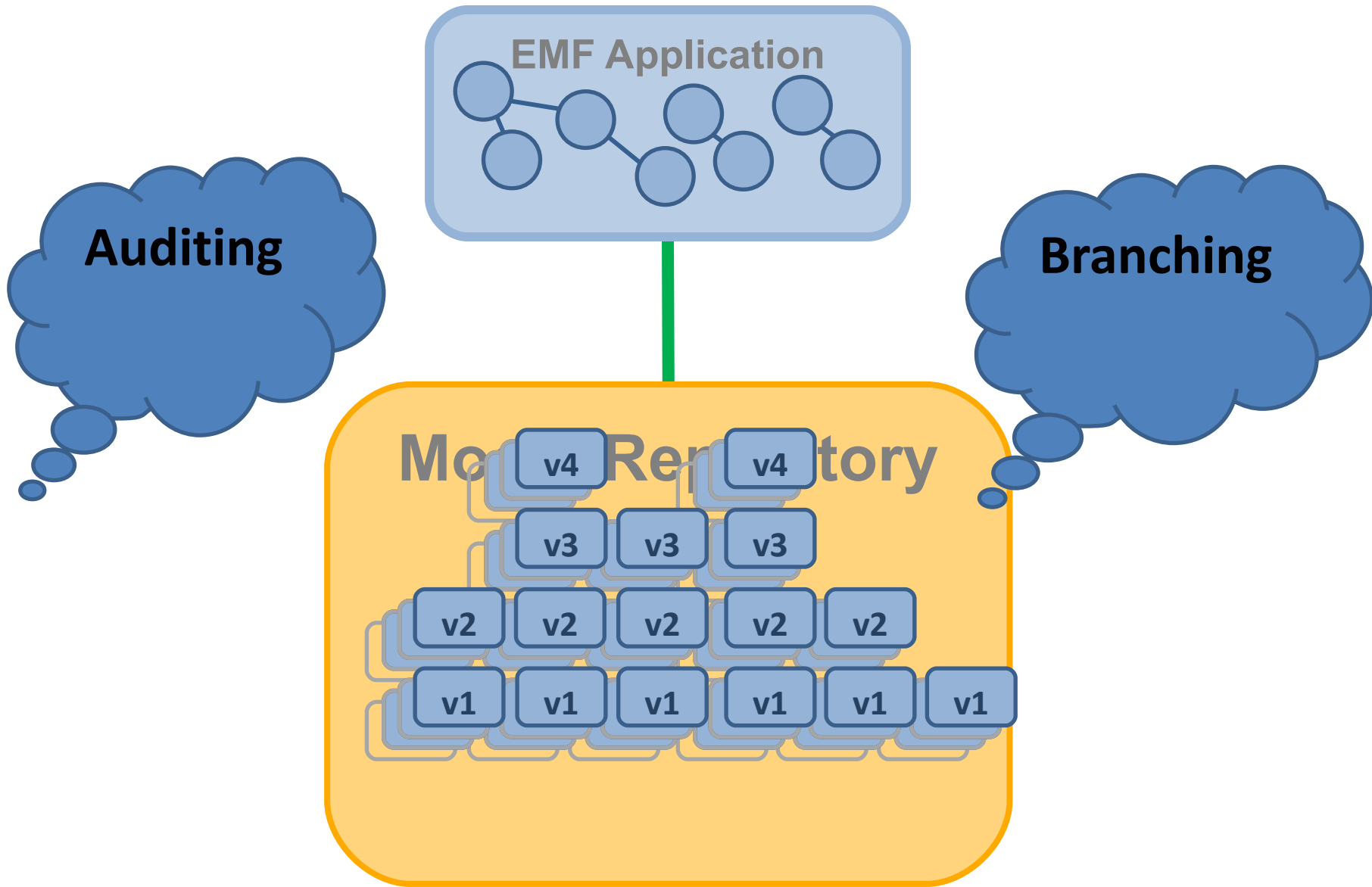












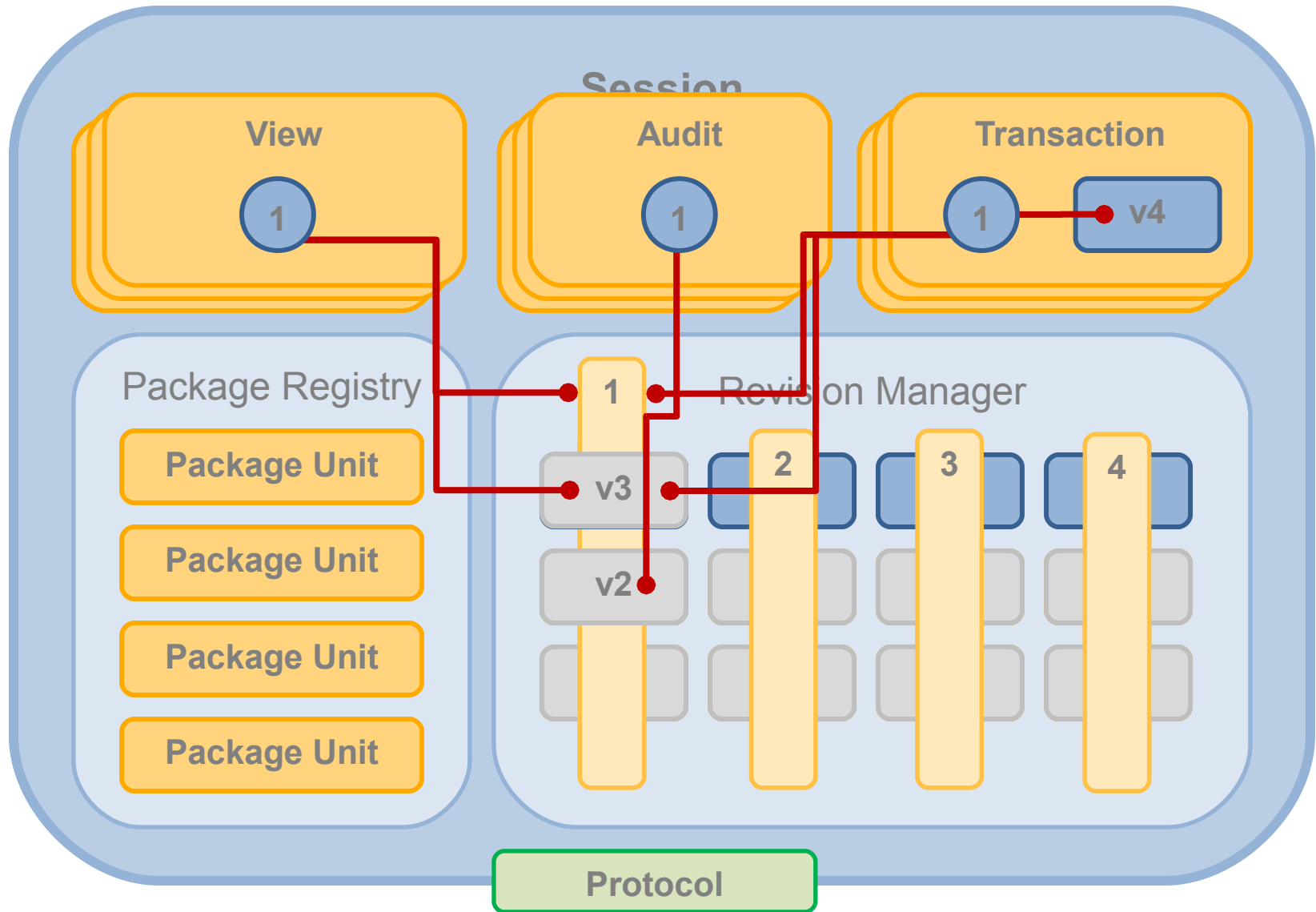
## CDORevision

<b>EClass</b>	<b>eClass</b>
<b>CDOID</b>	<b>id</b>
<b>CDOBranch</b>	<b>branch</b>
<b>h</b>	<b>version</b>
<b>int</b>	<b>created</b>
<b>long</b>	<b>revised</b>
<b>long</b>	

### Revision Data

<b>CDOID</b>	<b>resourceID</b>
<b>CDOID</b>	<b>containerID</b>
<b>int</b>	
<b>containerFeature</b>	
<b>Object[]</b>	<b>values</b>

