IBM Rational 软件创新论坛

IBM

Let's build a
smarter planet.

开发有道
创新"智"造

Innovate2010

模型驱动与软件资产

议程

- 大型软件企业开发现状
- 模型驱动开发的几个重要概念
- 模型驱动开发的原理和实现
- 模型驱动开发解决方案
- 模型驱动开发核心价值
- 模型驱动开发项目实施后续工作

大型软件企业开发现状

- 某大型企业主机应用开发的过程

参考数据结构，通过文档进行查询

代码拷贝修改，重复劳动量大

单元测试不测，直接进行联调测试

开发完补写文档，内容不准确

大型软件企业开发现状

管理部门关心的问题：

- IT系统作为业务模式的支撑越来越明显

如何规范化开发，提高产品质量，降低缺陷率？

- 产品版本上线的周期非常紧张

如何在保证质量的前提下提高软件开发效率，缩短上线周期？

大型软件企业开发现状

分析员，架构师关心的问题：

- 业务问题越来越复杂，对应用系统要求越来越高

如何有效地进行业务分析，需求分析，提高分析质量。

- 应用的架构非常复杂，涉及的技术广泛

如何有效掌控系统架构，规范开发，降低技术风险，提高系统的可维护性？

大型软件企业开发现状

开发人员关心的问题：

- 新技术不断出现，不断学习，应用这些技术到工作中

如何降低新技术的出现，对工作的影响，提高新技术学习的效率。

- 大量的重复编码工作量

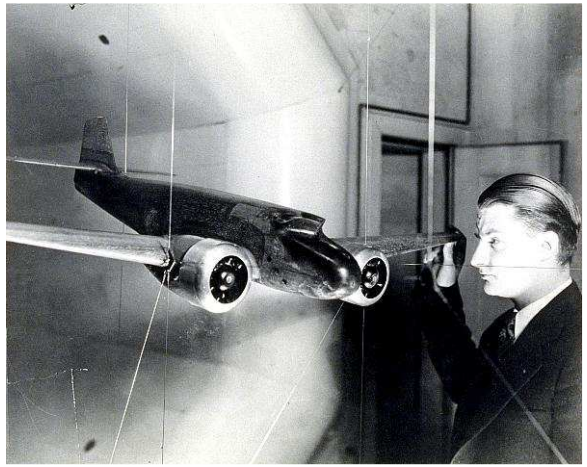
如何减少代码开发的重复工作，充分利用开发人员的智慧

大型软件企业开发现状

- 缺乏可跟踪可操作的开发管理流程
- 大量同类型的模块缺乏共享机制
- 机械重复性劳动缺乏自动化机制
- 文档资产不准确，不完善
- 无软件分析、设计资产可供重用

