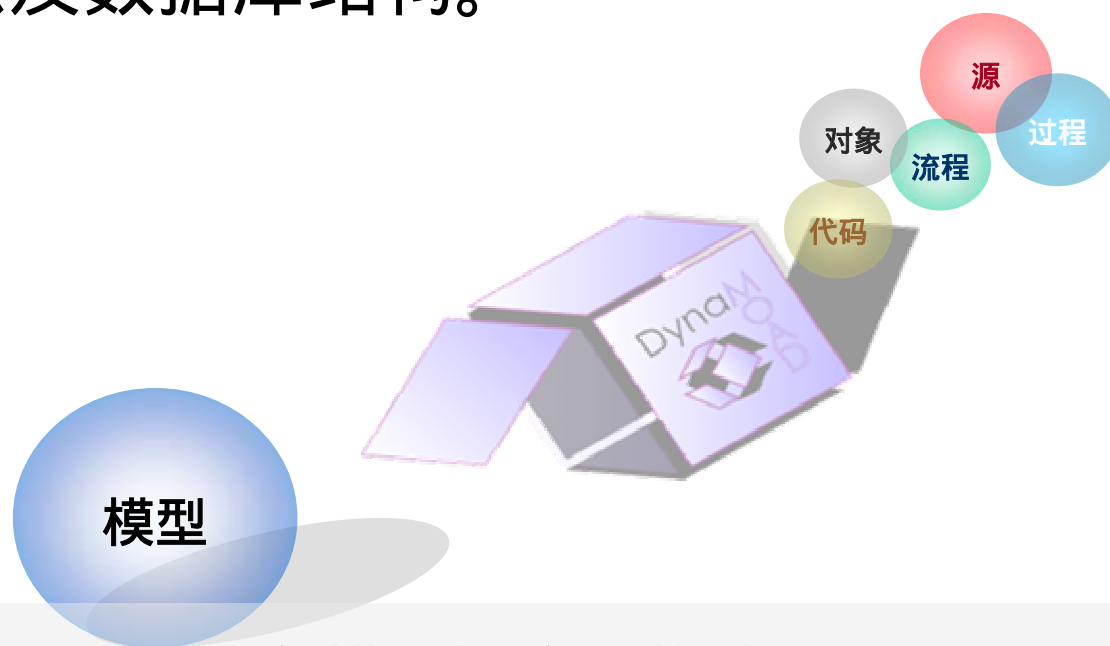




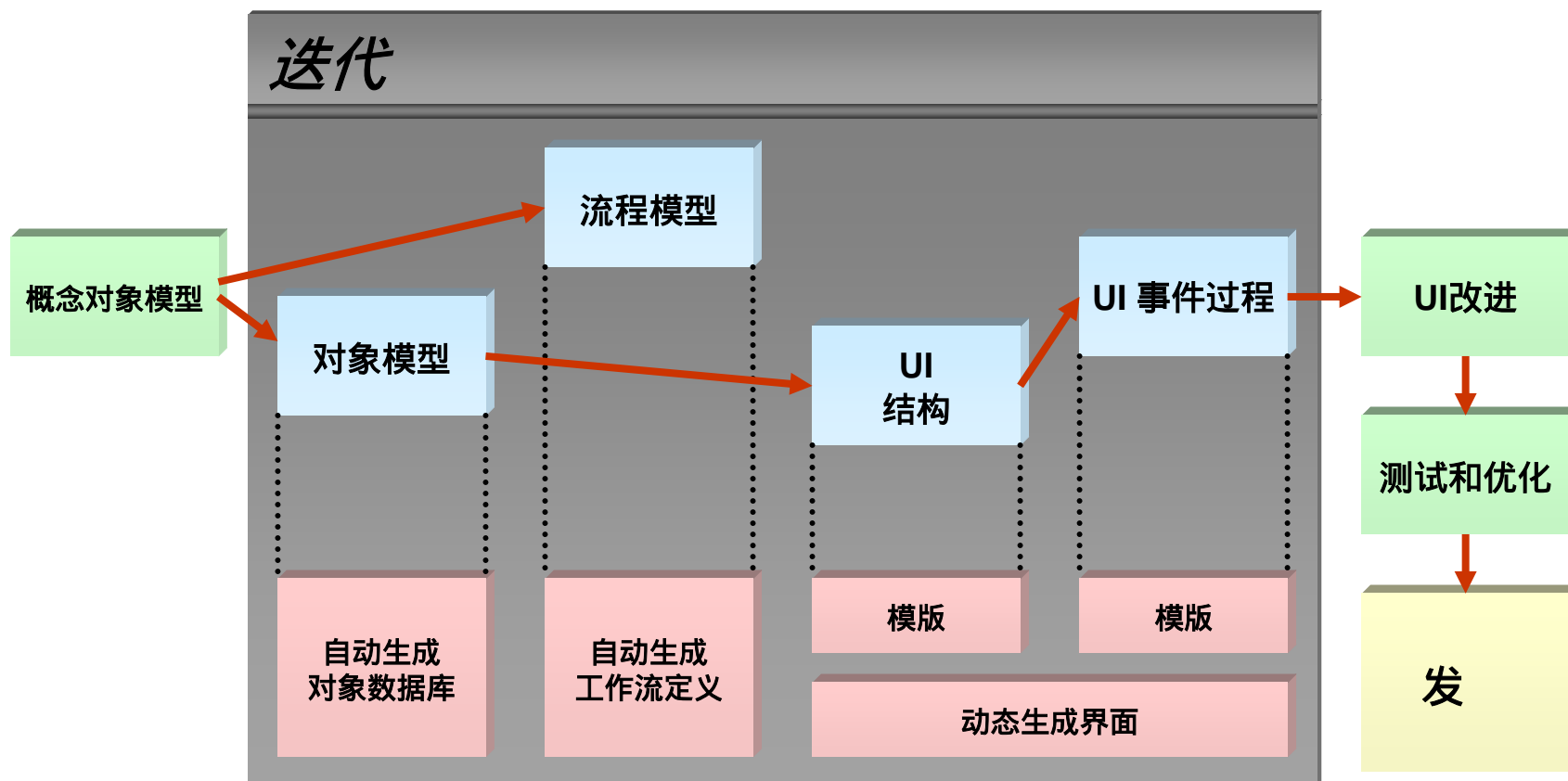
DynaMOAD 技术概要

- 为用户和实施人员提供了有效和快速构建PDM/PLM的环境。
- 有效地将业务模型转变成为应用系统，无需关注源代码以及数据库结构。

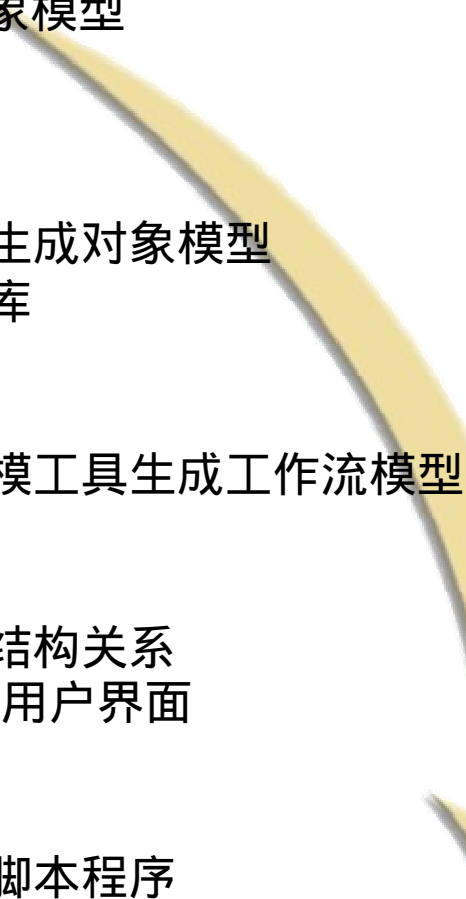


面向模型的应用构建

建模过程



建模过程

- 概念对象模型 (COM)
 - 使用UML, IDEF...等方式建立概念对象模型
 - 建立活动图和状态图
 - 对象模型 (OM)
 - 使用对象建模工具根据概念对象模型生成对象模型
 - 完成对象建模后, 自动生成对象数据库
 - 流程模型(PM)
 - 使用根据活动图和状态图使用流程建模工具生成 workflow 模型
 - 用户界面结构设计 (UIS)
 - 生成用户界面并定义用户界面之间的结构关系
 - 自动生成Java/Swing, Web Page...等用户界面
 - 用户界面的事件过程(UIEP)
 - 定义用户界面的事件, 并编写相应的脚本程序
- 