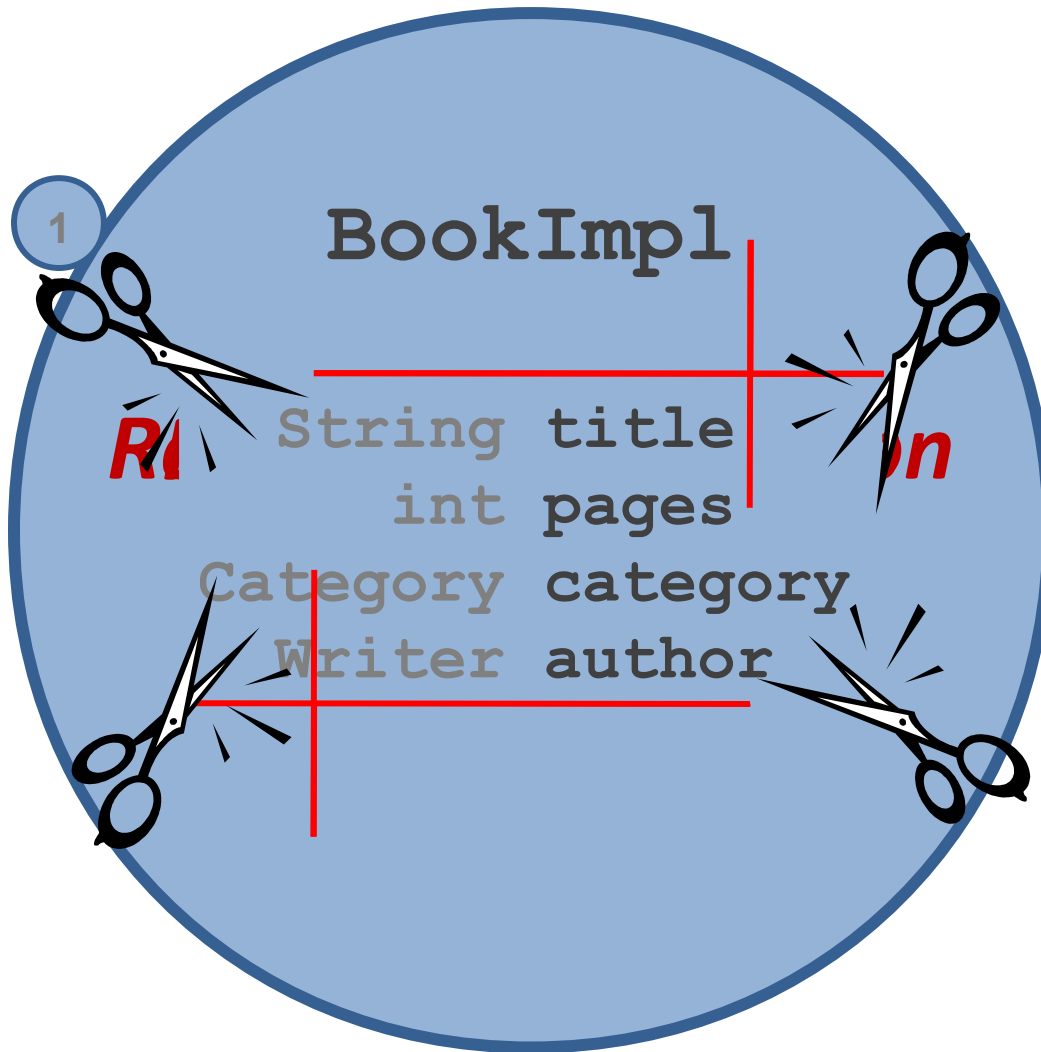


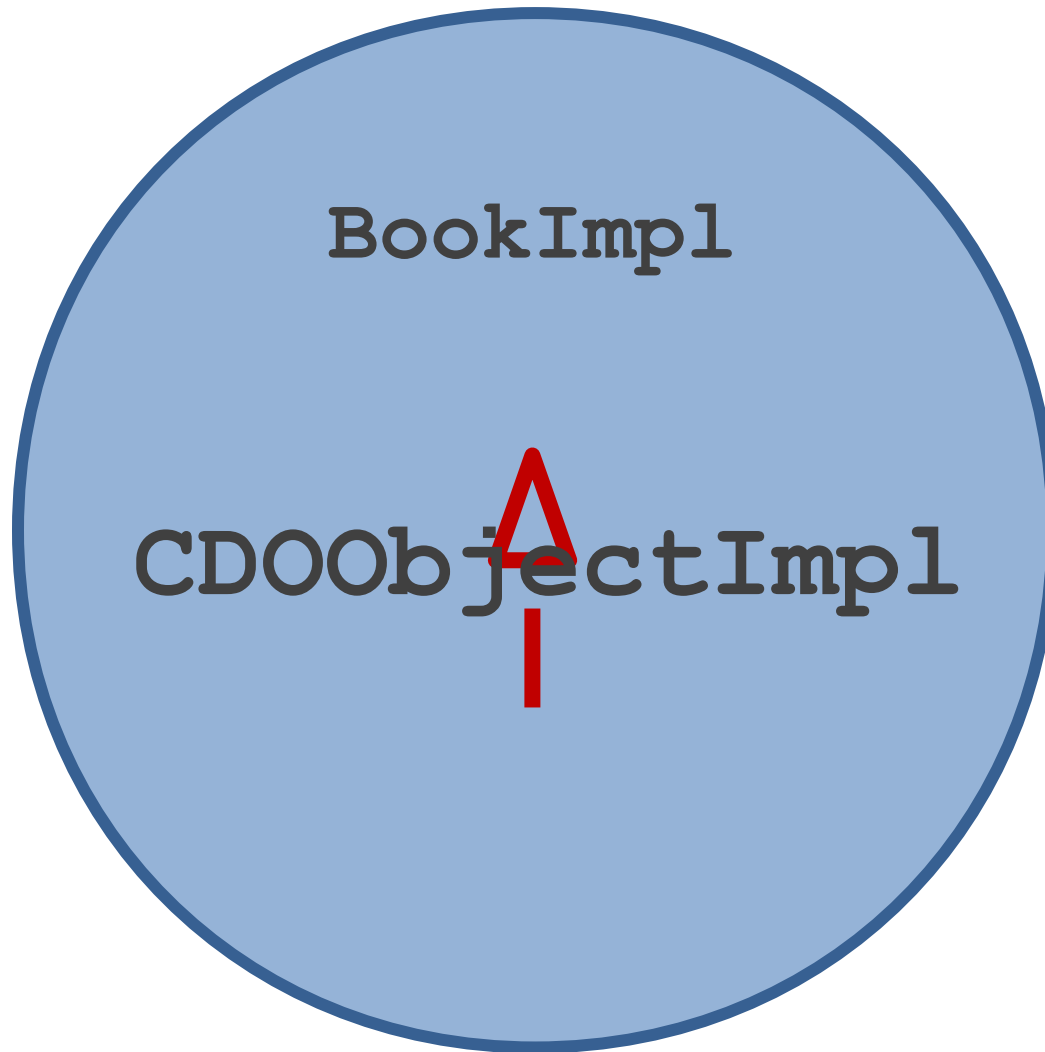


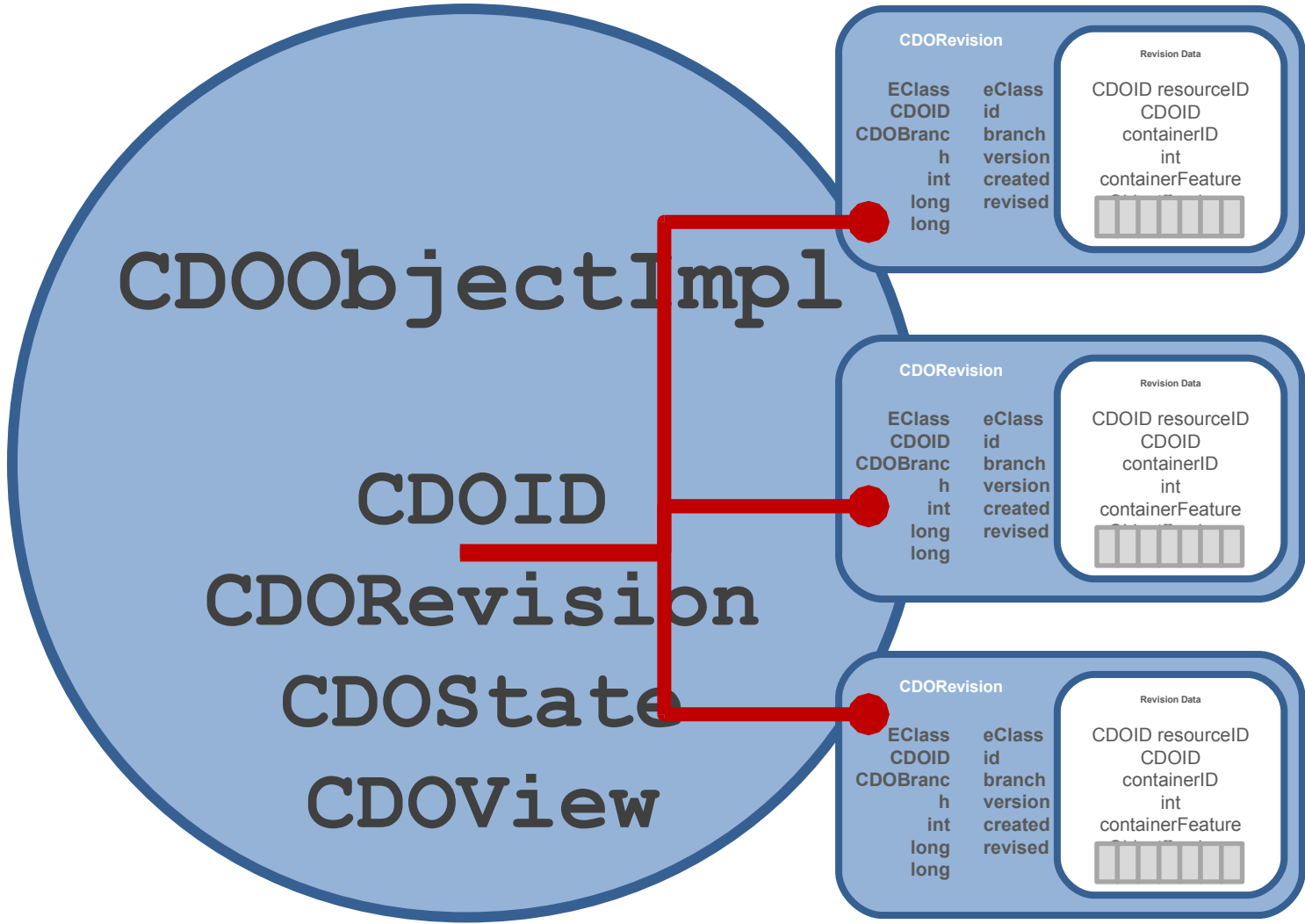


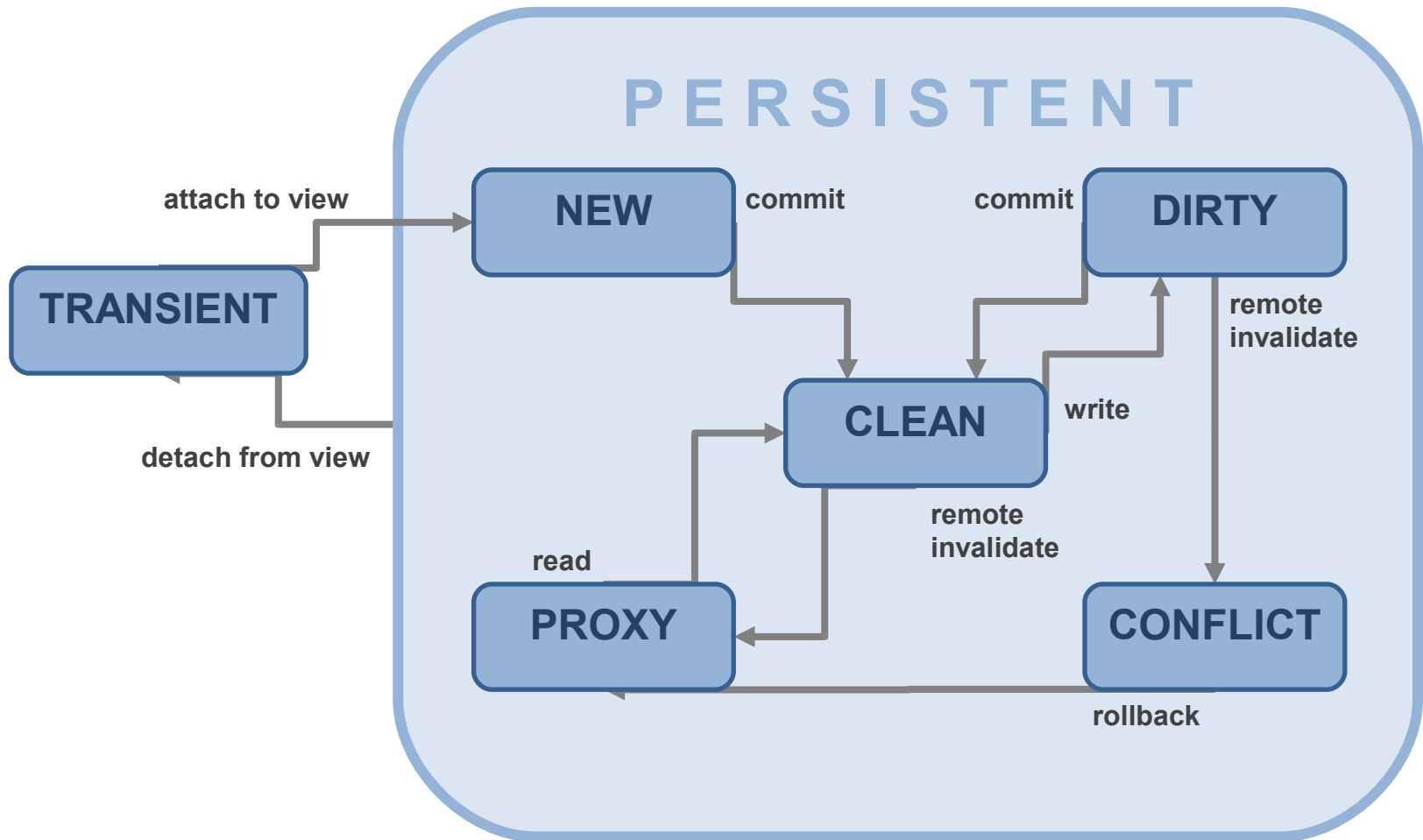
# Technical Challenges:

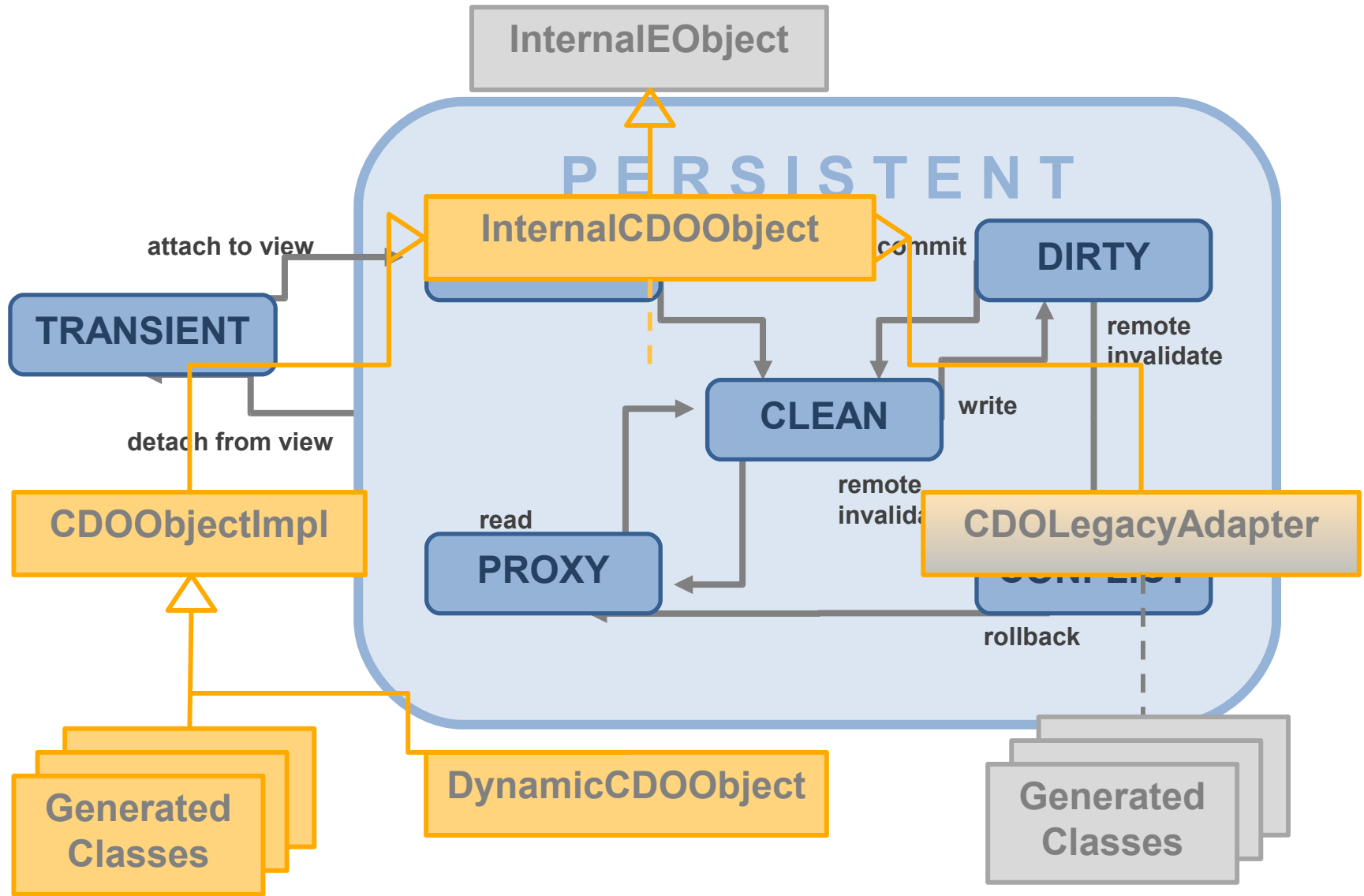
- **Transfer revisions over the network**
- **Swap revisions on remote invalidation**
- **Swap revisions when changing view time**
- **Swap revisions when changing view branch**
- **Make objects reclaimable by GC**