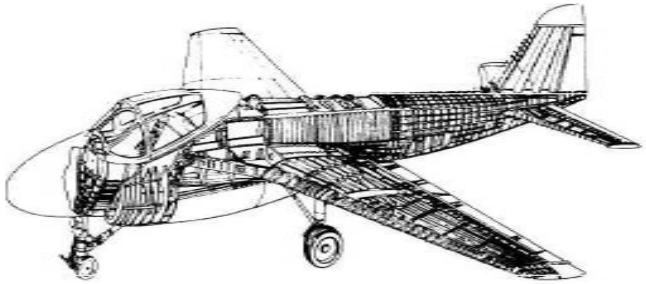
几个重要概念-模型和建模

模型：
从某一个特定的角度彻底描述系统

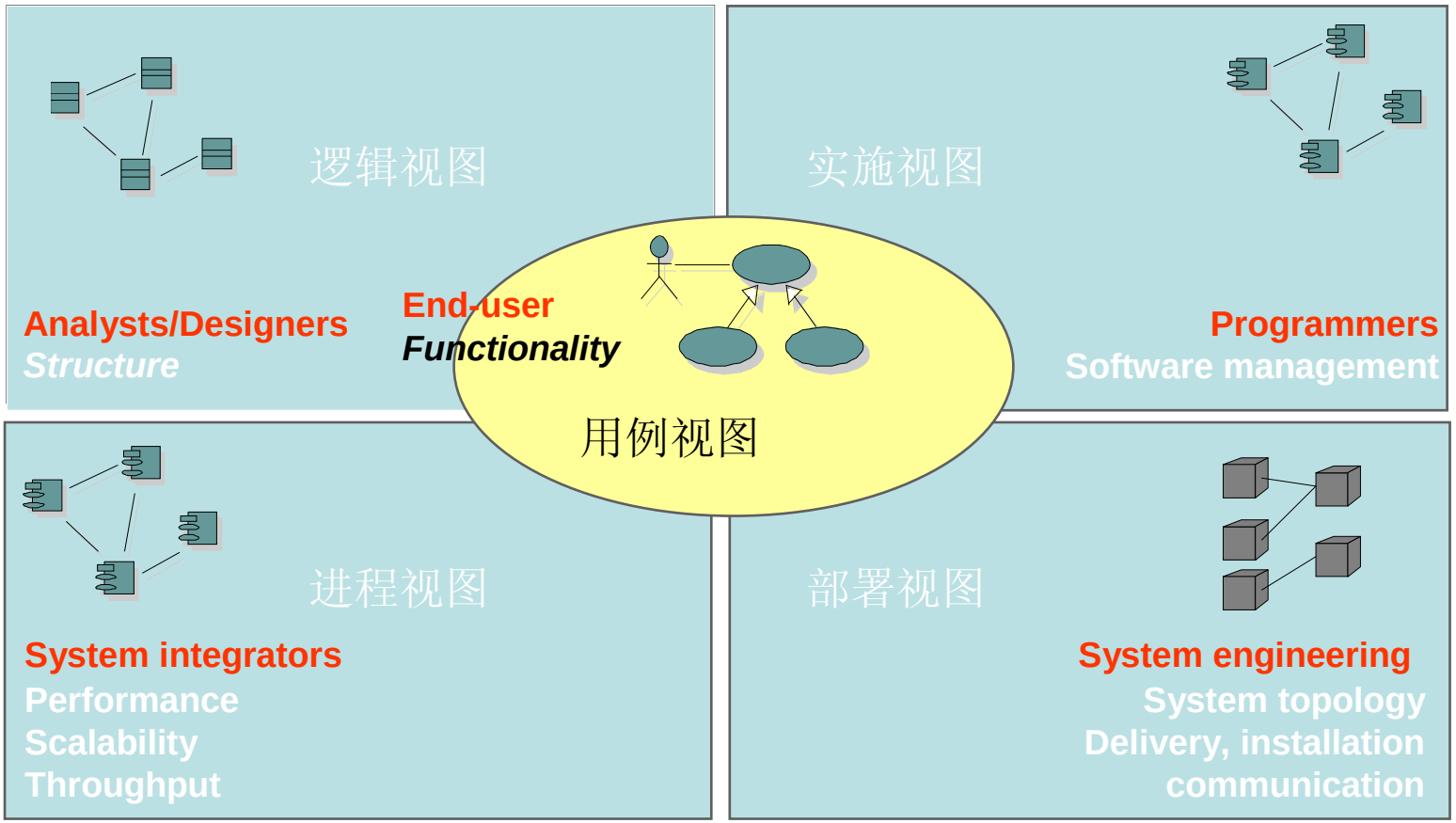


为什么要建模：

- 管理复杂度
- 在早期发现事物的错误和遗漏
- 采用不同的选项进行验证
- 采用模型更便于涉众进行沟通
- 模型可以驱动整个实施的过程

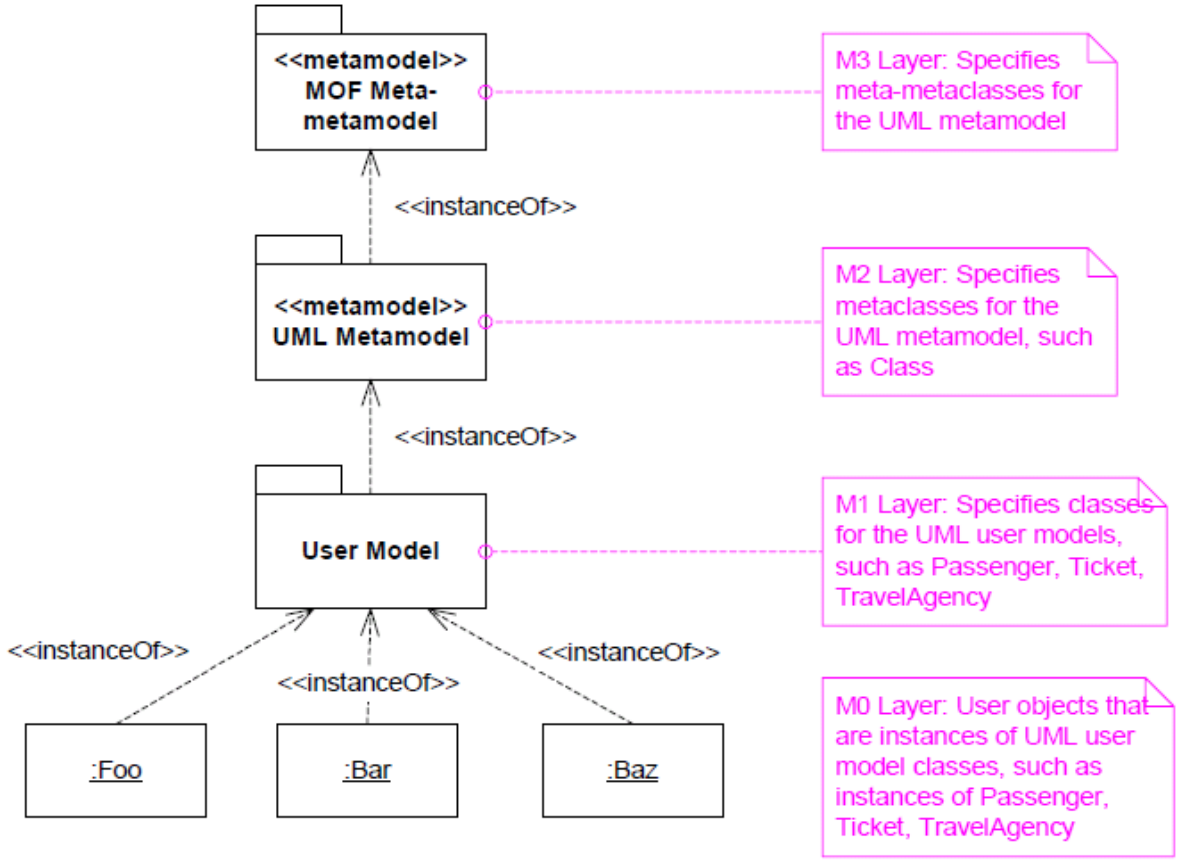


几个重要概念-软件建模

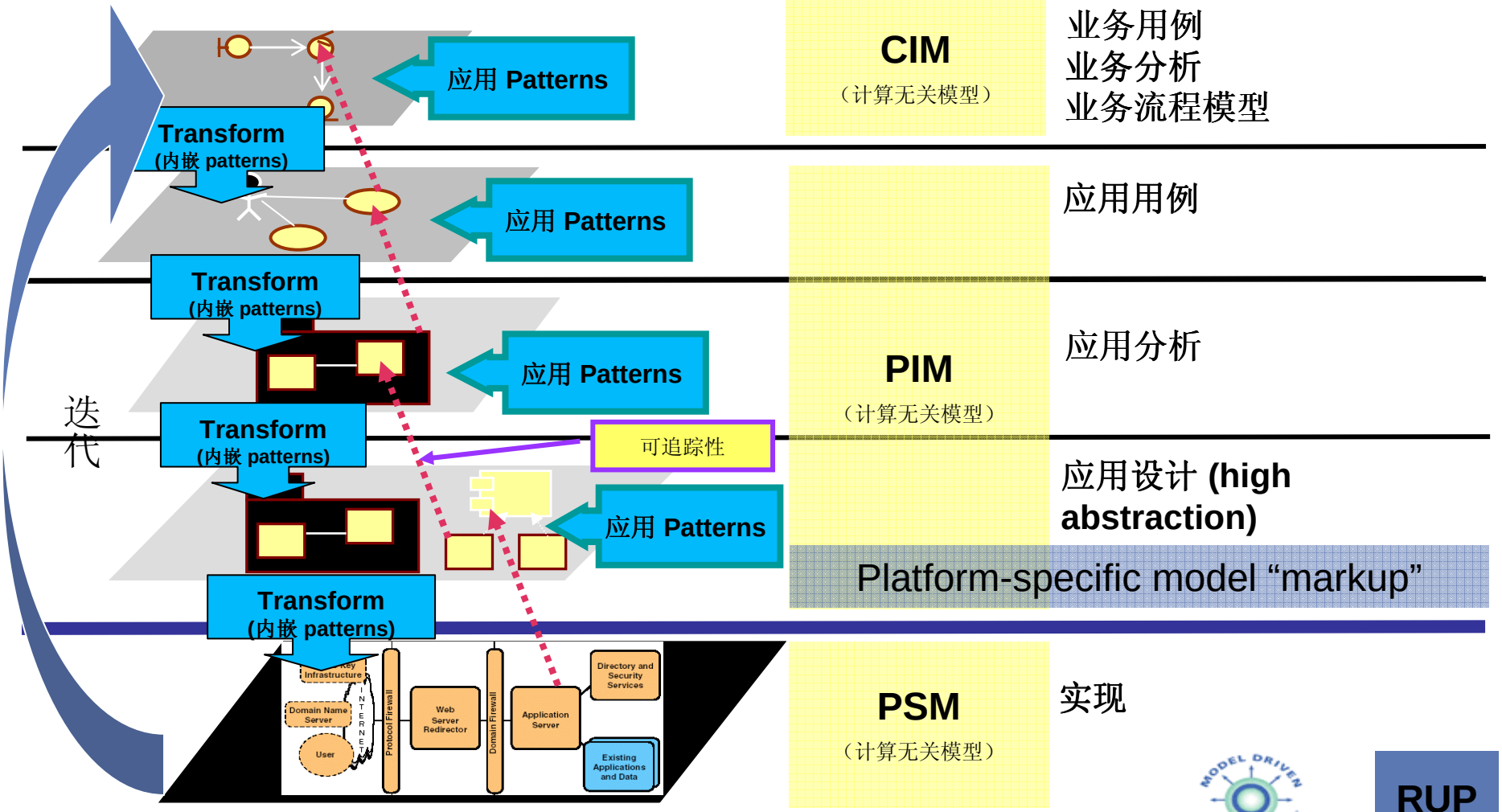

逻辑视图
实施视图
Analysts/Designers
Structure
End-user
Functionality
Programmers
Software management
用例视图
进程视图
部署视图
System integrators
Performance
Scalability
Throughput
System engineering
System topology
Delivery, installation
communication

几个重要概念-模型驱动开发

- OBJECT MODEL　　MOF (Model Object Facility)



