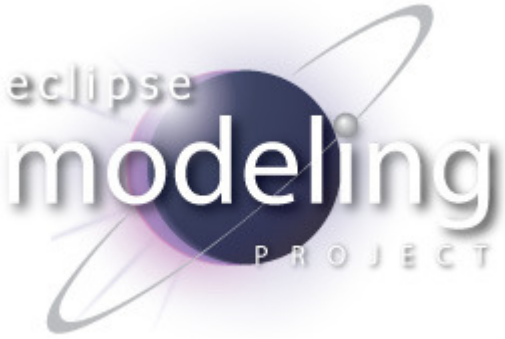


Eike Stepper

stepper@esc-net.de
http://www.esc-net.de
http://thegordian.blogspot.com

Berlin, Germany

CDO Model Repository

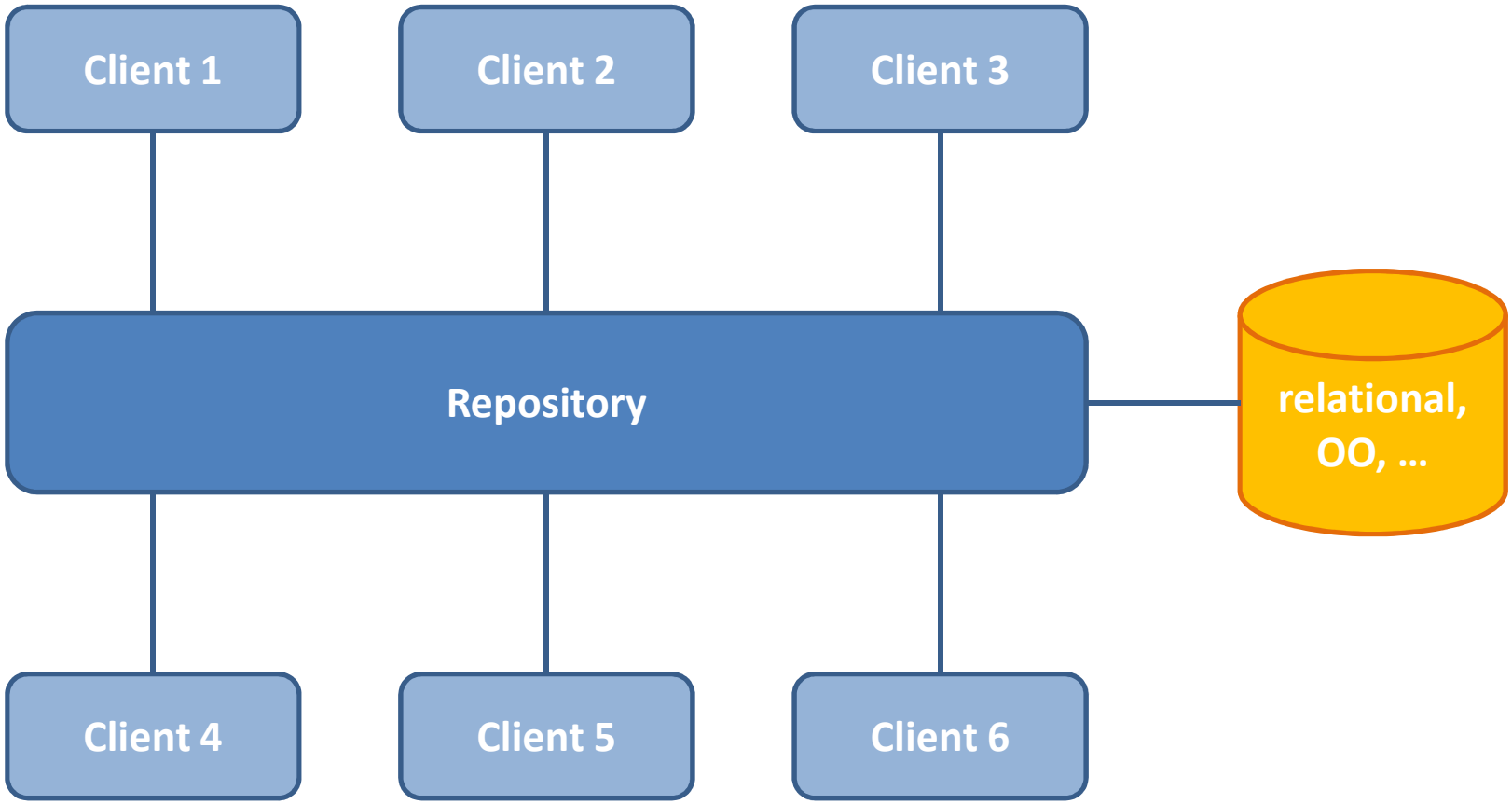
Where Models Live

SAP Modeling Meeting