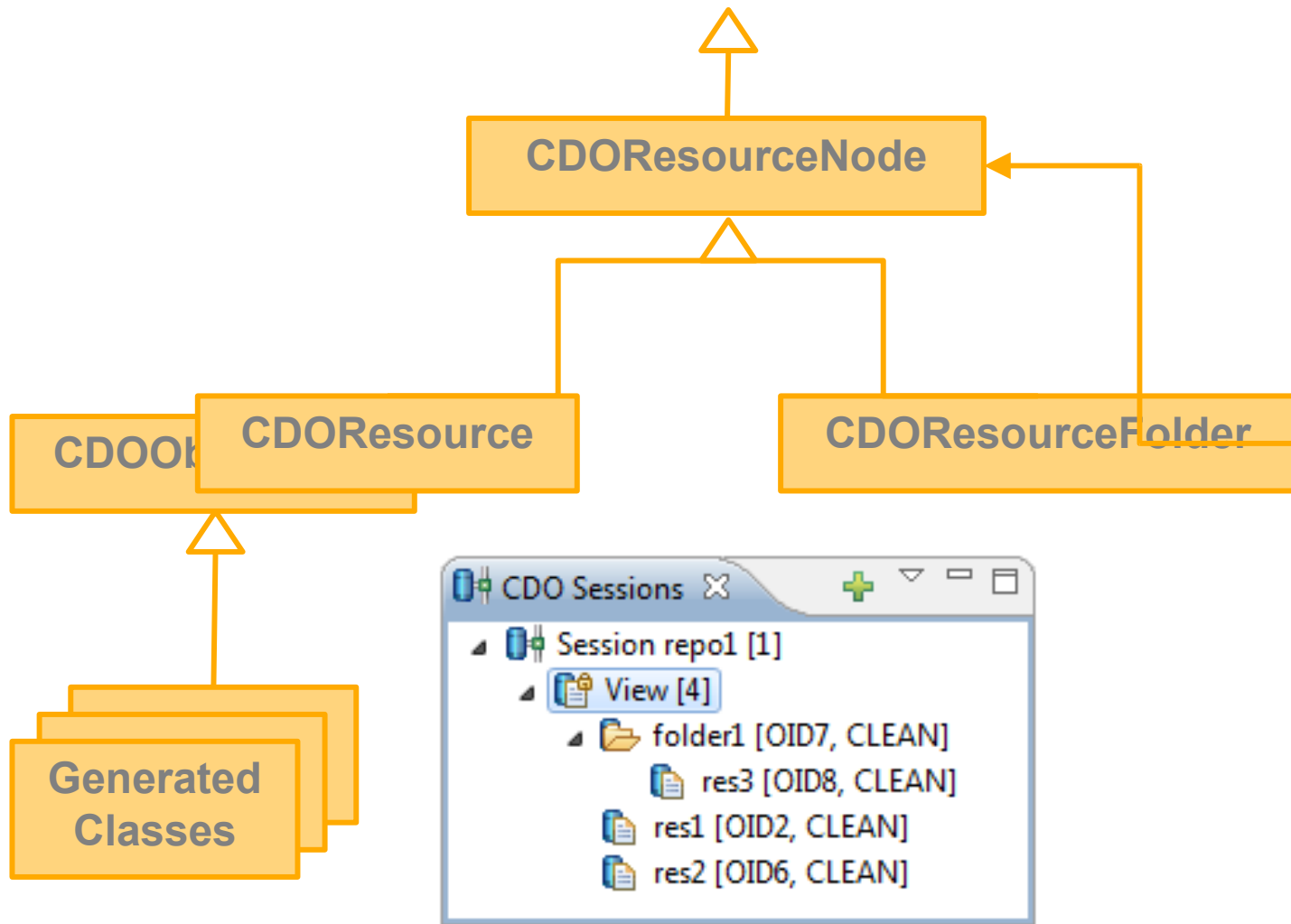


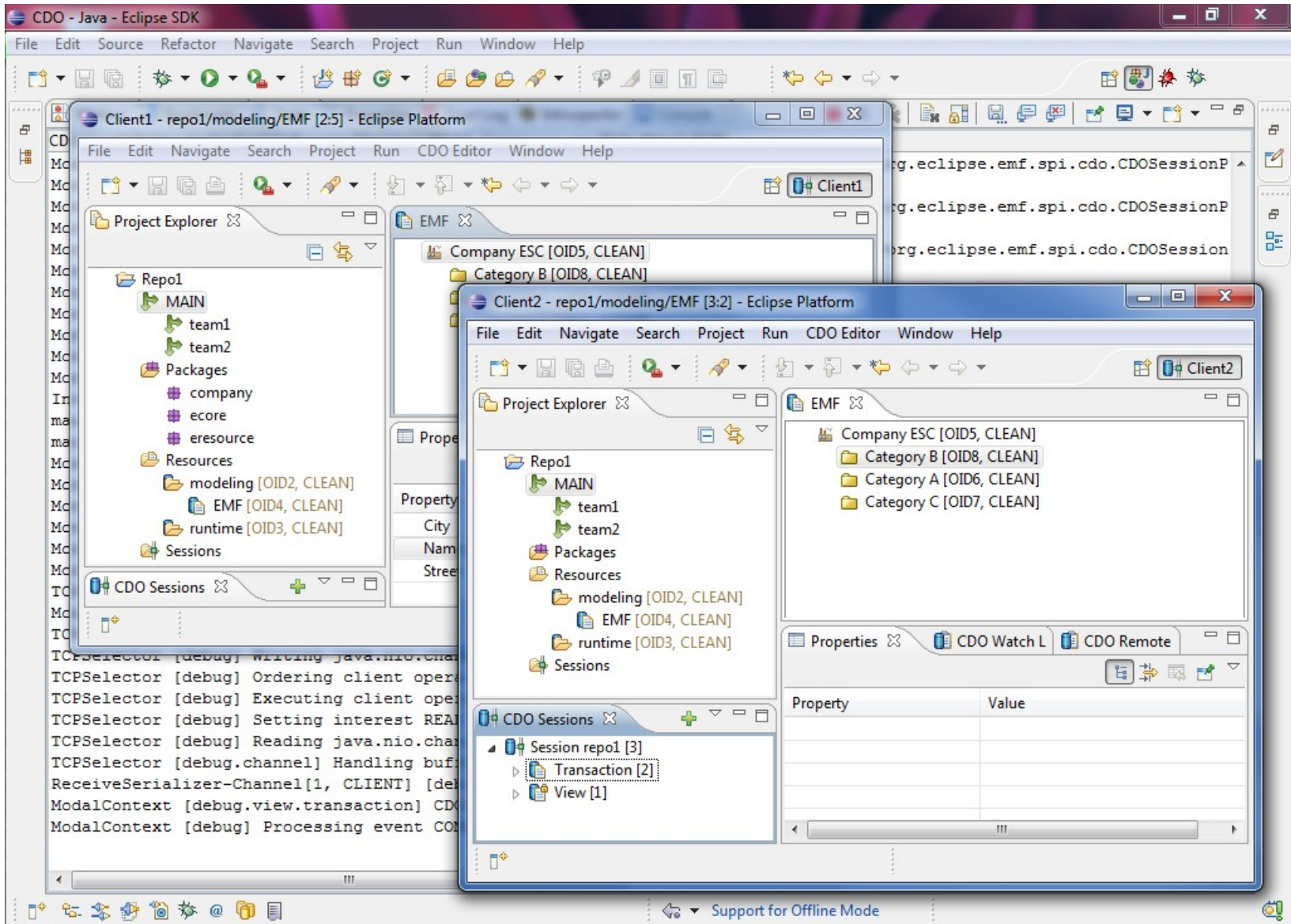




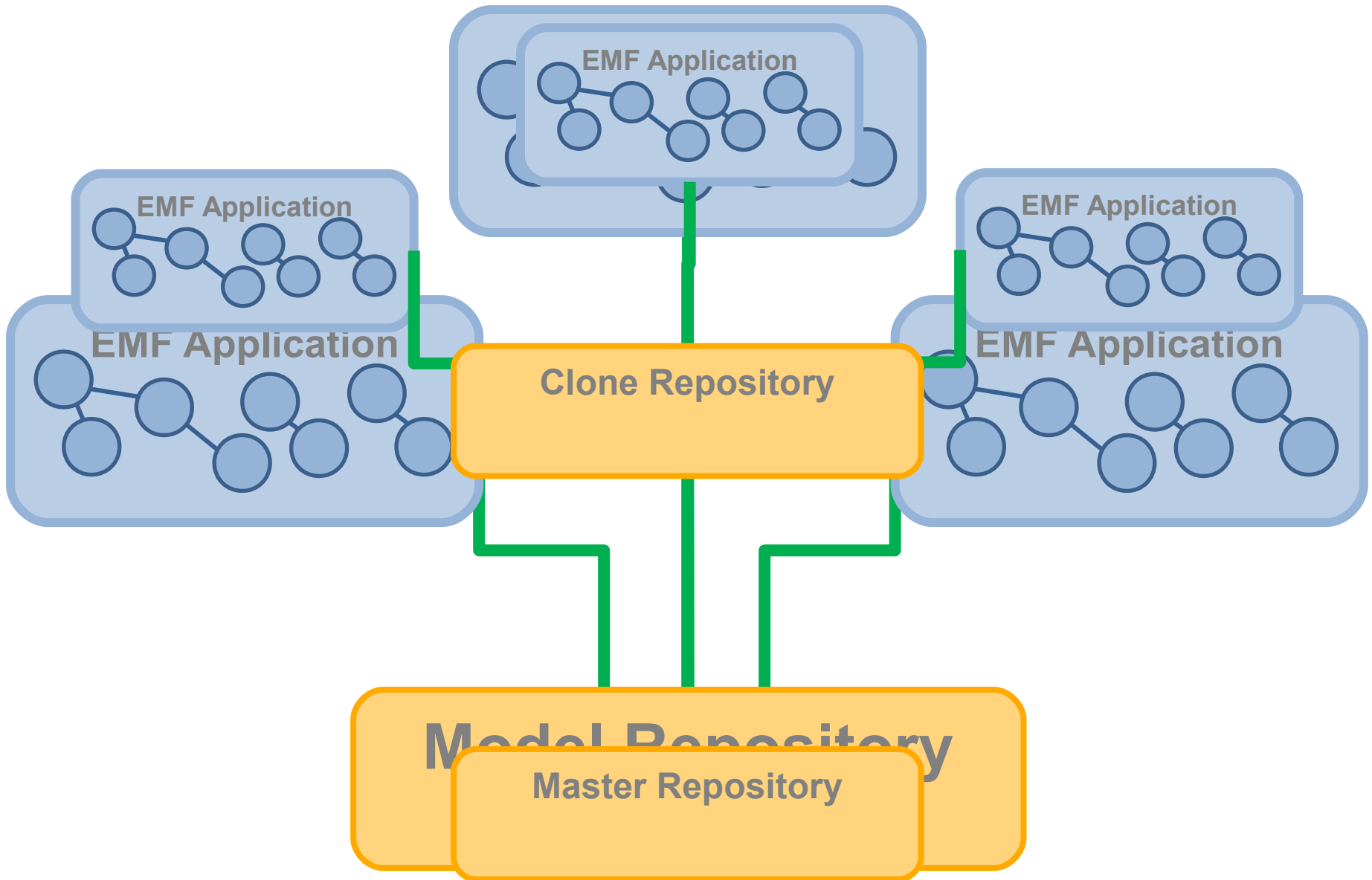