几个重要概念-模型驱动开发（Pattern和Transformation）



几个重要概念-模型驱动开发

- 关注业务：
  用针对业务的模型语言来描述业务 (Domain Specific Language)
- 抽象层次：
  用模型来表达软件实现业务的过程,
  业务到技术，从需求到软件 PIM ->PSM
- 架构的重要性：
  在体系结构中，强调对实现细节的更加有效的组织，屏蔽细节
- Transformation:
  通过工具实现代码和文档的自动生成

模型驱动开发的原理和实现

- 专家经验的获取和重用

固化在软件资产中，固化在Transformation的规则中。

- 领域专家的价值

根据领域知识，结合DSL，规划整个目标领域中的模型驱动过程描述规范。

- 技术专家的价值

固化软件平台架构，规范软件开发架构，依据DSL的描述规范化开发过程

开发Transformation 工具集。

模型驱动开发的原理和实现

模型驱动开发成功的四个要素：

概念：

可视化建模，面向对象，component，Service，model transformation等

标准：

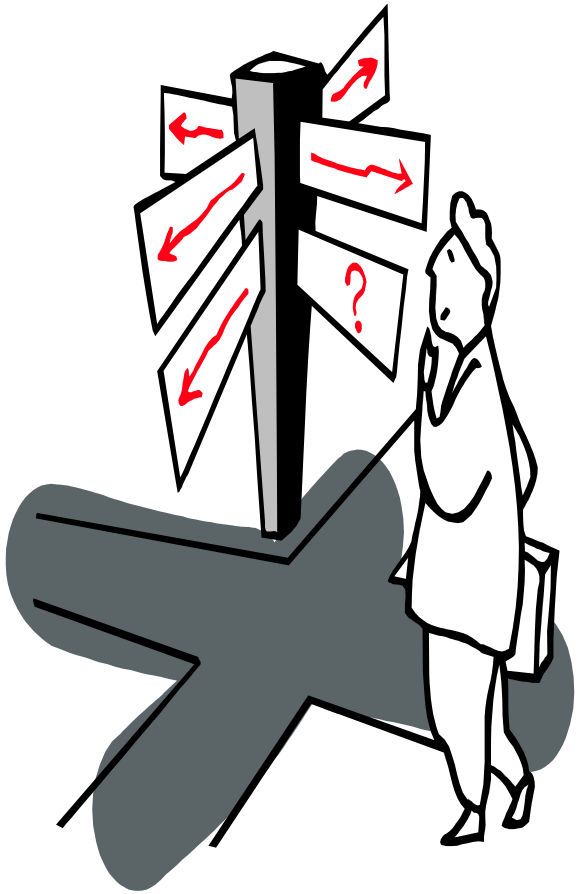
代码标准，产品标准，规程标准，产品架构标准，eclipse ，
MDA/MDD, UML,J2EE，WebService，Eclipse Plug-in，JET

