



IBM Software Group

成功地管理软件外包项目

IBM软件项目外包管理方案

Rational software

张剑平
IBM SWG/Rational



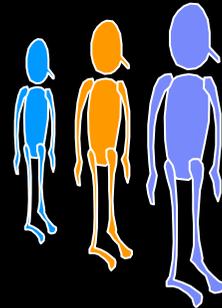
@business on demand software

议程

- ➔ ■ IT项目成功的关键
 - 如何管理外包软件开发项目
 - 软件项目外包管理最佳实践
 - IBM 软件开发平台



IT 项目的成功标志



质量

项目成功



成本



进度



项目成功的三个维度

- 质量
 - ▶ 系统能够满足业务部门的需求
 - ▶ 系统上线之后能够可靠稳定地运行
 - ▶ 能够支持大容量用户的并发访问
- 进度
 - ▶ 供应商能够按时交付产品
 - ▶ 系统能够按时上线
- 成本
 - ▶ 把开发成本控制在预算之内
 - ▶ 将系统投资的回报最大化



系统质量的常见问题

- 系统运行不稳定，发生异常宕机
- 系统需要定期重启才能正常工作
- 无法支持大容量的用户访问
- 在繁忙时段系统响应时间太慢
- 不能完全满足业务需求
- 客户在系统部署前对系统质量缺乏信心
- 系统无法快速响应业务需求的变化



质量问题的原因

问题	原因	IBM SDP解决方案
系统不稳定	系统的可靠性不高	PurifyPlus
需要定期重启	系统存在内存泄漏	PurifyPlus
无法支持大容量用户	系统的性能没有达到设计指标	压力测试解决方案
响应速度慢		
无法满足业务需求	对客户的业务需求理解不充分	需求管理解决方案
对质量缺乏信心	缺乏一些客观的指标来帮助判断系统质量	测试管理解决方案
无法响应业务需要	系统难于扩充、维护	可视化建模技术



项目进度方面的问题

- 有那么多项目同时进行, 很多时间都被花在
 - ▶ 协调各个供应商之间的协作, 但有限的资源(人力、设备等)没有被有效使用在最关键的项目上
 - ▶ 整理项目管理数据, 如项目进度、人员工时等
- 无法快速响应业务需求的变化
 - ▶ 市场竞争要求每月每周都能推出新业务
 - ▶ 开发团队无法快速地推出相应的版本
- 不能按时交付高质量的软件产品
 - ▶ 软件项目经常延期
 - ▶ 或系统匆匆按时上线, 但有很多质量问题



保证项目开发进度

- 对企业内部的所有项目组合进行综合管理
 - ▶ 管理项目之间的依赖关系，保证每一个项目的进度
 - ▶ 有效调度安排，避免关键资源的使用冲突
- 使用基于构件的软件架构
 - ▶ 重用已有的软件构件，提高开发效率和软件质量
 - ▶ 增强系统灵活性，易于扩展、维护
- 采用迭代化软件开发流程
 - ▶ 及早降低项目风险
 - ▶ 增加项目开发的可预见性
- 利用自动化的开发工具来提高软件开发生产力
 - ▶ 配置管理工具可以有效支持并行开发
 - ▶ 自动化测试工具可以提高软件测试效率
 - ▶ 建模工具可以自动生成文档和代码，并保证两者之间的一致性



软件成本方面的问题

- 缺乏项目成本控制的手段
 - ▶ 项目开发成本没有量化统计
 - ▶ 没有对项目的投资回报做量化分析
 - ▶ 与供应商的合同定价缺乏足够的依据
- 软件系统的生命周期太短
 - ▶ 很多系统上线几年后就无法继续满足业务需求，需要重新开发
 - ▶ 重复性开发的东西太多
 - ▶ 缺少系统维护所需要的需求、设计等文档



软件开发成本的控制

- 量化管理项目开发的成本
 - ▶ 分析每一个项目的投资回报
 - ▶ 细化项目成本的组成，合理定价
- 延长软件系统的生命周期
 - ▶ 构建一个灵活的系统架构，能够很好地响应业务需求的变化
 - ▶ 增强系统的可维护性，建立规范的软件开发体系