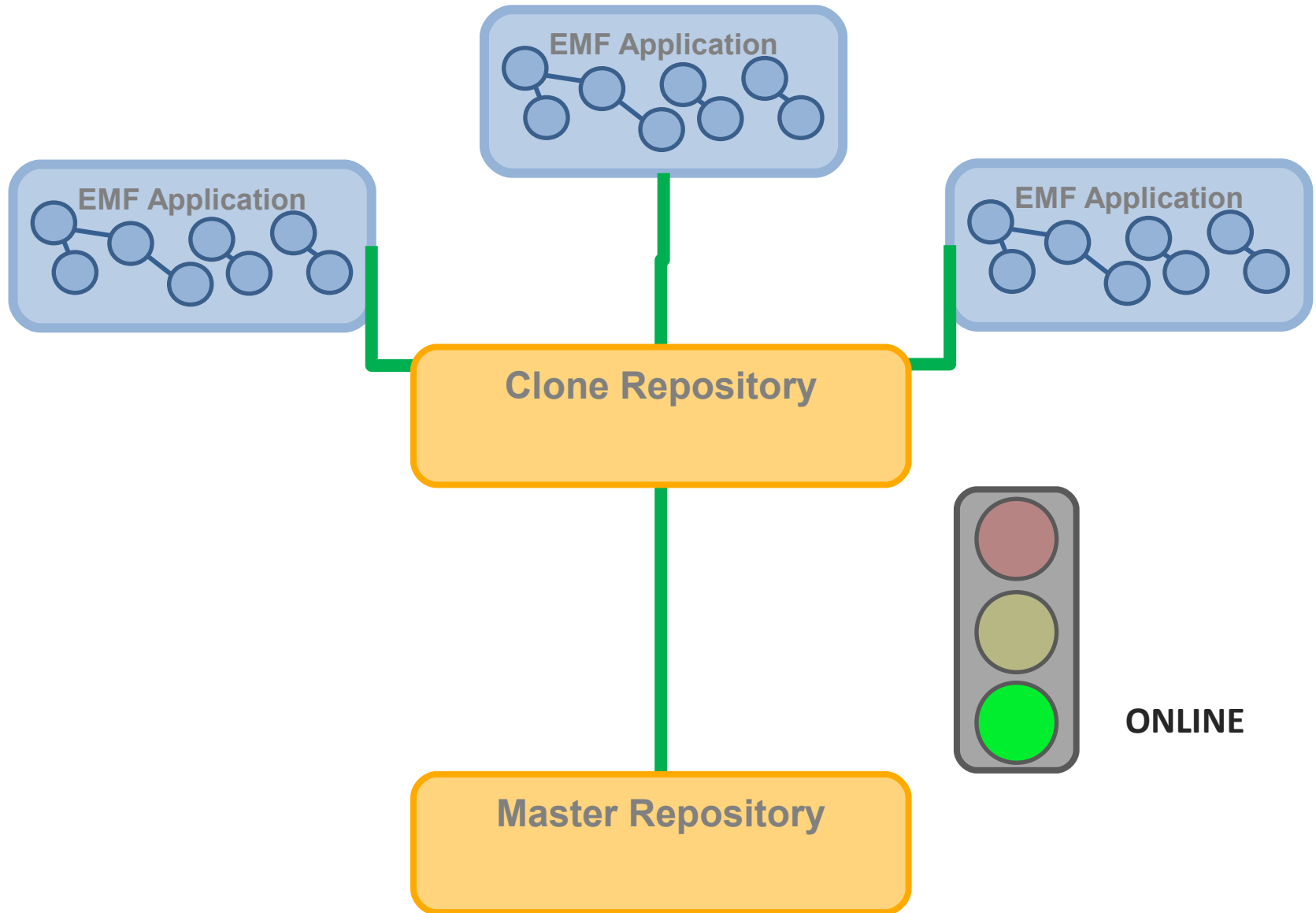


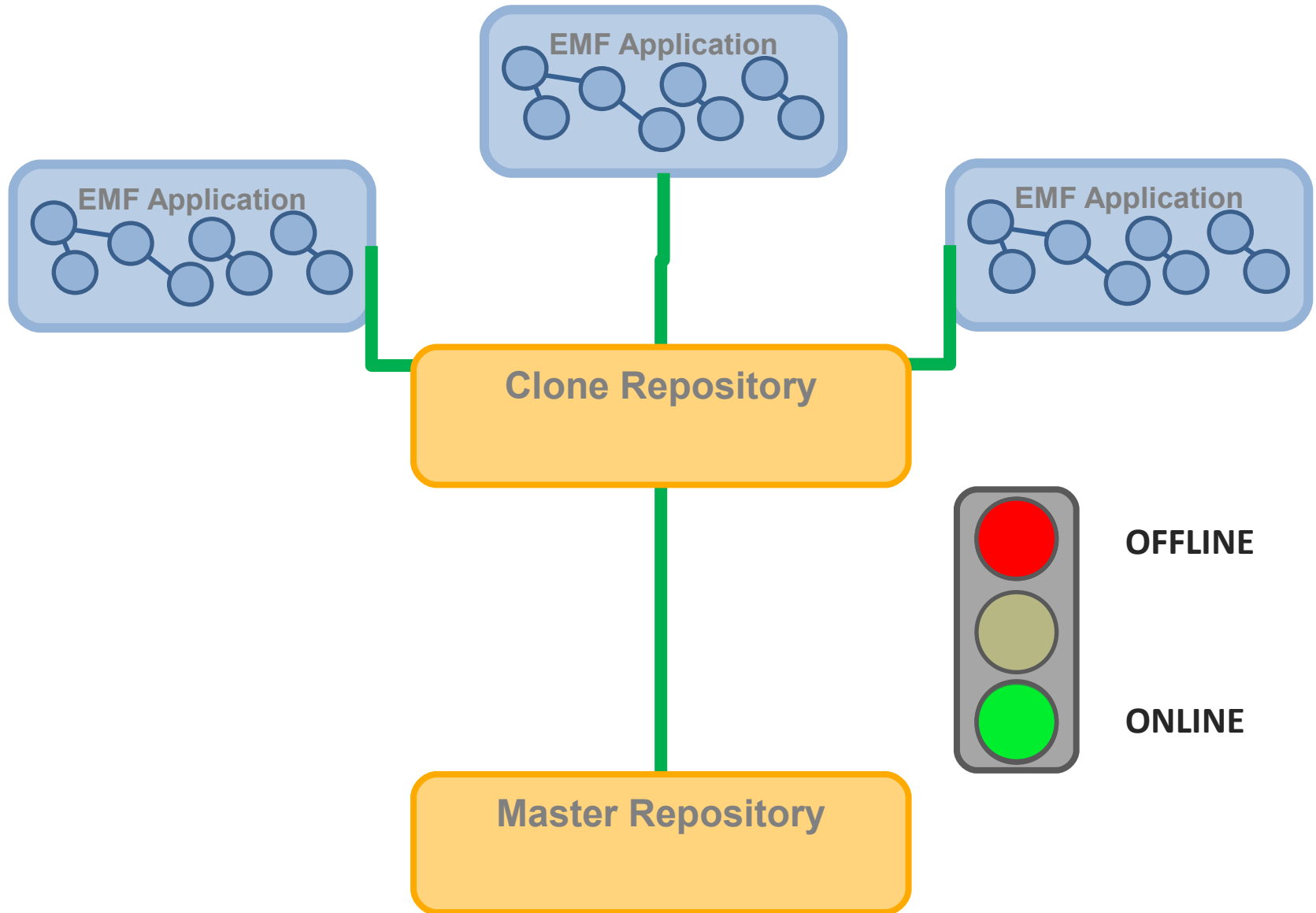