流程：

开发相关的流程，质量管理流程，资产管理流程,RUP

自动化：

UML tools ，MDA tools，RAS tools，Eclipse tools
代码生成技术和SDK工具，



模型驱动开发的原理和实现

实施的总体原则

- *建模过程标准化*

元素的标准表达方式，元素位置的固定存放方式，建模过程的模式化

- *工作内容模板化*

创建模型项目模板，创建用例组织包结构模板，创建用例模板

- *代码生成自动化*

基于UML模型生成通讯区代码，DSR等, 基于流程图生成EGL 交易代码代码
基于UML模型生成文档等

- ***IT治理集约化***

基于模型进行IT治理

- *最佳实践资产化*

软件开发过程中的最佳实践提炼成中心的软件资产，得到广泛重用

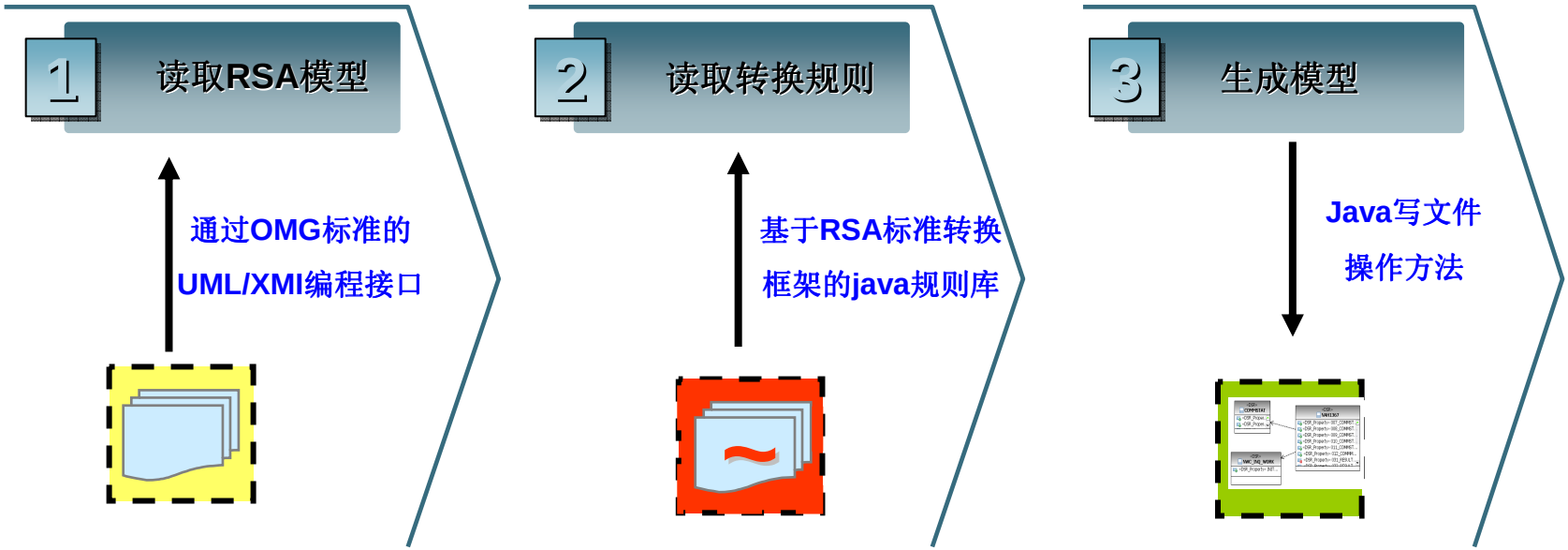
模型驱动开发的相关术语

- 构造型(stereotype)：在基于已经存在的模型元素的类型的基础上定义的新类型的模型元素。构造型可以扩展已存在元模型类的语义。
- 概要文件(profile)：一组构造型组成的文件。
- 转换(transform)：基于自定义规则的模型到模型，模型到代码的转换，实现自动化的软件开发。
- 模式(pattern):针对特定问题的固定的好的解决办法.能存在于从代码到设计到业务模式等任意抽象层次.