处理客户的变更需求

分析

- 分析客户的变更请求对对象模型和流程模型的影响
- 报告模型的变更（描述增加的功能和对象模型摘要）



决策

- 决策增加或修订功能的方案
- 决策补充的脚本以及外部应用



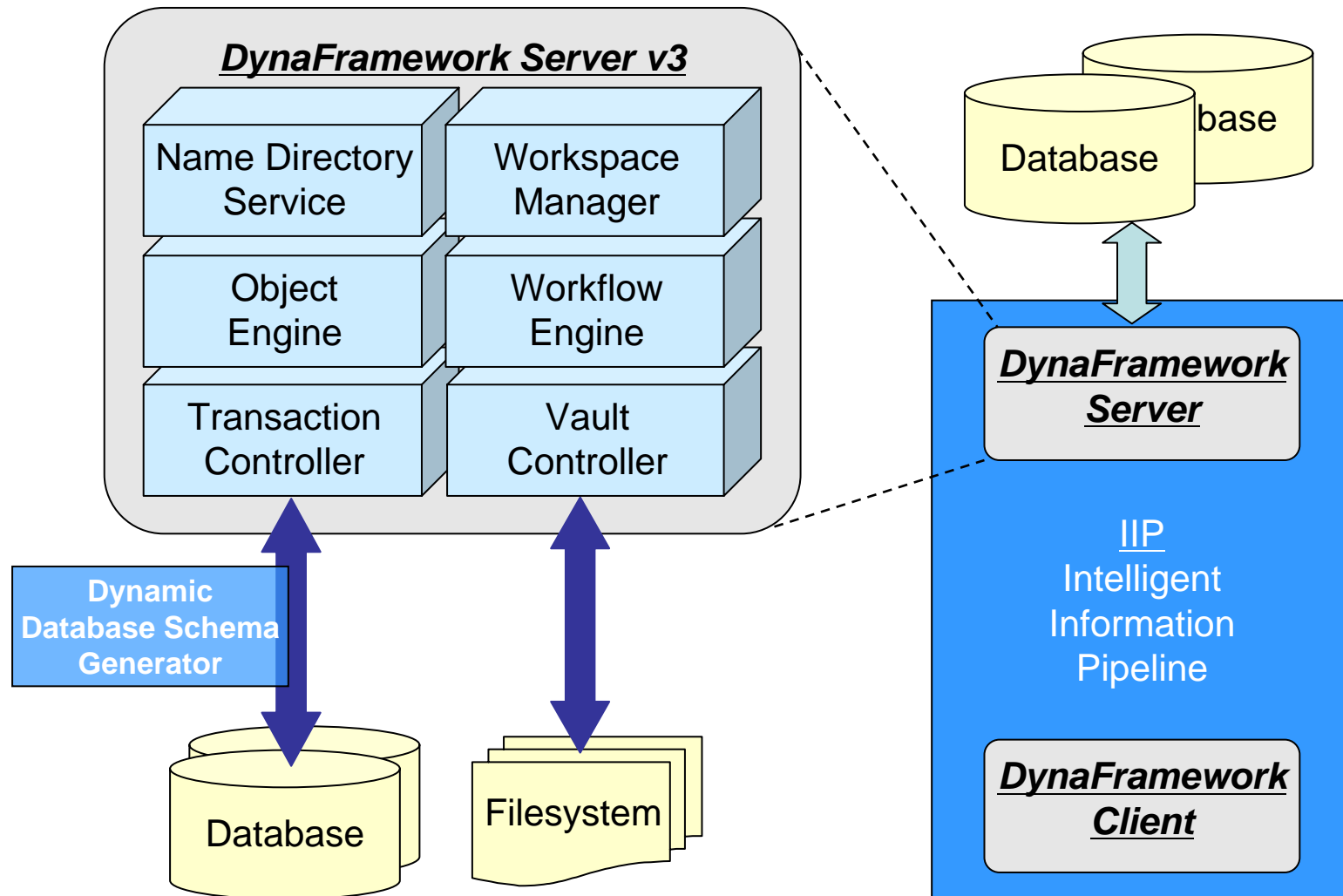
应用

- 使用DyanMOAD的各种建模工具实现新的模型
- 修订用户界面
- 基于系统提供的API开发脚本以及外部应用

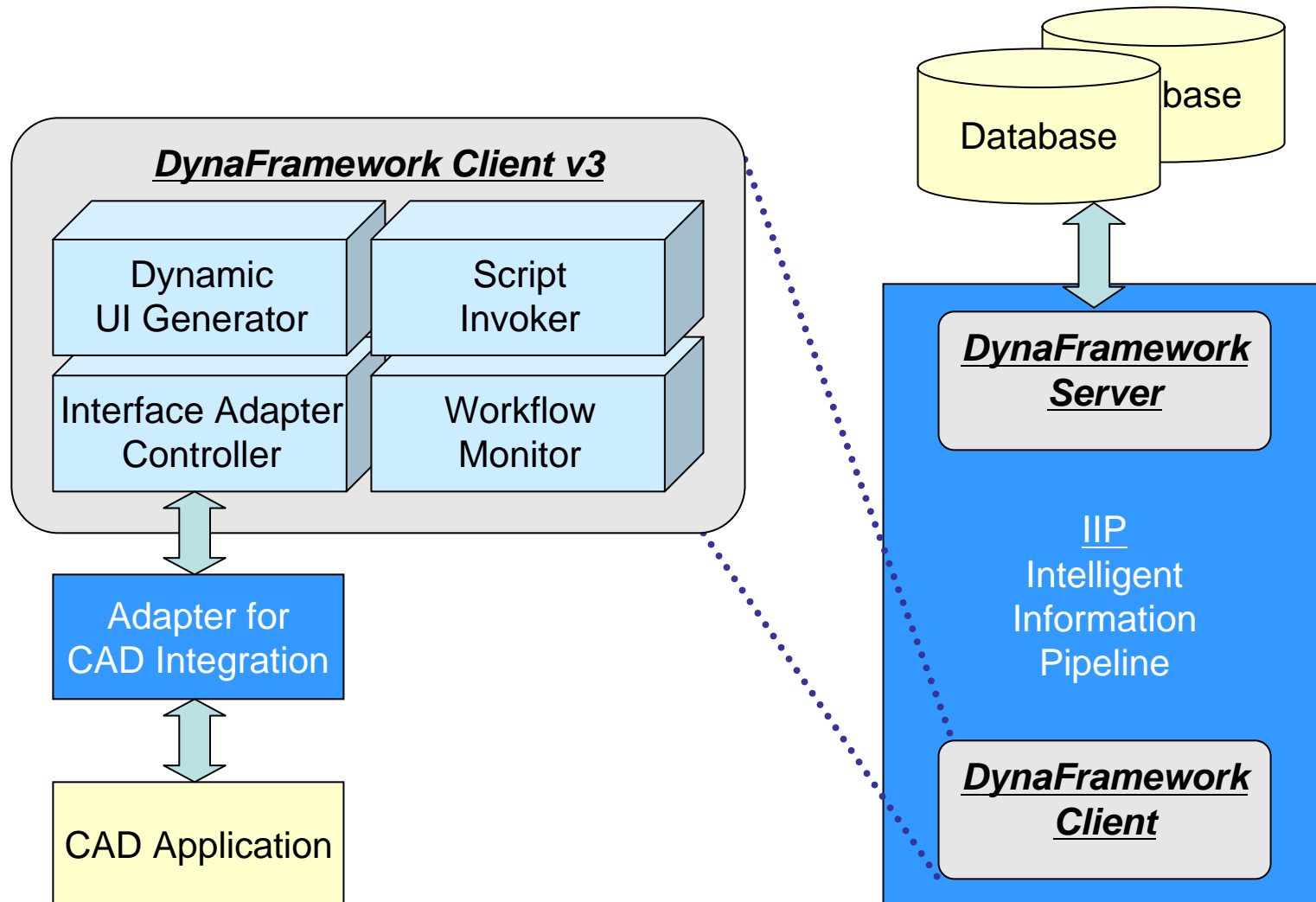
脚本和外部应用

- 脚本
 - 内置3种类型的脚本解释器
 - Java Beanshell, TCL, Python
- 外部应用
 - 支持使用DyanMOAD开发Java的独立应用。
 - 支持通过JNI与其他应用（C,C++）的集成。
- 脚本+外部应用
 - 脚本和外部应用可同时使用，并可实现集成。