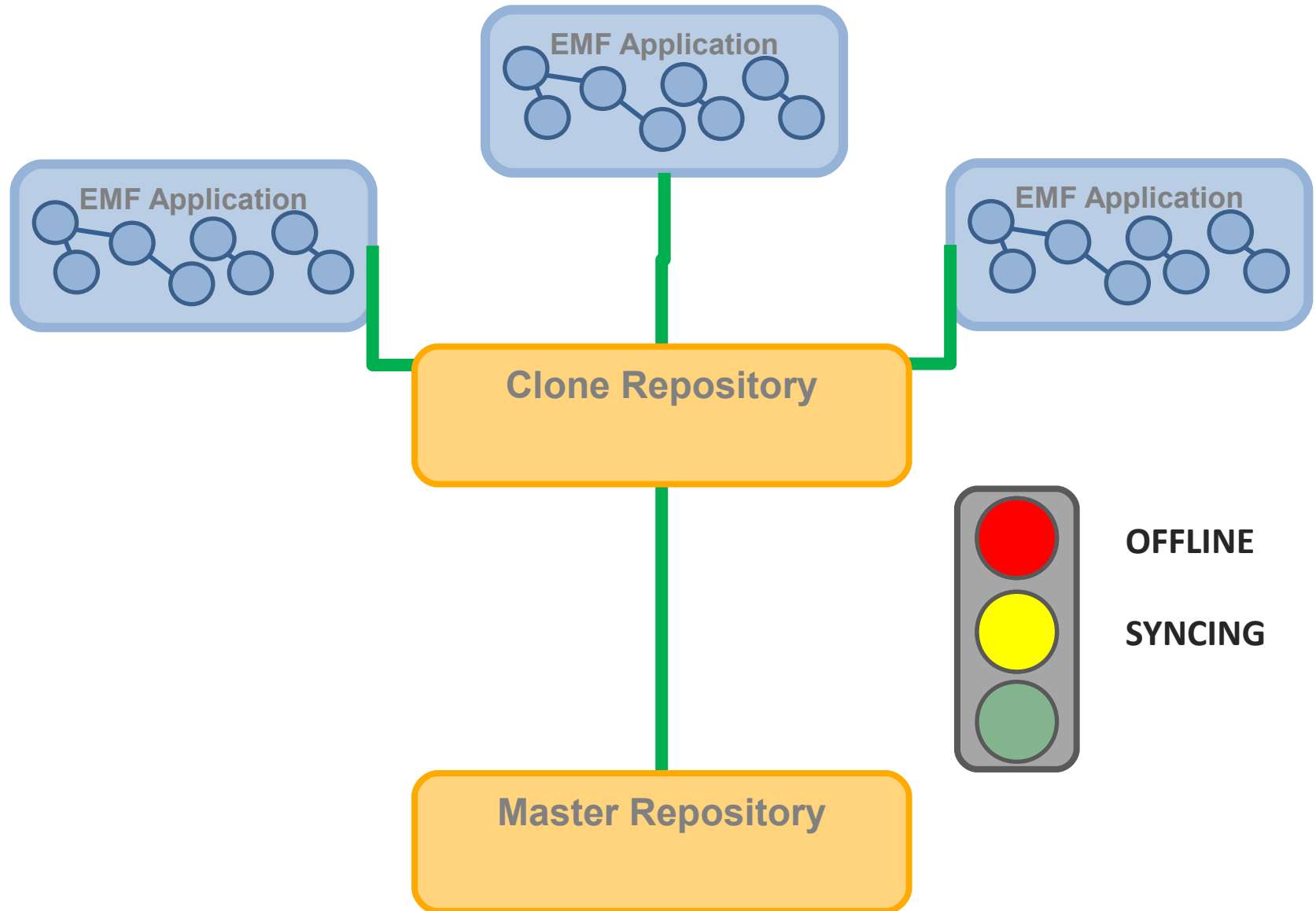


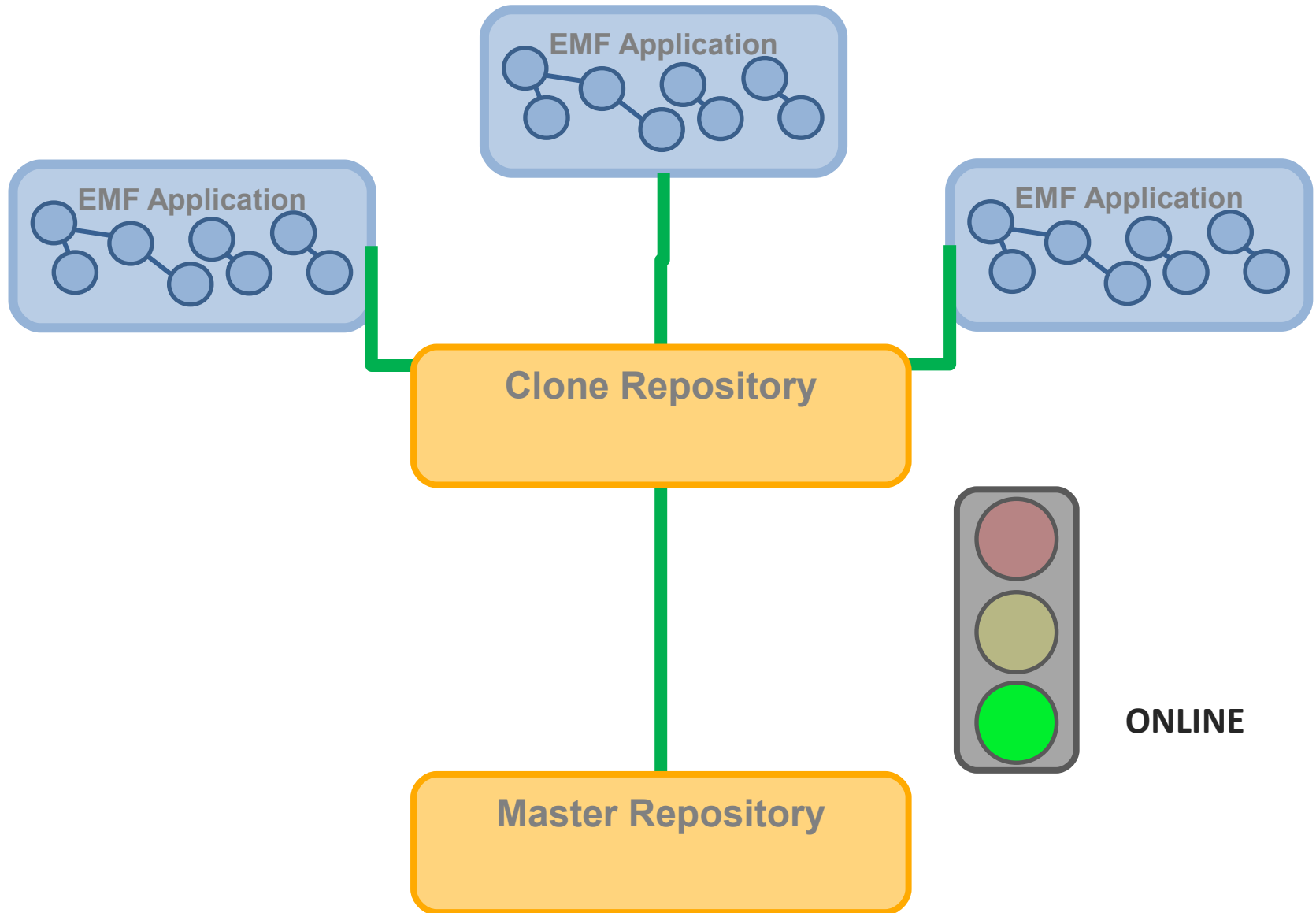
Scale, Share and Store your Models with CDO