stereotype　　profile

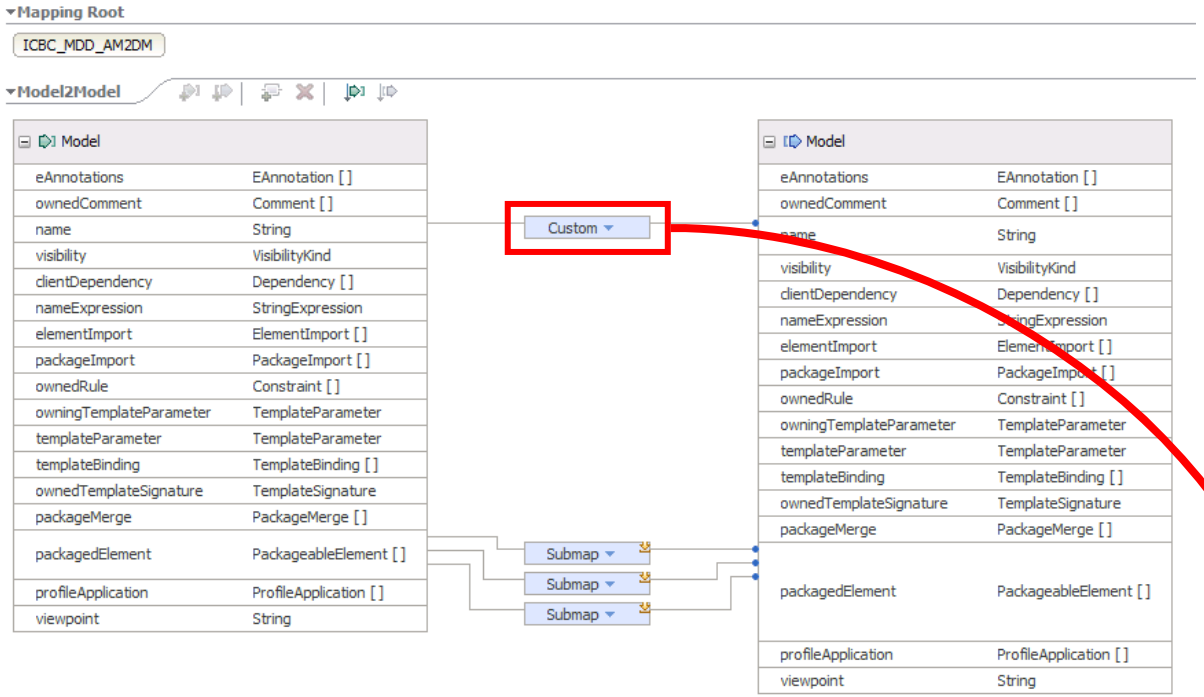
自动化模型到模型转换原理

移到MDD

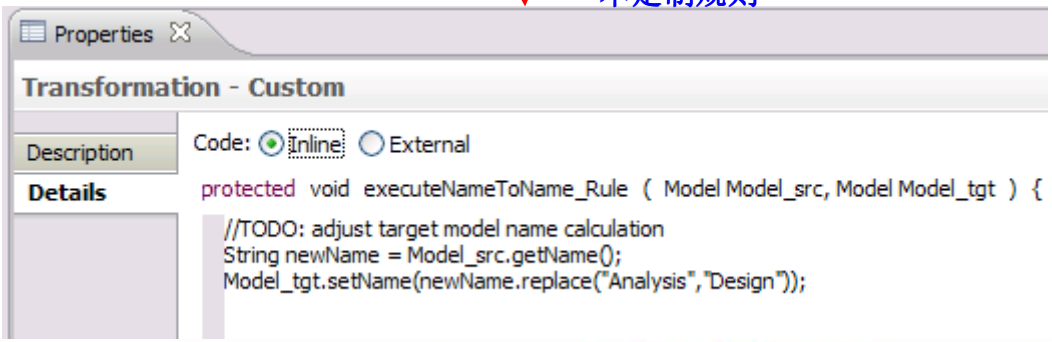


- 全部基于开放的技术标准

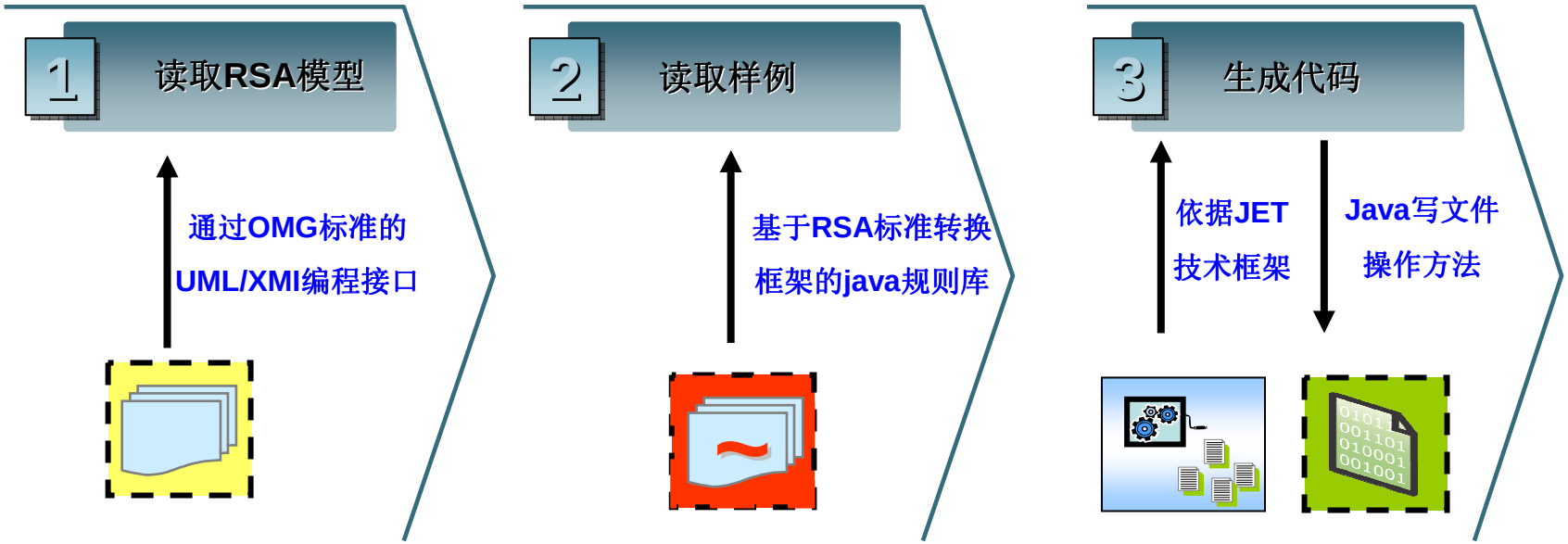
RSA强大的图形化模型到模型转换规则定义器



支持通过Java脚本定制规则



自动化代码转换原理

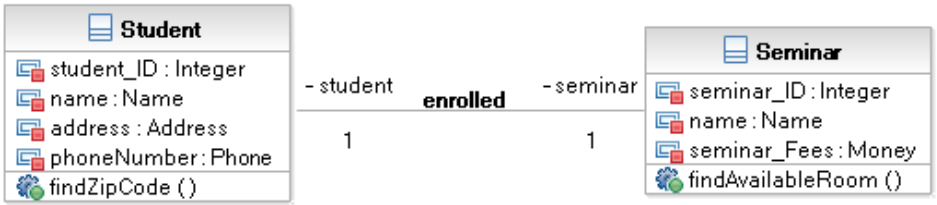


- 全部基于开放的技术标准

模型到代码转换原理


```
<root>
    <coverLetter>
        <company name = "FedEx" position="delivery speacialist"/>
        <skillset>
            <skill name="driving"/>
            <skill name="punctuality"/>
            <skill name="looking good in brown"/>
        </skillset>
    </coverLetter>
</root>
```

典型的pattern转换



Read

```
Function GetStudent(searchRecord Student inout,
status StatusRec)
  try
     get searchRecord;
     if (SysVar.sqlData.sqlCode == 100)
        HandleDBRecordNotFound(status, "Student");
     else
        HandleSuccess(status);
     end
        SysLib.commit();
  onException (exception SQLException)
        HandleException(status, exception);
  end
end
```

```
Function AddStudent(newRecord Student, status
StatusRec)
 if ( IsValid(newRecord) )
   try
        add newRecord;
        HandleSuccess(status);
   onException (exception SQLException)
HandleException(status, exception);
   end
 else
   HandleInvalidDBRecord(status);
 end
end
```

Create

```
Function UpdateStudent(updateRecord Student,
status StatusRec)
  try
     if ( IsValid(updateRecord) )
       replace updateRecord noCursor;
       HandleSuccess(status);
     end
  onException (exception SQLException)
       HandleException(status, exception);
  end
end
```

Update

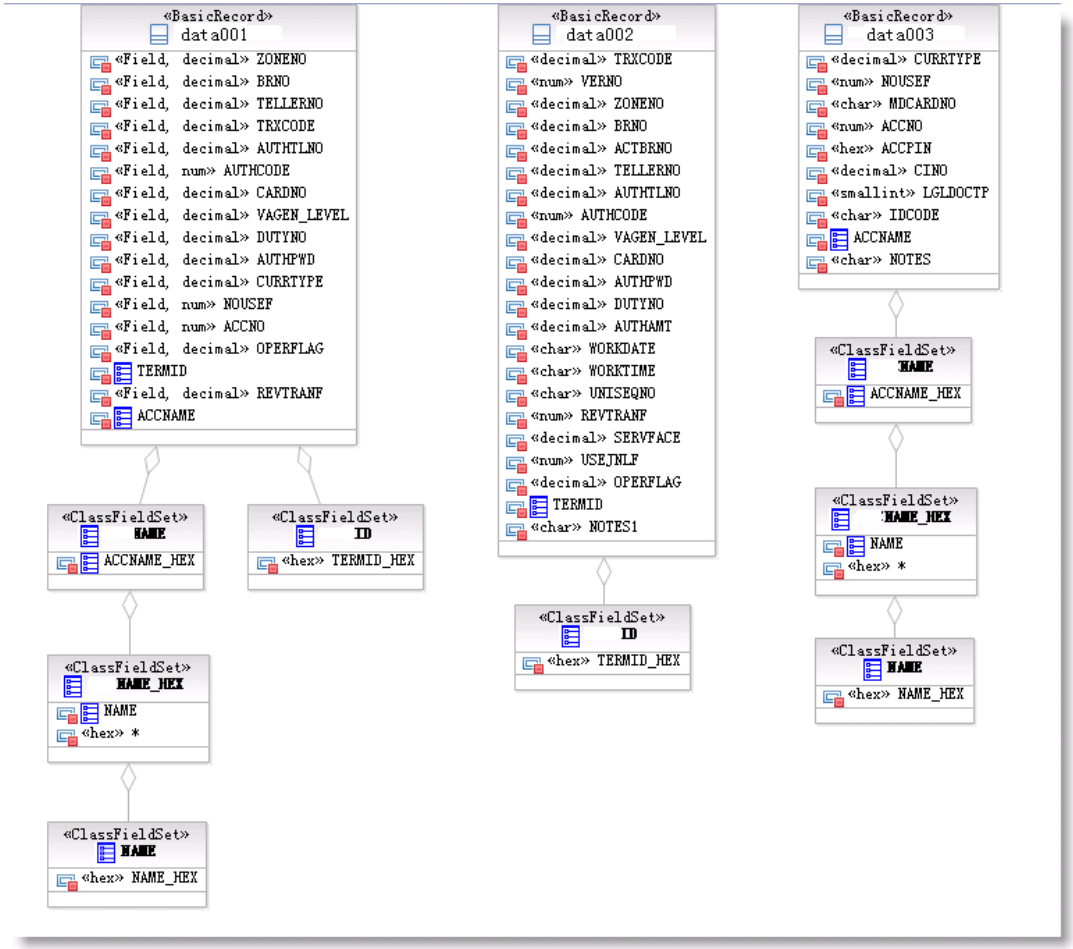
```
Function DeleteStudent(deletionRecord Student,
status StatusRec)
  try
    delete deletionRecord noCursor;
    HandleSuccess(status);
  onException (exception SQLException)
    HandleException(status, exception);
  end
end
```