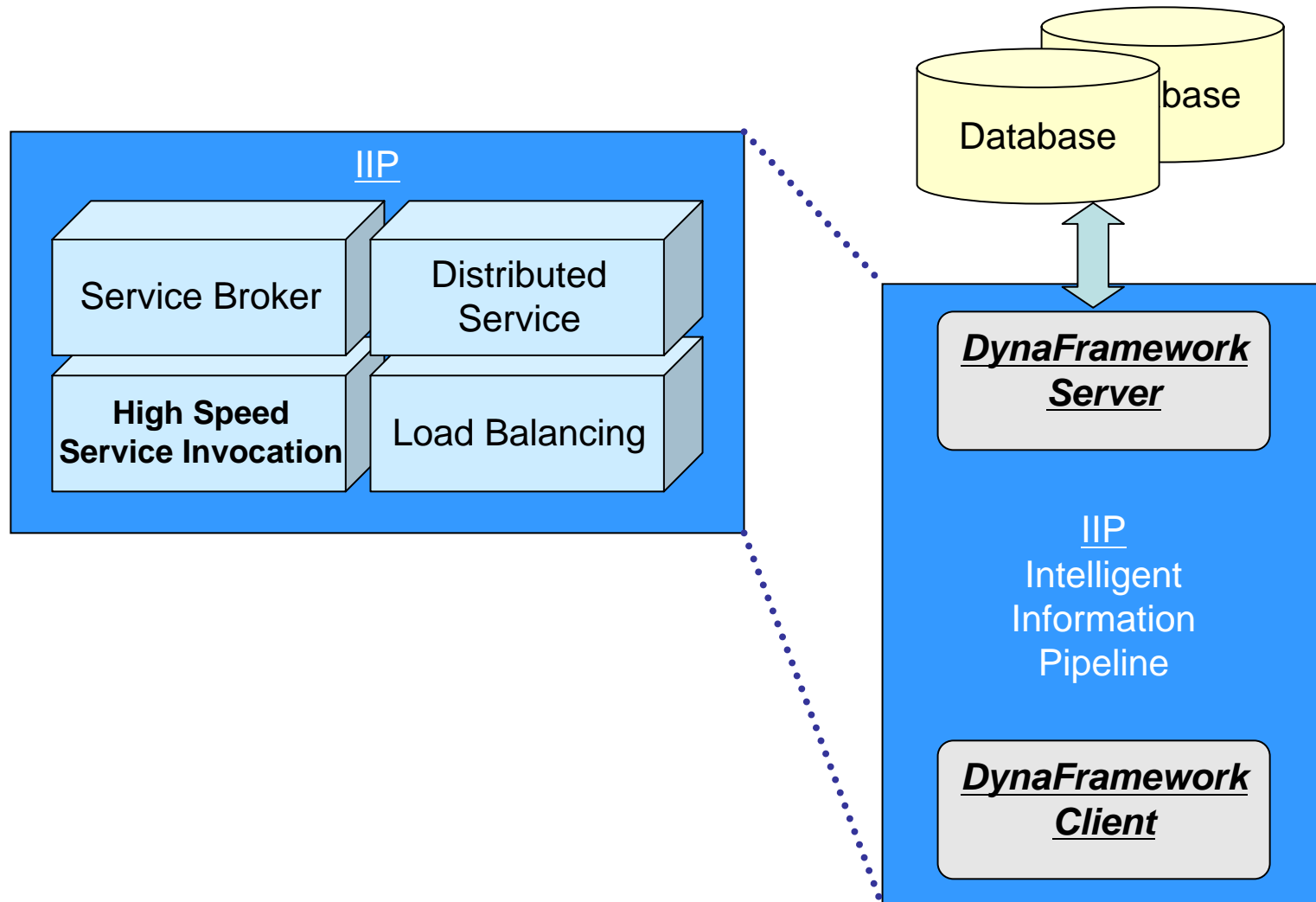
DynaMOAD Framework (服务端)



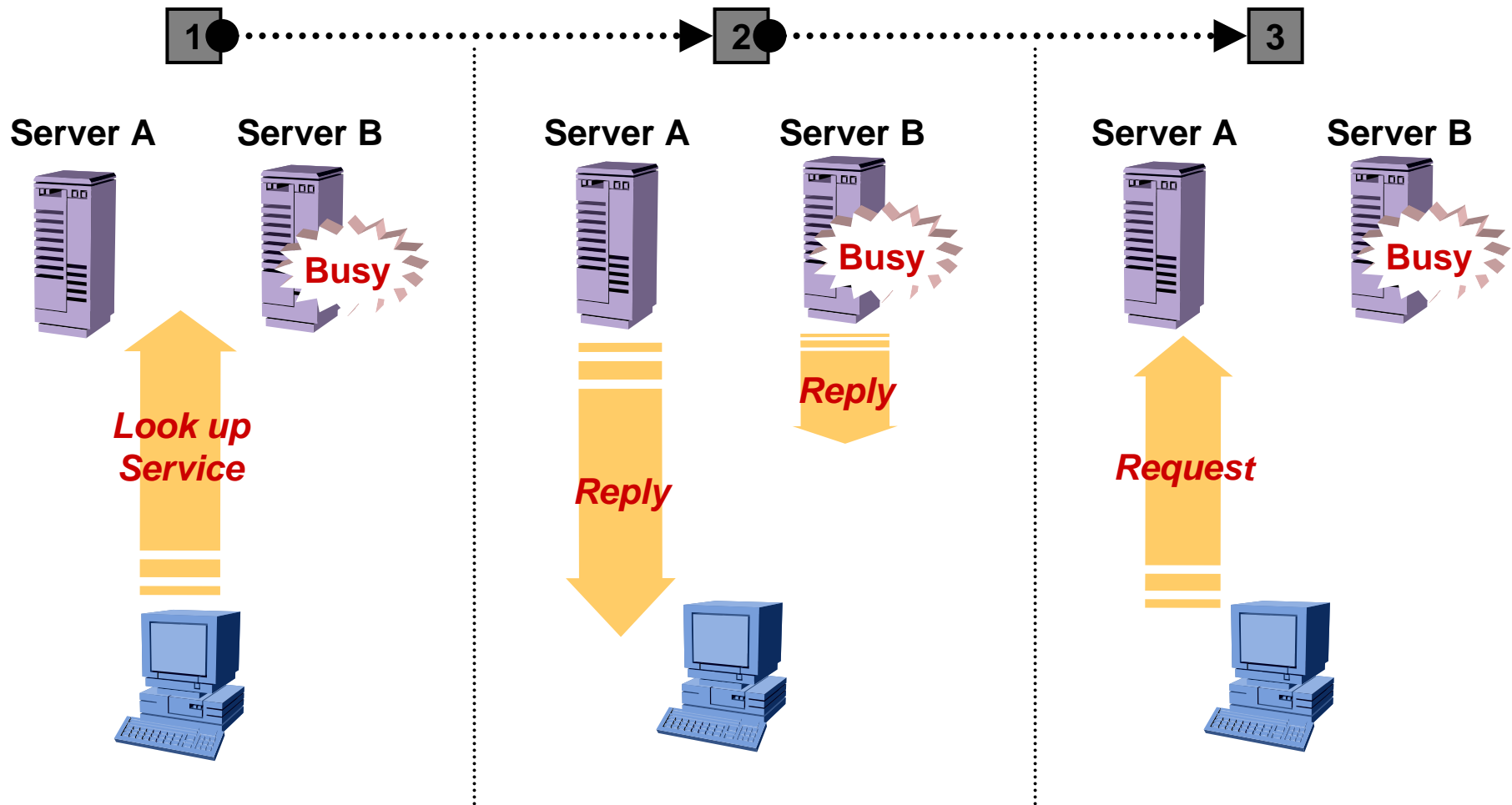
DynaMOAD Framework (客户端)