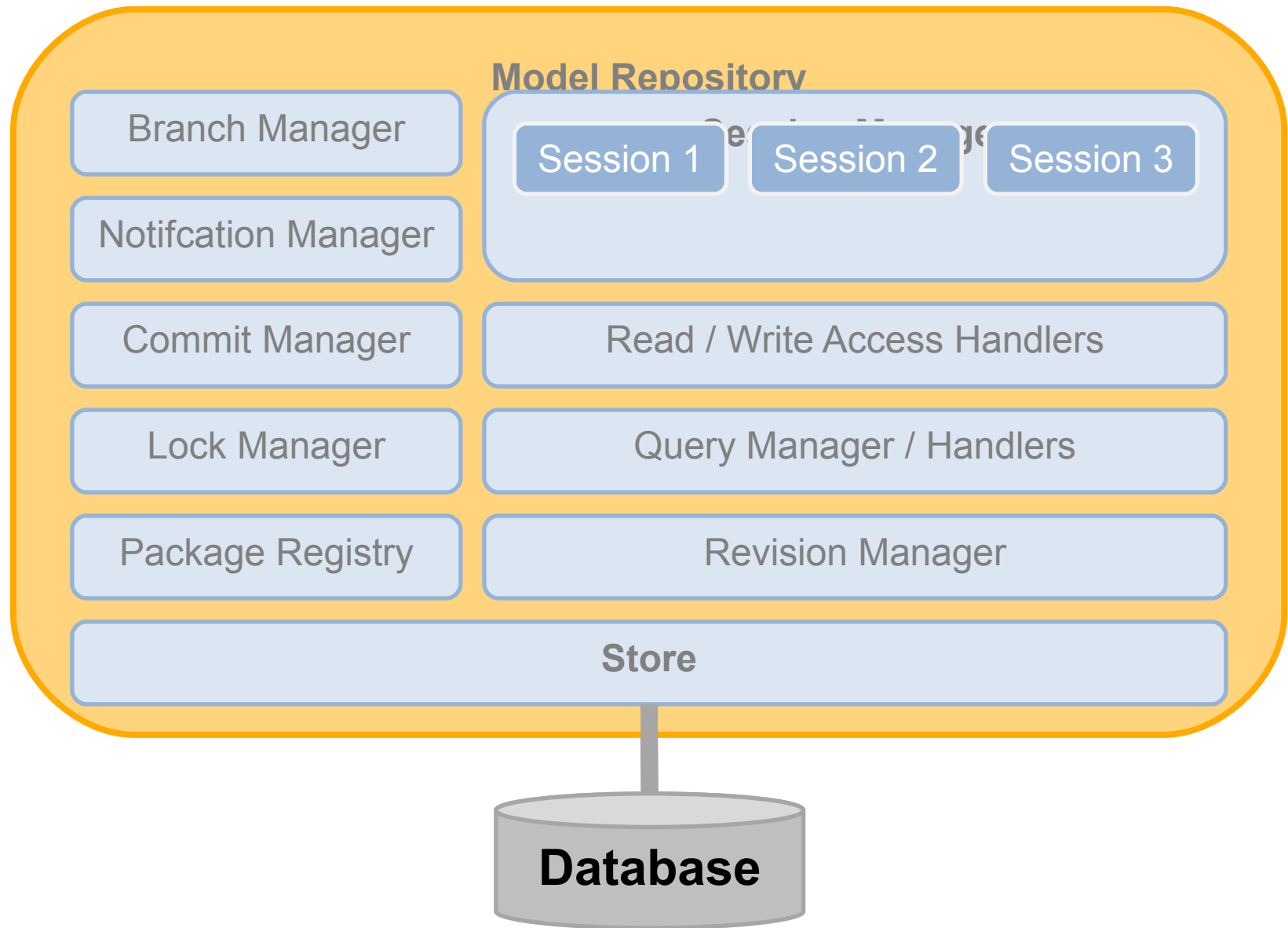


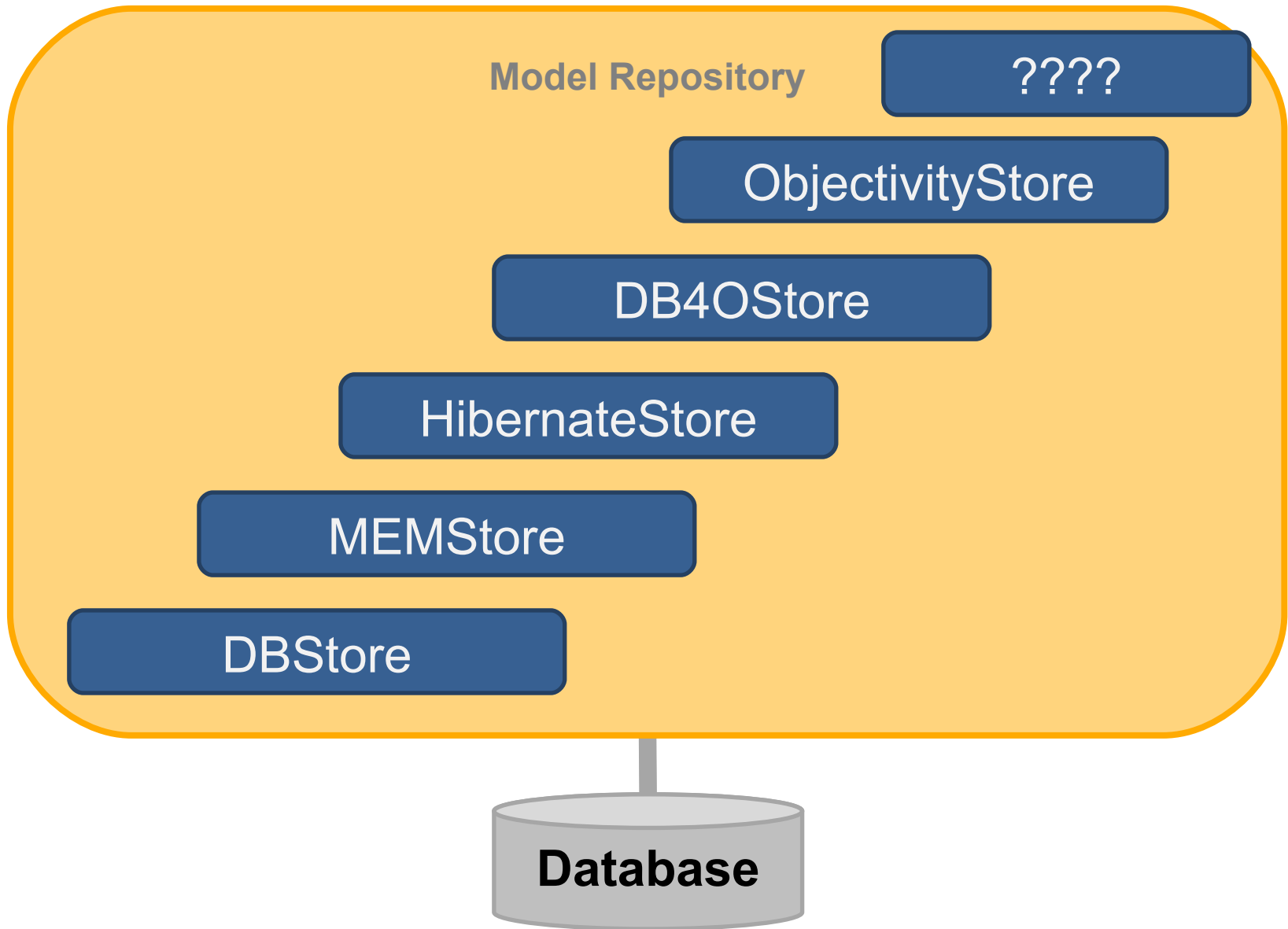


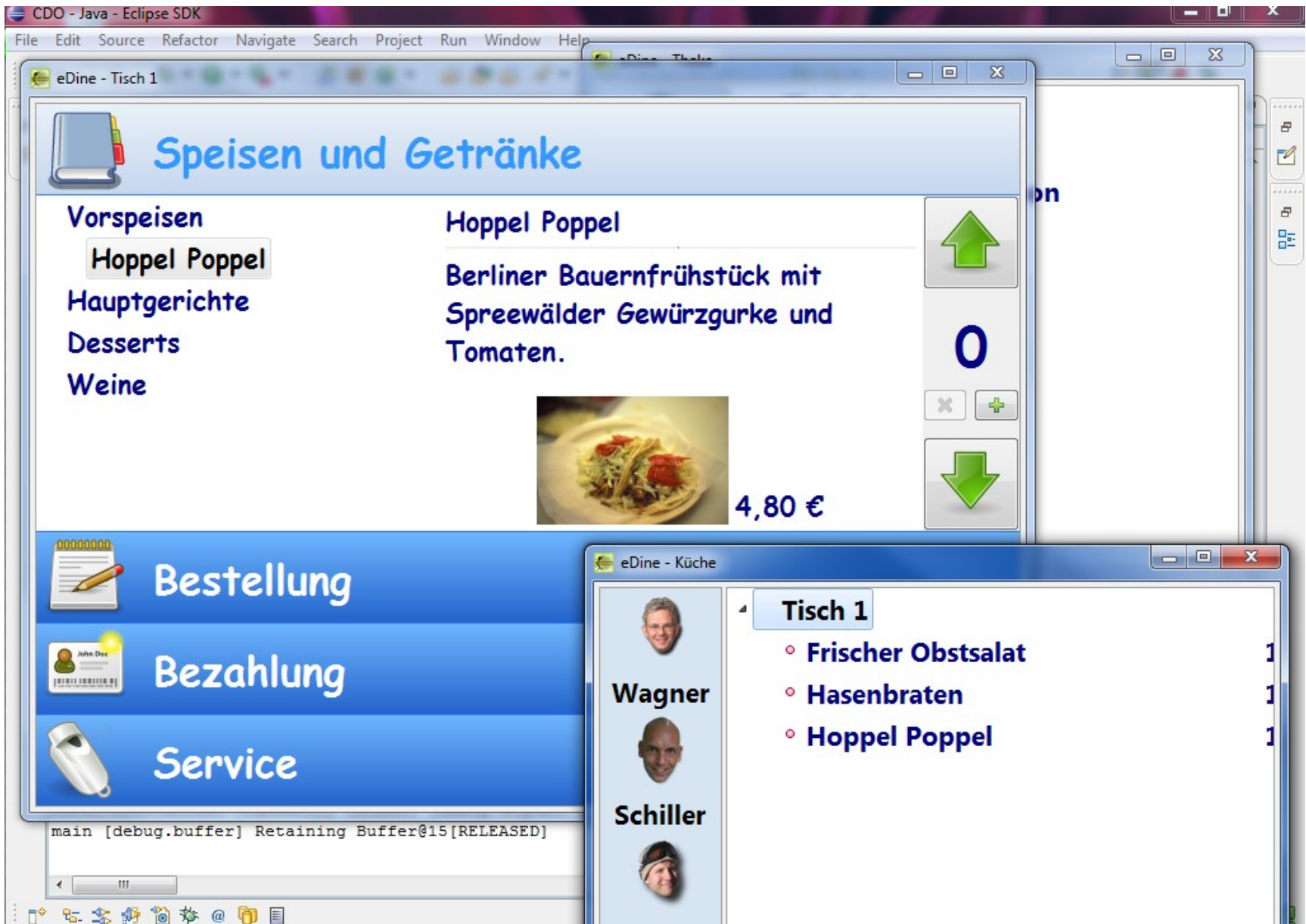












# CDO Core Features

# Distribution

- **Various ways to set up an IRepository**
  - XML config file, programmatically, Spring, ...
  - OSGi, stand-alone, ...
  - All components customizable
- **Various ways to open a CDOSession**
  - Net4j: TCP, HTTP, embedded, ...
  - CDO: embedded
  - Other transports possible
- **Offline mode coming soon**

# Persistence

- **Pluggable storage backend adapters (IStores)**
  - DBStore (CDO's own O/R mapper)
  - HibernateStore / Teneo
  - ObjectivityStore
  - DB4OStore
  - MEMStore
- **Changing the store type does not affect client applications!**

# Resources

- **A CDOResource is an EObject**
- **A repository contains CDOResourceNodes**
  - CDOResourceFolders
  - CDOResources
- **The resource tree is**
  - Navigable through EMF
  - Queryable through CDO

# Versioning

- **CDO supports record temporality**
  - Must be supported by IStore
  - Can be configured per IRepository
- **CDO supports branching**
  - Must be supported by IStore
  - Can be configured per IRepository
- **A CDOView provides consistent graphs**
  - From a particular branch
  - From a particular point in time

# Scalability

- **Lazy loading at object granule**
- **Lazy loading without container object**
- **Partial collection loading, chunking**
- **Adaptive prefetching**
- **Manual prefetching**
- **Automatic unloading at object granule**

# Queries

- **CDO includes a generic query framework**
  - Supports any query language
  - Supports named parameters
  - Supports synchronous execution
  - Supports asynchronous execution
- **Query language handlers can be**
  - plugged into an IRepository (OCL?, EMF-Q?, ...)
  - implemented by an IStore (SQL, HQL, custom, ...)

# Transactionality

- **Strong transactional safety at model-level**
- **Multiple transactions per session**
- **Multiple save points per transaction**
- **Rollback to any save point**
- **Commit with progress monitoring**
- **Hooks for custom transaction handlers**
- **Conflict detection and fail-early-transactions**
- **Pluggable conflict resolvers**
- **Explicit read/write locking on object granule**

# Collaboration

- **Passive Updates**
  - Asynchronous commit notifications
  - Invalidation of objects, lazy reload if needed
  - Can be switched off per session
- **Change subscriptions**
  - Asynchronous change delta delivery
  - Registration with repository per object
  - Automated through pluggable adapter policies
- **Remote session manager**

Scale, Share, and Control with D

© 2010 by Eike Stepper, Berlin, Germany. Made available under the EPL v1.0

# Integration

- **Integrates with EMF at the model level, not at the edit- or UI-level.**
- **Uninvasive to the .ecore file.**
- **Best results with regenerated models (native)**
- **Regeneration not needed (legacy)**
- **Dynamic models supported**
- **Multiple repositories per ResourceSet**