Delete

EGL 生成的数据访问代码

模型驱动开发解决方案-整体开发流程

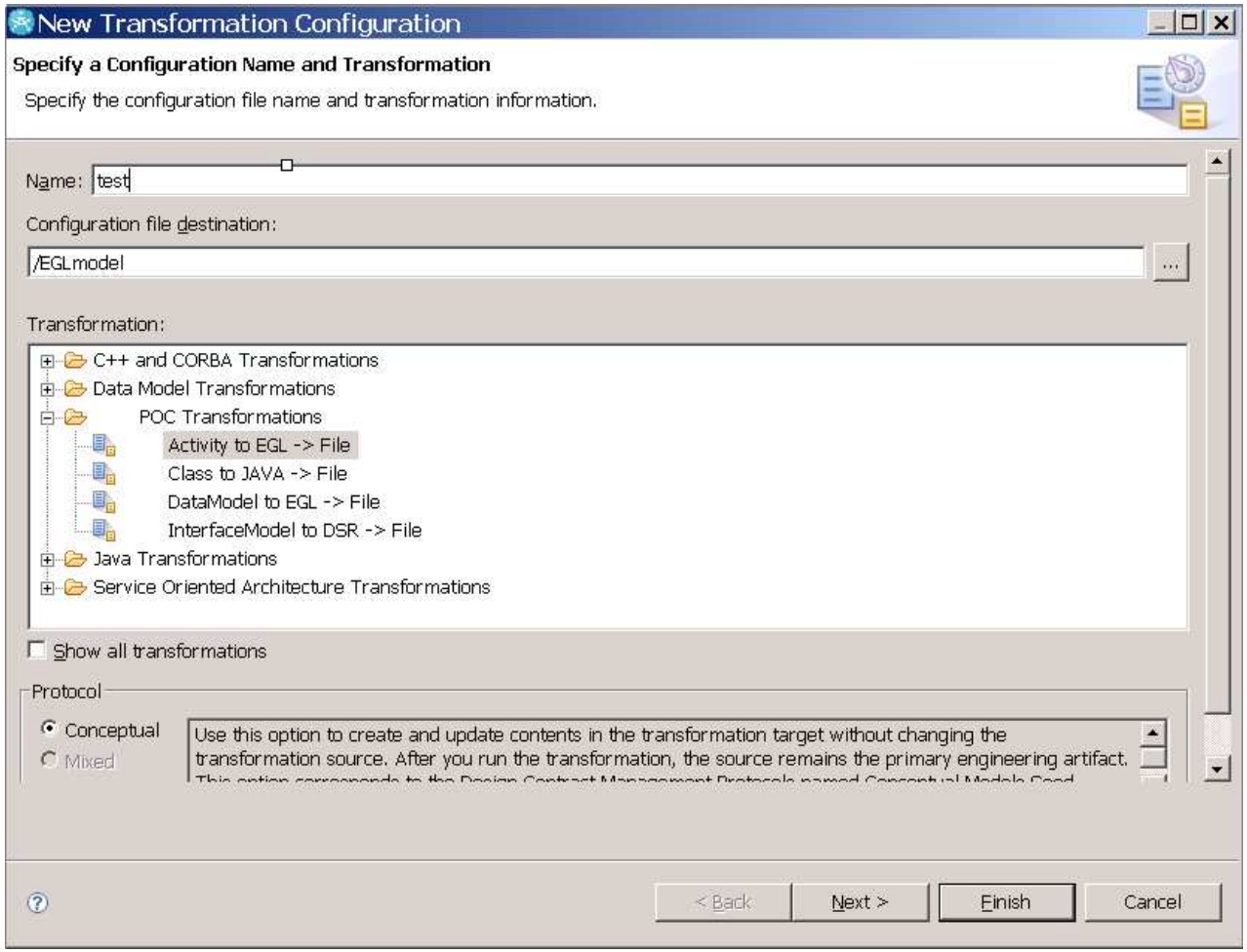
模型的表示方式



提供给各应用程序使用

生成交易代码，接口代码等的依据

模型到代码转换的实现方法



1. 逻辑处理流程图- > EGL主程序代码
2. 数据验证模型-> 数据校验交易代码
3. 数据模型-> 对数据原子操作的EGL代码
4. 通讯区模型-> DSR数据文件
5. UI模型 -> xform 界面文件

模型生成文档的实例

- 基于模型的文档生成自动化
- 基于UML模型生成业务需求文档（对应RUP的软件需求文档），软件需求文档 （对应RUP的设计文档）

第二章　软件需求描述

2.1 应用列表

应用名称	功能描述是否全显示	业务规则评估方法
[ownedApplication]	否	功能点法
Footer Row		

2.2 功能列表

需求名称	需求内容描述	功能ID	功能名称	涉及角色	需求属性	附属说明	所属应用
row['requirementName']	[requirementContent]	[functionId]	[functionName]	[actorName]	[requirementType]	[comment]	[ownedApplication]
Footer Row							

2.3 企业级客户信息应用

Header

Detail

"2.3."+(row.__rownum+1)+" "+row["functionName"]

(1) 输入要求描述

a. 输入画面设计

假无

b. 输入字段定义表

字段名	种类	数据类型	是否必输	长度	范围(基度)	输入限制	说明
[chineseName]	输入	[dataTypeName]	[mustFill]	[length]	----	row['constraint']	[comments]

c. 输入字段间的约束关系

假无

Header

(2) 业务功能描述

Detail

功能名	[name]	功能标识	row._outer['functionId']
功能描述	row._outer['functionDescriptoin']		
使用用户	[actorName]		
先行条件	无		
业务处理流程：			