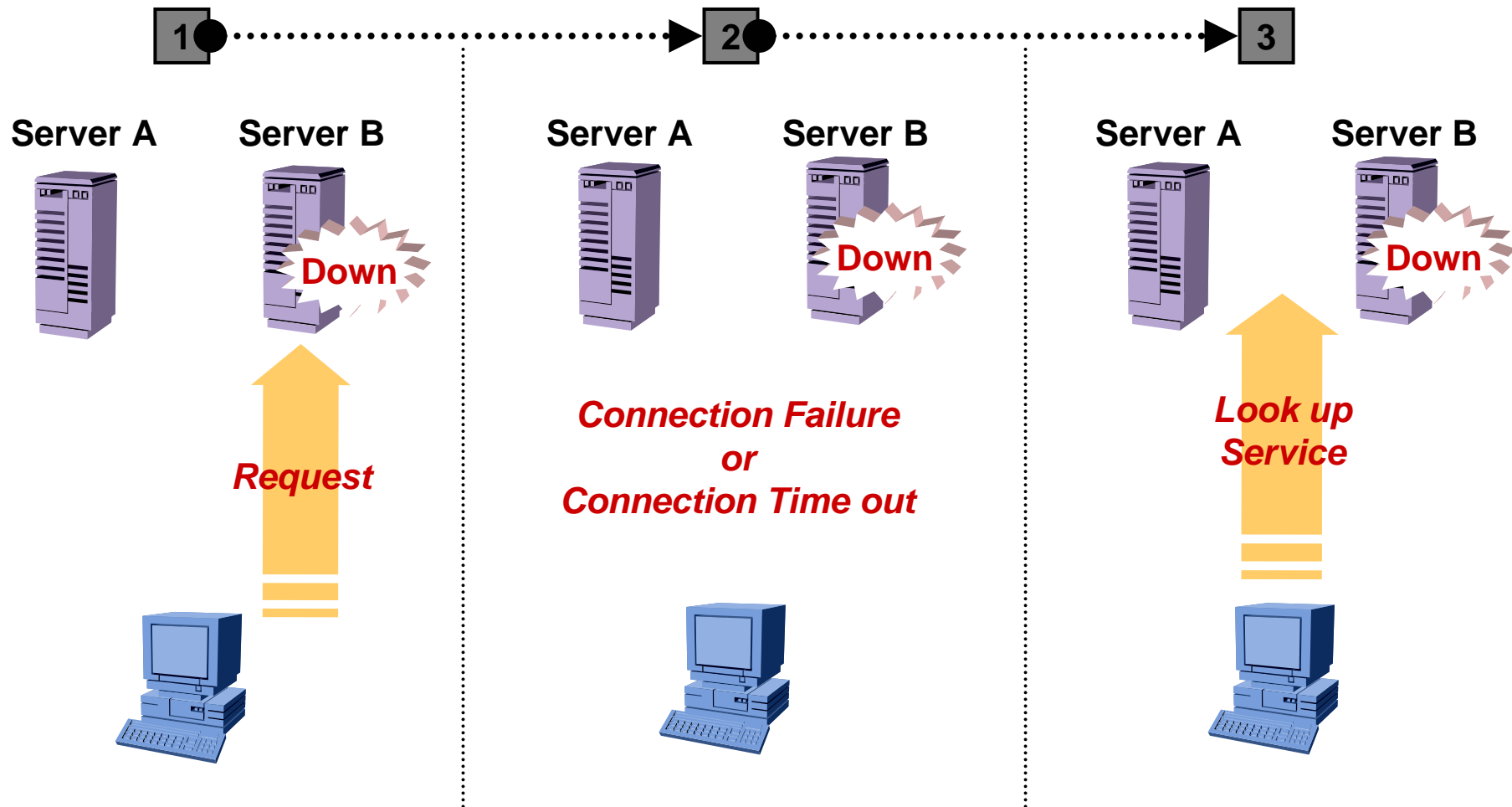
DynaMOAD Framework C/S (IIP)



Load Balancing Feature of IIP



Load Balancing Feature of IIP(HA Support)



IIP V.S. RMI (or other RPC Protocol)