Wednesday, January 27, 2010

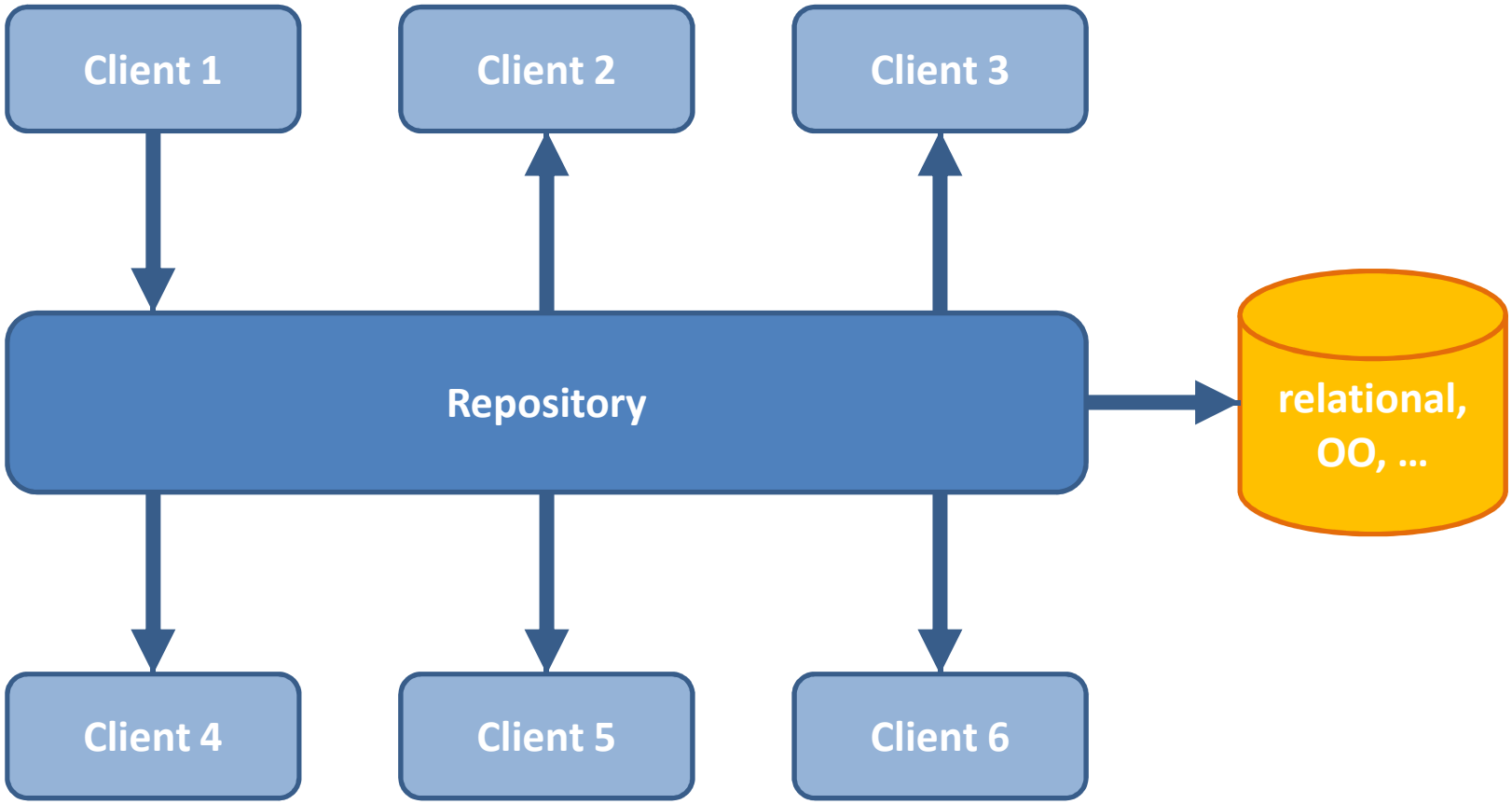
Agenda

- Overview
- Distribution
- Persistence
- Resources
- Versioning
- Scalability
- Queries
- Transactionality
- Collaboration
- Integration
- Usage Example
- Related TODOs