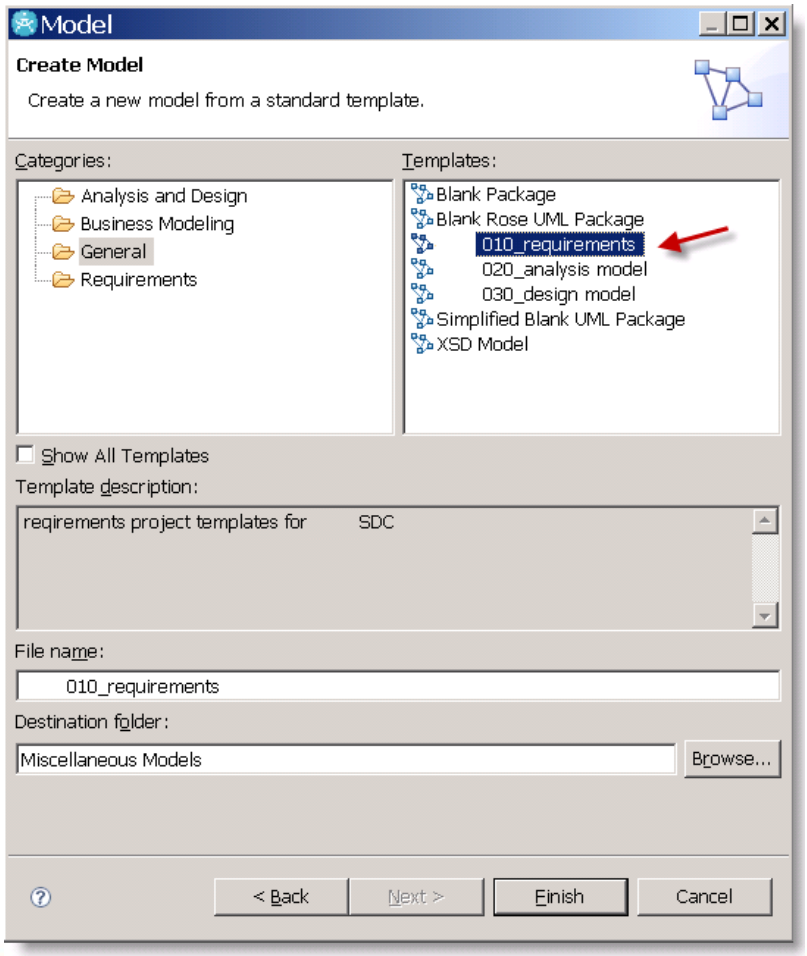
生成PDF文档

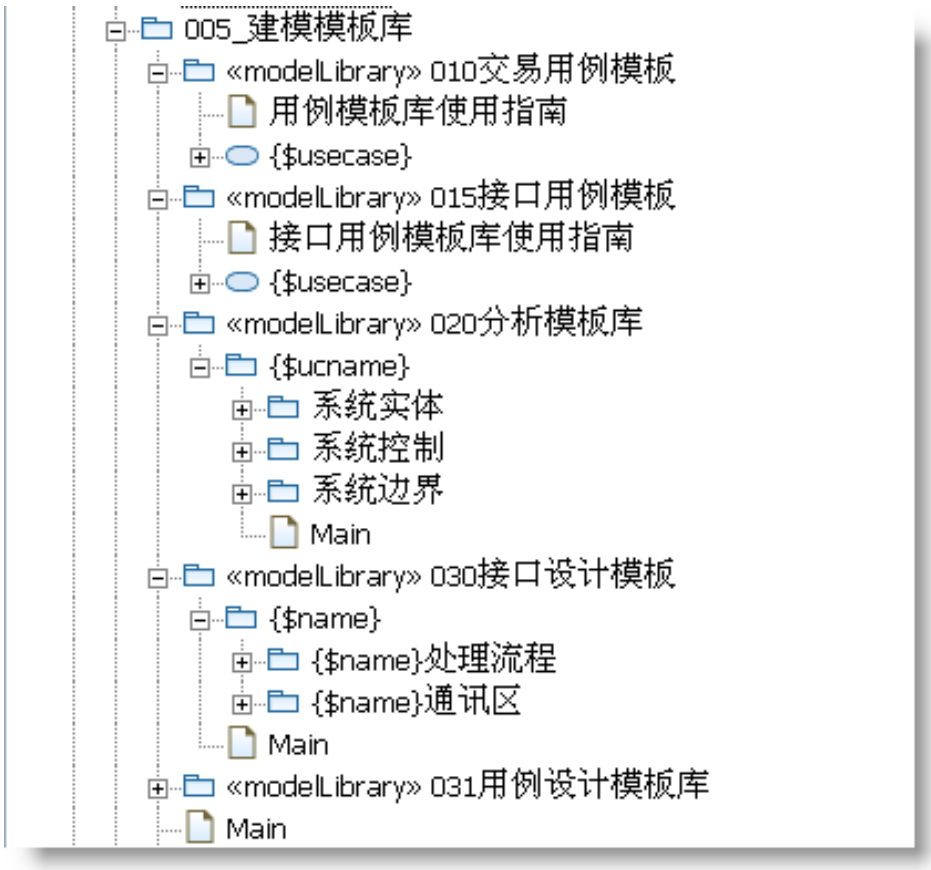
生成Word文档

模型驱动开发解决方案

- 工作内容模板化

创建模型项目模板，创建用例组织包结构模板，创建用例模板





模型驱动开发解决方案

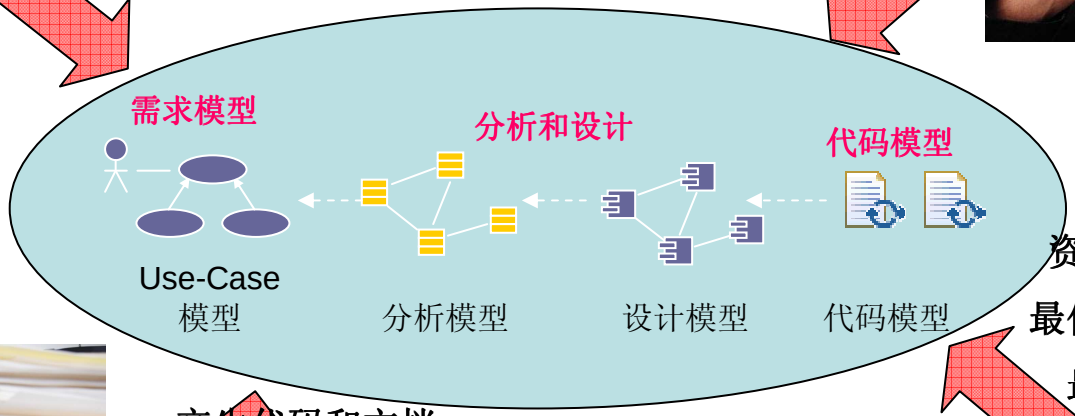
- 最佳实践资产化

可重用的代码模板，代码模式，静态代码片段
可重用的文档模板
可重用的数据模型资产，接口模型资产，公共函数模型资产，通讯区模型资产
可重用的大量的业务模式
可重用的Transformation规则