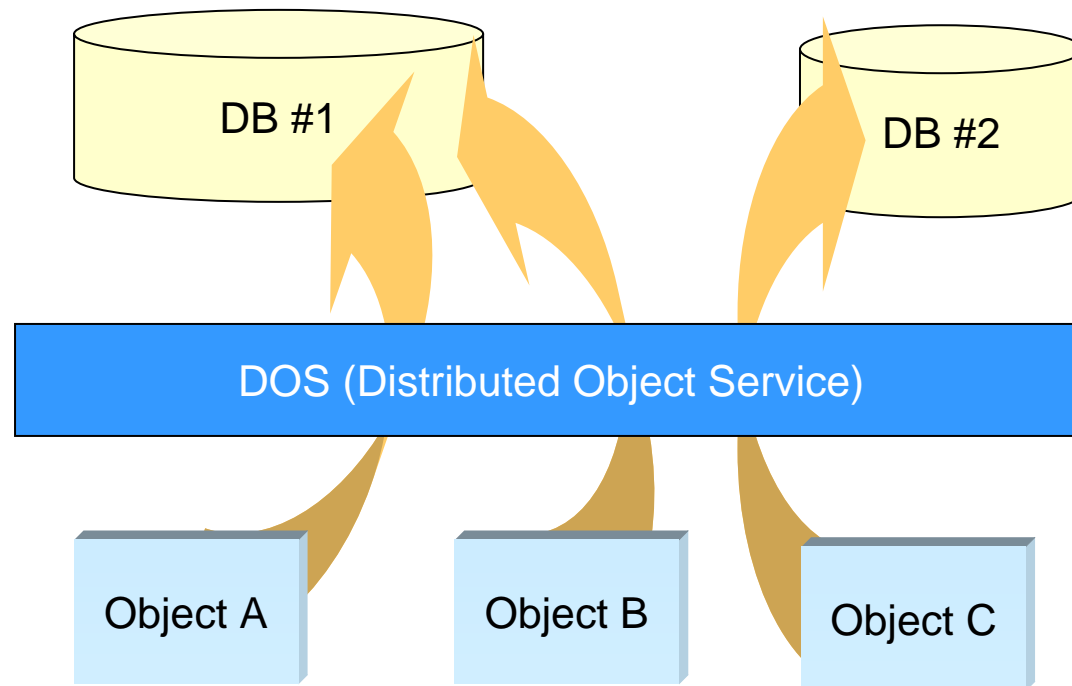
- IIP vs. RMI
 - 在同样的参数和返回值的情况下, IIP的调用速度超过RMI192%.
 - 越快的IIP就越能传输复杂的对象。比如参数和返回值
 - 例如, 过于重复的 Linked-List 或者 Hash Map Object
- Stub/Skel 代码生成
 - 无需再编译, IIP在内部动态生成Stub/Skel代码
 - 而RMI不同, 只要方法变化, 就需重新编译
- COM和EJB
 - IIP根据高性能和易用性, COM, EJB使用了类似RPC方式.

基于UML的对象引擎

- 基于UML
 - 支持基于UML的对象模型和大部分与UML类似的对象模型
 - 支持对象和关系的多重继承
- 用户界面的继承
 - 使用对象引擎管理用户界面的定义
 - 用户界面的定义同样可被继承，所以无需通过修改源代码来实现用户界面的修改.
- 管理包
 - 类的定义按包的形式管理
 - 共享已有的包可以极大程度复用已有的资源