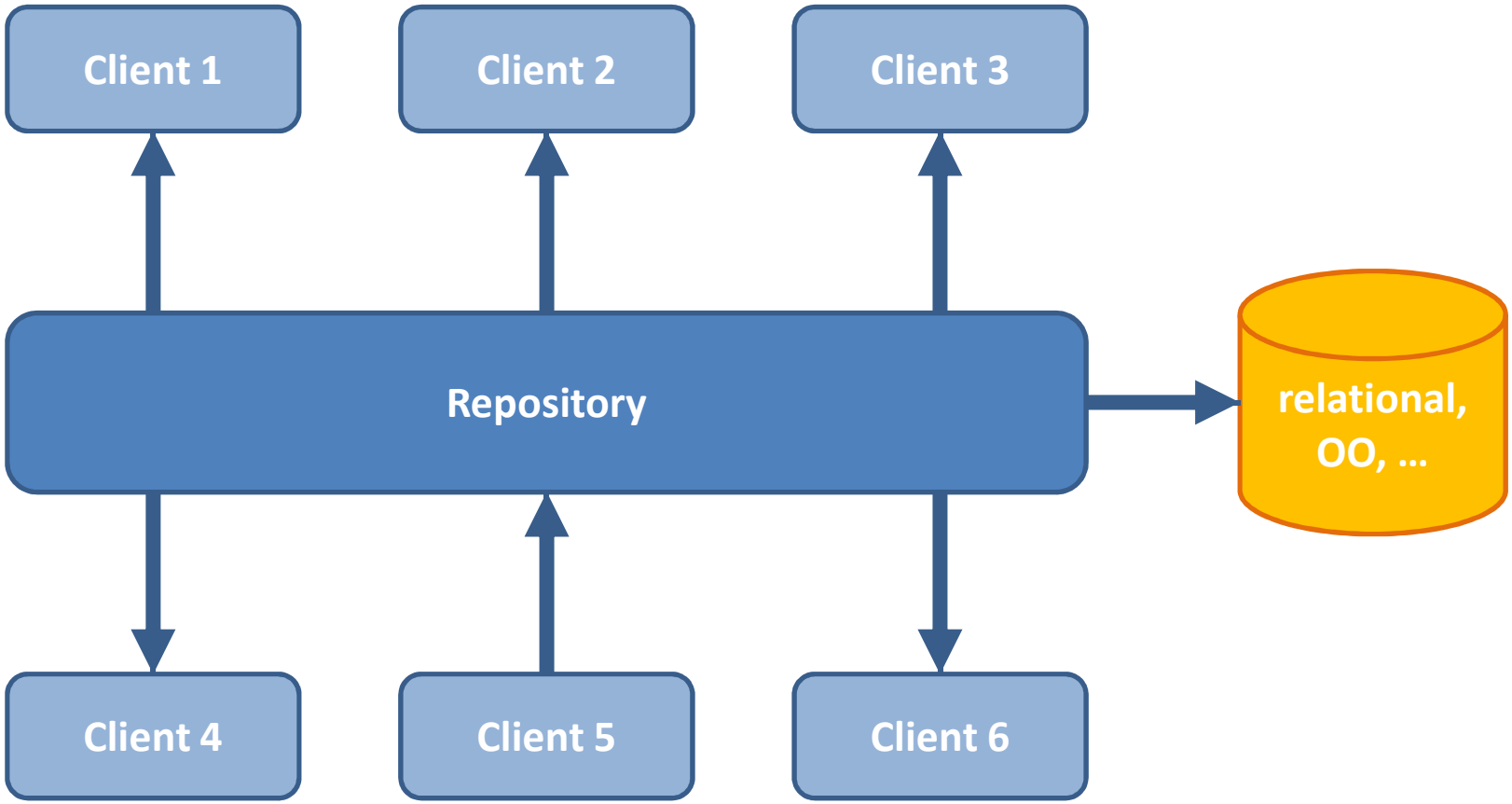
Overview



Overview



Overview



Distribution

- **Various ways to set up an IRepository**
  - XML config file, programmatically, Spring, …
  - OSGi, stand-alone, …
  - All components customizeable
- **Various ways to open a CDOSession**
  - Net4j: TCP, HTTP, embedded, …
  - CDO: embedded
  - Other transports possible
- **Offline mode coming soons**
  - Cloned and sync'ed repository, normal sessions

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 (coming soon)**
  - 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
- XA transactions to multiple repositories

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**
  - Notifies about state of other sessions
  - Supports sending/receiving of arbitrary messages

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

```
CDOSession session = config.openSession();
CDOBranch teamBranch = session.getBranchManager().getBranch("MAIN/team1");
CDOBranch branch = teamBranch.createBranch("stepper");

CDOTranaction transaction = session.openTransaction(branch);
CDOResource resource = transaction.getResource("/client1/facility3");
resource.getContents().add(facility);

CDOCommit info = transaction.commit(progressMonitor);
System.out.println(info.getUserID(),
                   info.getComment());

CDOView view = session.openView(info.getBranch(), info.getTimeStamp());
CDOResource readOnlyResource = view.getResource("/client1/facility3");
Facility object = readOnlyResource.getContents().get(0);
System.out.println(object.cdoID(),
                   object.cdoState(),
                   object.cdoView());

CDORevision revision = f.cdoRevision();
System.out.println(revision.getTimeStamp(),
                   revision.getRevised(),
                   revision.getVersion());
```

Relevant TODOs

- **Model evolution**
  - I.e. instance migration
  - Conceptually and technically complex
- **Access control**
  - I.e. authorization
  - Comparingly easy
- **Composite views**
  - I.e. objects from different branch points (tags)
  - Probably medium complexity
- **Native design time models (non-DSLs)**
  - I.e. Ecore, UML2, GMF Notation
  - Medium complexity, maintenance challenge
- **Common query language**

THE END