模型驱动开发实施-建立以模型为中心的软件生态圈

知识积累

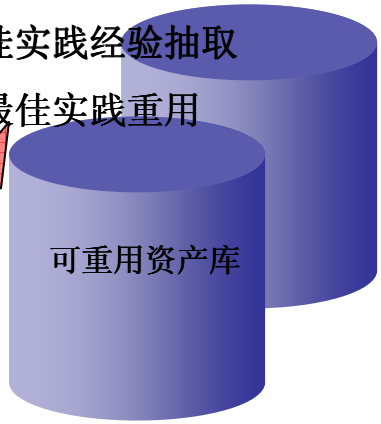
增强管理层
的控制力

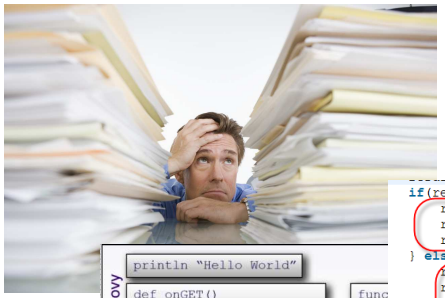


资产的双向流动

最佳实践经验抽取

最佳实践重用



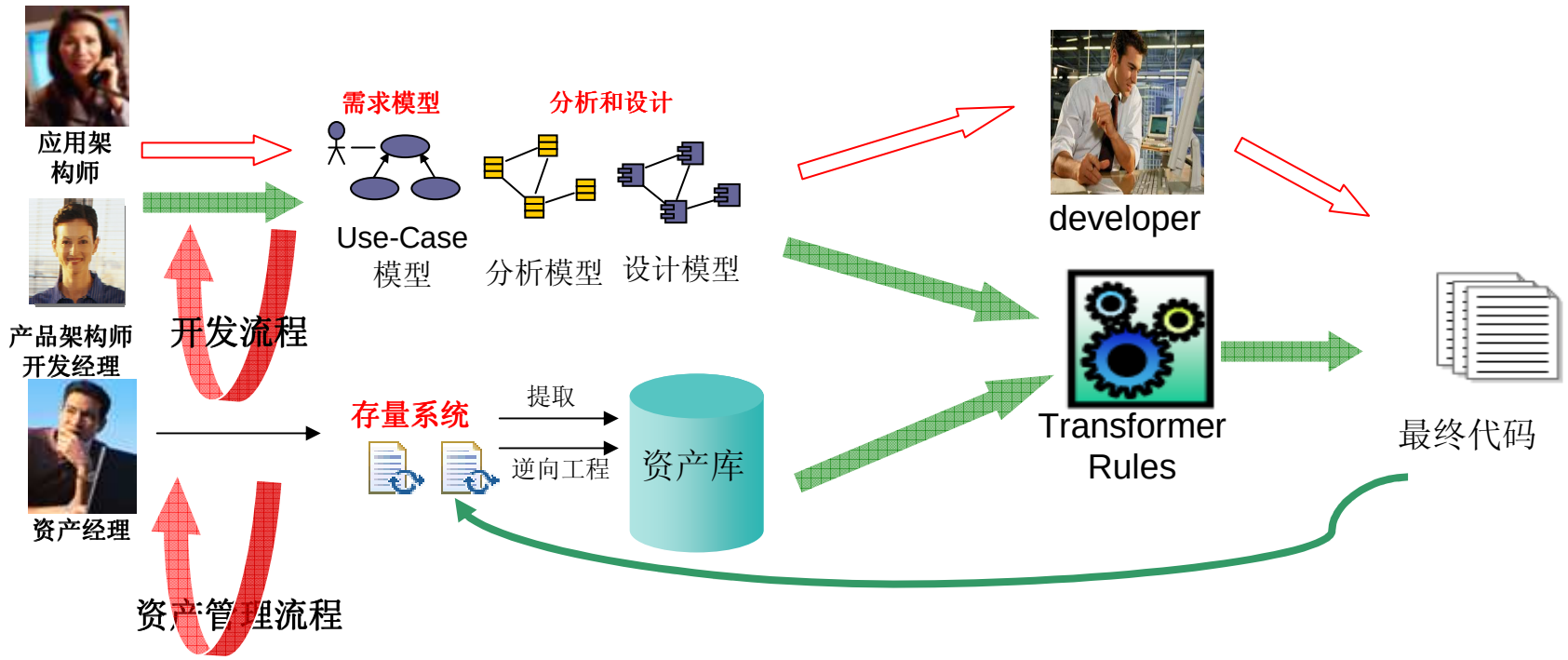


产生代码和文档
(Transform)

QA关心的内容

模型驱动开发项目的生命周期

- 基于资产的模型驱动开发过程的生命周期



模型驱动开发核心价值

- 可靠的软件资产被充分重用

软件开发的质量和效率提高软件开发更加规范化，代码风格统一，代码质量可控，稳定,开发效率大大提高。充分体现重用，企业中的代码资产将变得越来越重要了。

模型驱动开发核心价值

- 软件和文档的一致性

软件开发过程中对需求的关注度逐步提高，以模型为软件开发的核心，生成代码框架和过程文档，内容更加统一，过程更加有效，沟通更加有效。

模型驱动开发核心价值

- 软件的可维护性

增加了架构师对软件质量的把握和控制能力，软件架构起到了核心作用，软件的维护以架构为中心。

模型驱动开发核心价值

- 软件的资产的有效性

大量重复编码由高质量的代码模板替代，系统自动完成，软件开发效率，质量将有所提高并可控。软件的可变性，重构的能力大大增强。

模型驱动开发核心价值

- 软件设计过程的可追踪性

软件开发的过程中以模型为核心，更便于记录，便于沟通，提高了对分析设计过程的可追踪性。

模型驱动开发核心价值

- 提高开发人员工作积极性

程序员从大量重复的劳动中解脱出俩，更加关注软件资产的发掘，
他们的价值可以更大的发挥。

模型驱动开发核心价值

- 提高项目开发和产品交付速度
  这是从A点到B点的最快方式
- 提高交付系统的质量
  每个模型都可以作为“质量闸门”
- 融合不同涉众的观点
  涉众可以具有完全不同的，但却相互关联的观点或考虑
- 改进团队工作能力
  模型形成了不同项目角色之间的“合约”基础
- 开发效率更高
  可以贯穿整个生命周期重用模型及模型中的元素
- 提高软件治理能力


IBM
谢谢大家
Innovate2010