基于UML的对象引擎

- 分布式对象
 - 支持分布式数据库
 - 每个类都可以存放在不同的数据库中
 - 数据根据定义存放在指定的数据库中

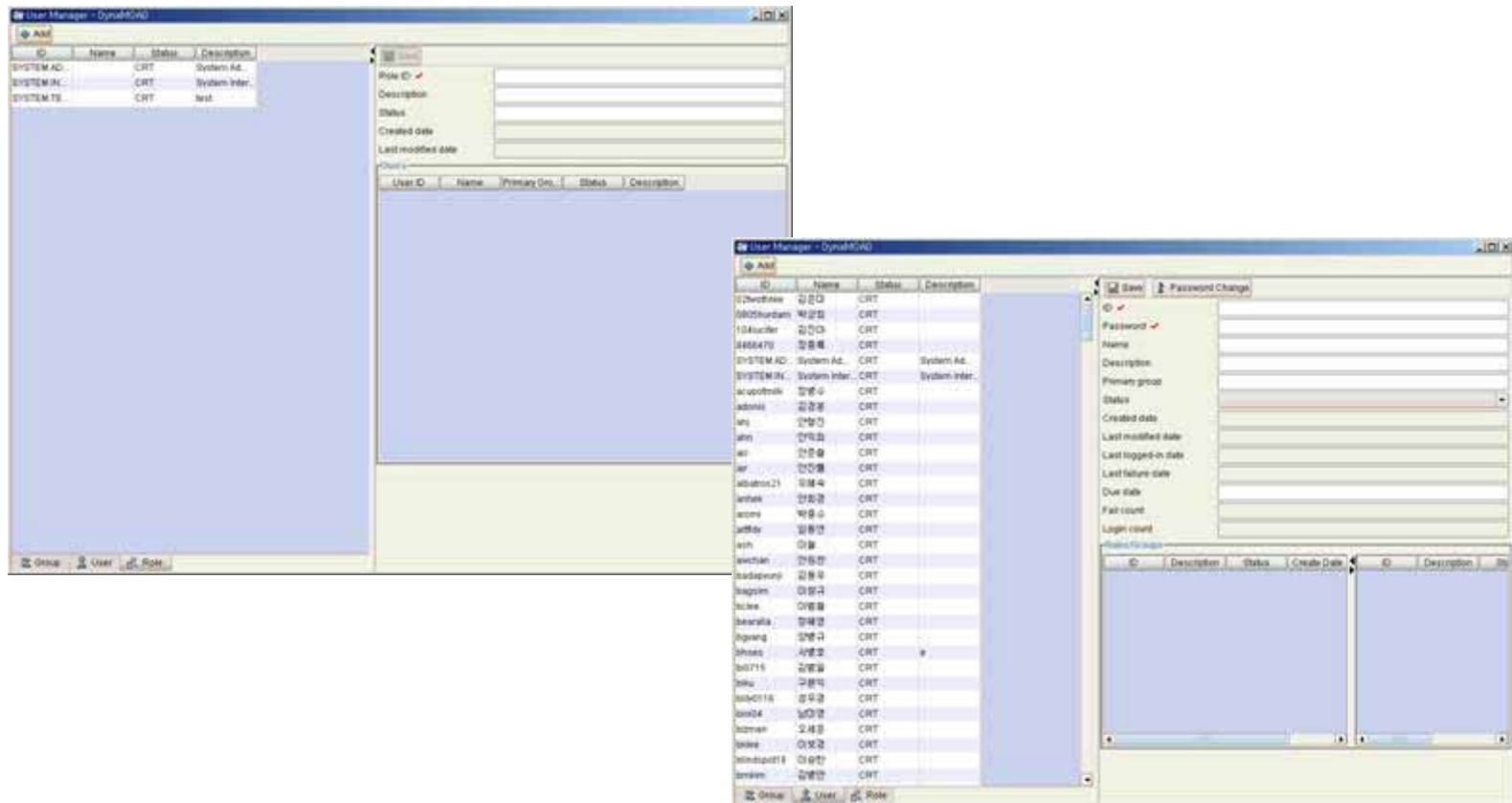


易于维护

- 简化对象模型的修订
 - 无需更改源代码，DynaMOAD可实现对象模型与对象数据库的同步，并保持一致。
- 简化UI的变更
 - 通过重新指定继承的关系便可实现UI的结构变更。