- **External references**

# Dawn – Rise of Graphical Collaboration

The image displays two windows side-by-side. The left window is a web browser showing a URL: `http://localhost:8080/DawnServer/showDiagram.do?projectname=my.classdi`. Below the browser, there are navigation tabs for 'Home', 'Projects', 'Users', 'Roles', and 'myProjects'. The main content area shows a UML ClassDiagram with the following elements:

- MyInterface** (Interface): `public printName():void`
- MySubClass** (Class): Inherits from MyInterface.
- MyClass** (Class): Inherits from MyInterface. Attributes: `private name:String`. Operation: `public getName():String`.
- MyAssociation** (Association): Connected to MyClass.
- MyComposition** (Composition): Connected to MyClass.
- MyAggregation** (Aggregation): Connected to MyClass.

The right window is the Eclipse IDE, titled 'Java - ClassDiagram/my.classdiagram\_diagram - Eclipse SDK'. It shows the same UML ClassDiagram in a graphical editor. The diagram elements are:

- MyInterface** (Interface): `public printName():void`
- MySubClass** (Class): Inherits from MyInterface.
- MyClass** (Class): Inherits from MyInterface. Attributes: `private name:String`. Operation: `public getName():String`.
- MyAssociation** (Association): Connected to MyClass.
- MyComposition** (Composition): Connected to MyClass.
- MyAggregation** (Aggregation): Connected to MyClass.

The Eclipse IDE interface includes a menu bar (File, Edit, Diagram, Navigate, Search, Project, Tomcat, Run, Window, Help), a toolbar, a palette on the right with categories like 'Node' (Class, Interface, AnAttribute, AnOperation) and 'Connecti...' (inherits, implements, association, aggregation, composition), and a bottom status bar.

Scale, Share and Store your Models with CDO

© 2010 by Eike Stepper, Berlin, Germany. Made available under the EPL

# Dawn – Rise of Graphical Collaboration

- Conflict handling
  - Dawn provides detection and handling mechanisms for conflicts
  - It will build on the CDO conflict mechanisms and provide flexible and intuitive UI to handle conflicts
  - Conflicts are displayed inside the diagram editor. Conflicts that cannot be visualized inside the editor will be show in a special view (Dawn Conflict View)
- Locking
  - Dawn will support locking on different hierarchy levels

in the GMF diagram

# Dawn – Rise of Graphical Collaboration

- Do not change existing code
  - A dynamic design and a flexible generator will make it possible to “collaborate” existing GMF editors even if the source is
  - Existing editor do not need to modified
- Firewall transparency mode
  - Allows to operate from within restricted networks
  - This mode will use a web-based protocol on CDO
- Network independence (Offline Mode)
  - Using one of the latest CDO features (offline support)



© 2012 PFC. All Rights Reserved.

Scale, Share and Store your Models with CDO

© 2010 by Fike Stepper, Berlin, Germany. Made available under the EPL v1.0