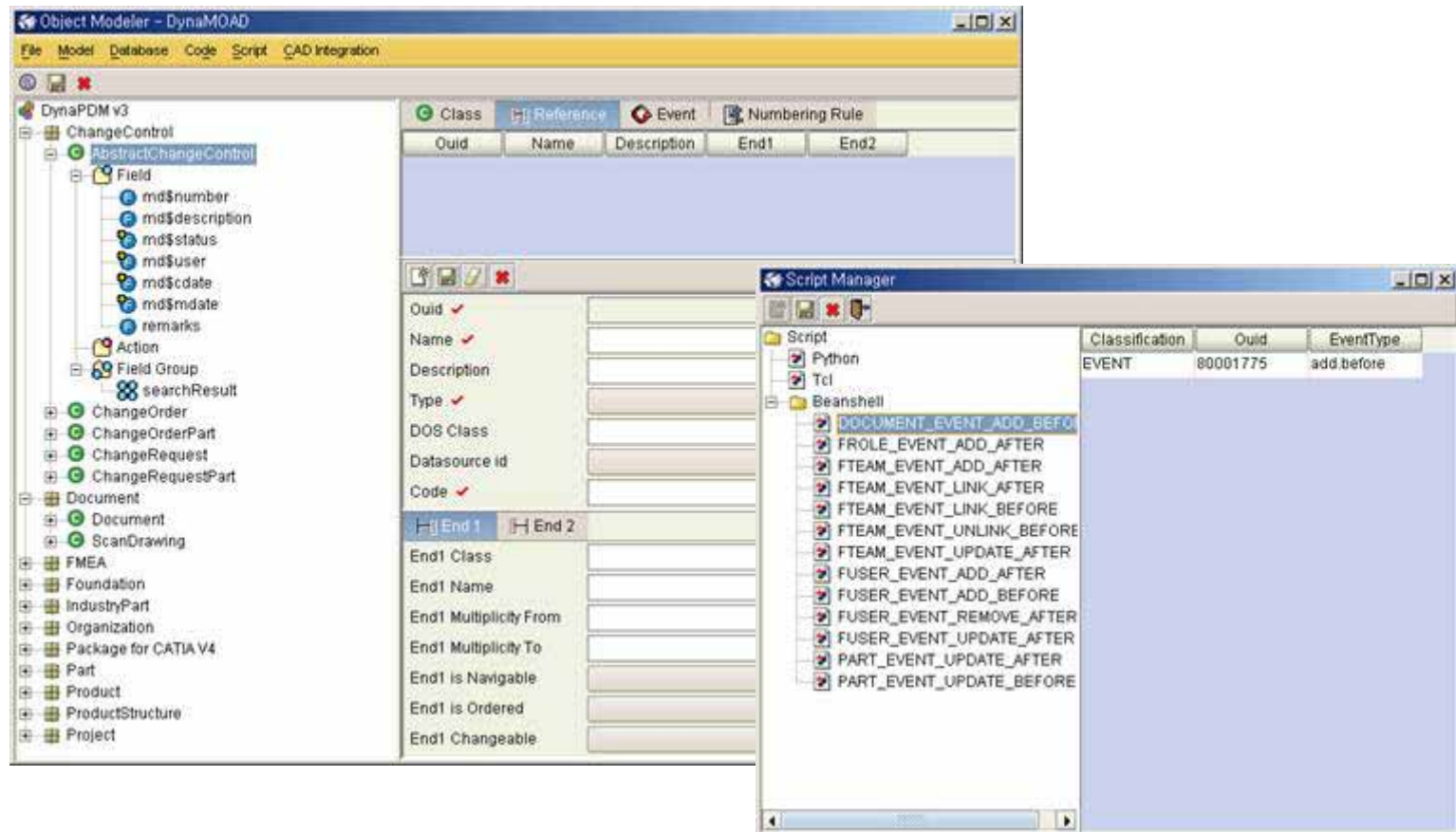
用户管理

- 用户、组、角色的与权限的管理



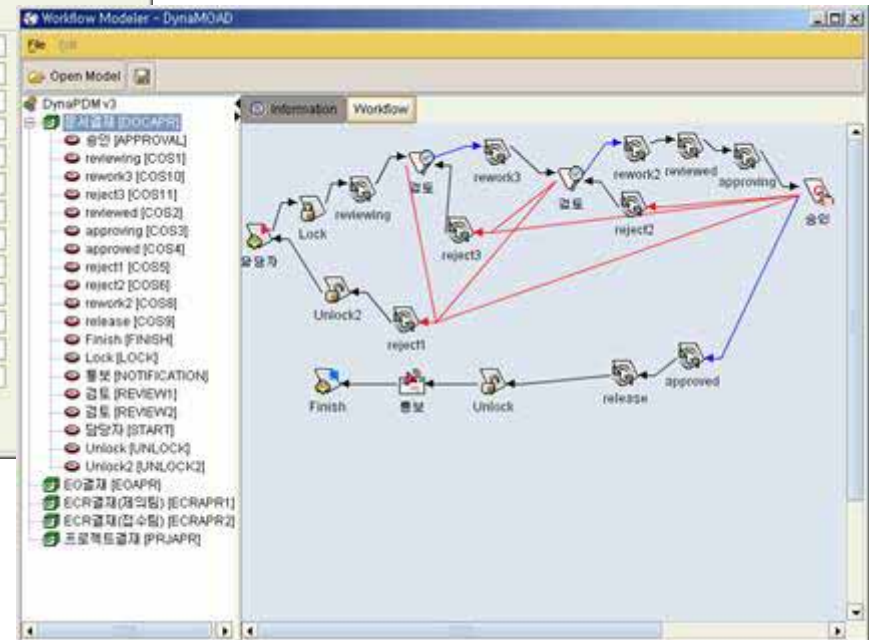
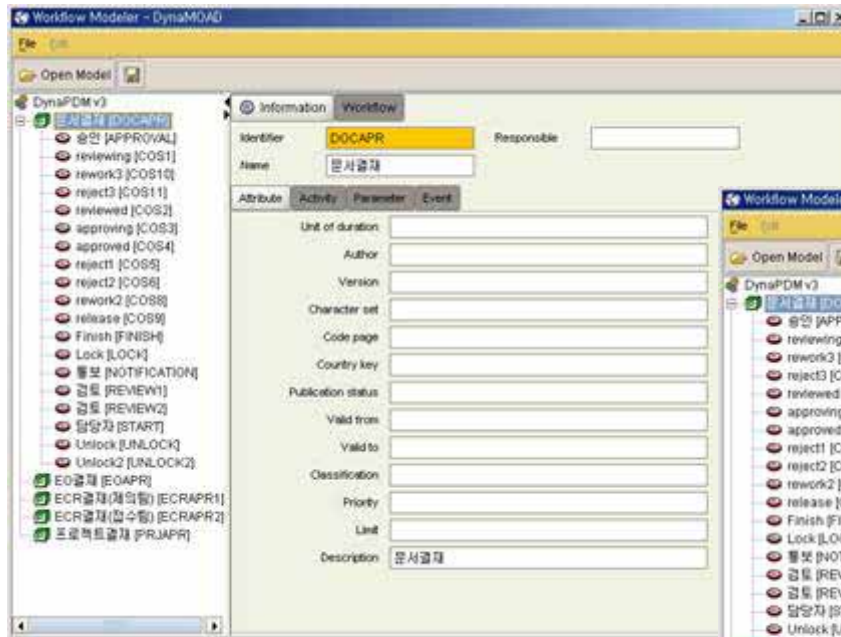
对象建模

- 强大的对象建模工具可定义多种复杂的类

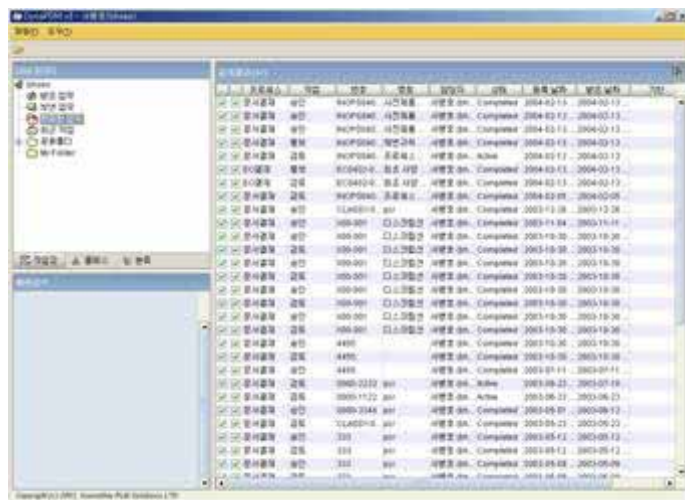


workflow建模

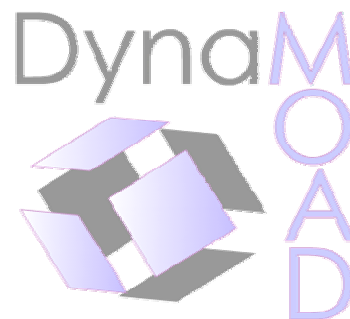
- 遵循WfMC标准的强大工作流引擎



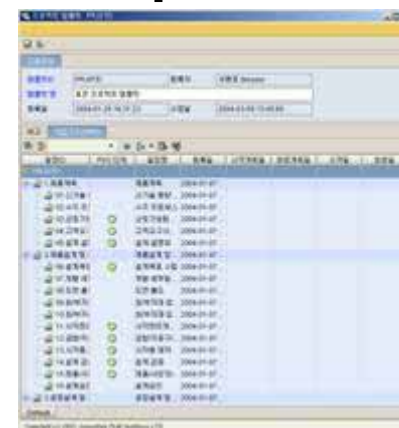
由DynaMOAD构建的应用



[主窗口]



[workflow窗口]



[对象信息的不同形式的窗口]