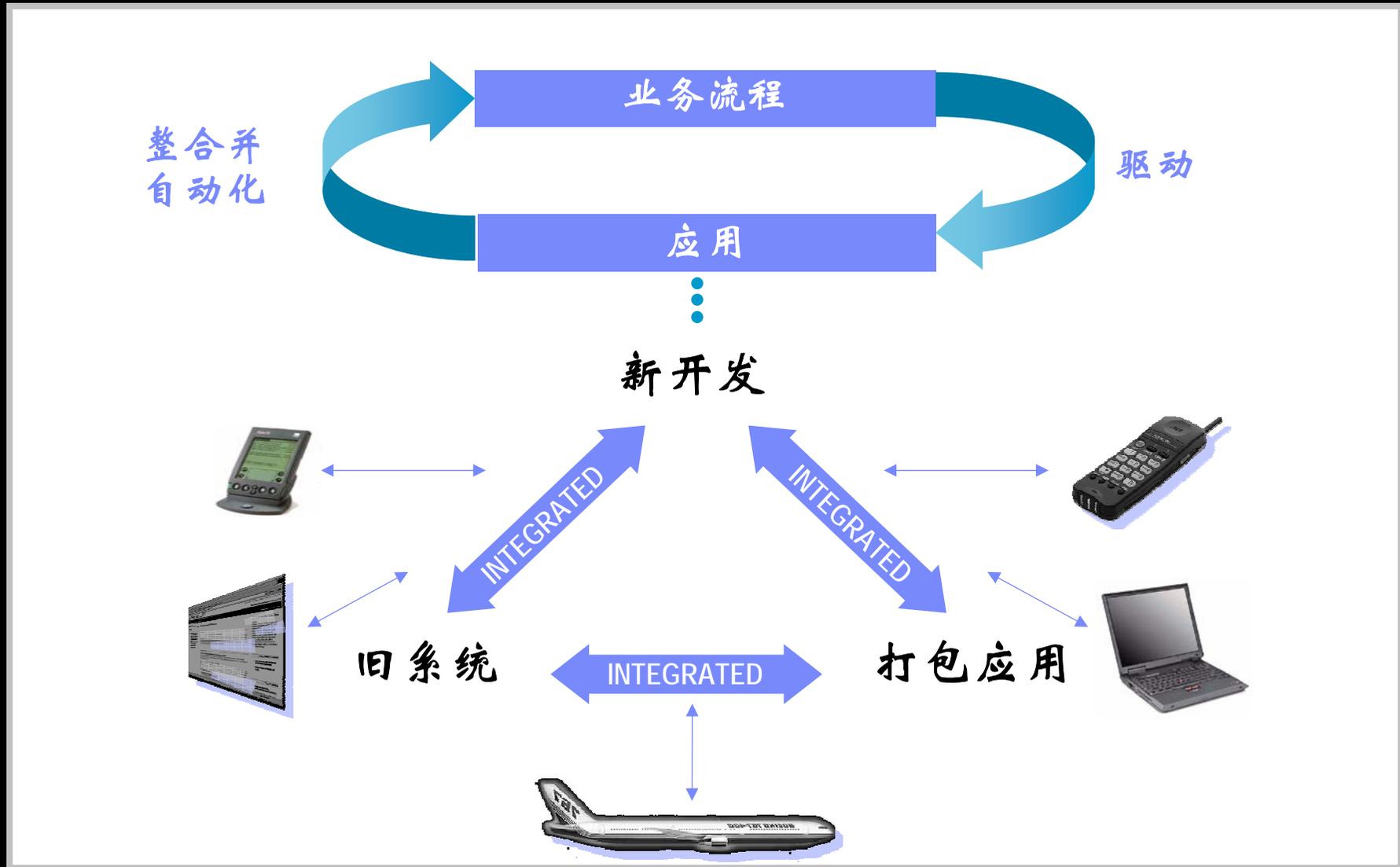


议程

- IT项目成功的关键
- ➔ ■ 如何管理外包软件开发项目
- 软件项目外包管理最佳实践
- IBM 软件开发平台



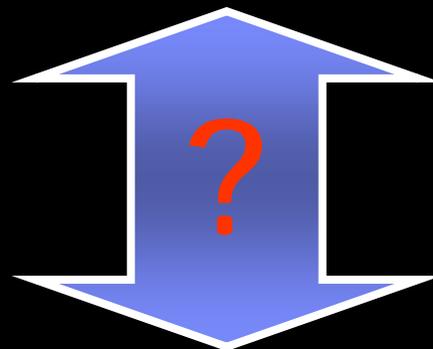
应用系统建设的核心是开发和整合



应用系统建设的两难选择

自主开发

- 项目进行过程中，人力紧张；项目完成后，人力空闲；
- 技术进步非常快（编程语言，操作系统，数据库等），需要大量的培训投资来追赶技术进步；



全部外包

- 项目的进度和质量难于控制；
- 文档不全或不准确，造成维护困难；



甲方担心的主要问题

■ 如何保证项目满足自己的业务需求

需求管理

■ 如何保证系统满足公司扩张发展需要

架构设计

■ 如何保证项目按照既定的时间表完成

过程跟踪控制

■ 如何保证项目质量

质量测试

■ 如何确保参与项目的多个组织进行顺畅的沟通

统一平台

■ 如何保证自身对系统有充分的了解

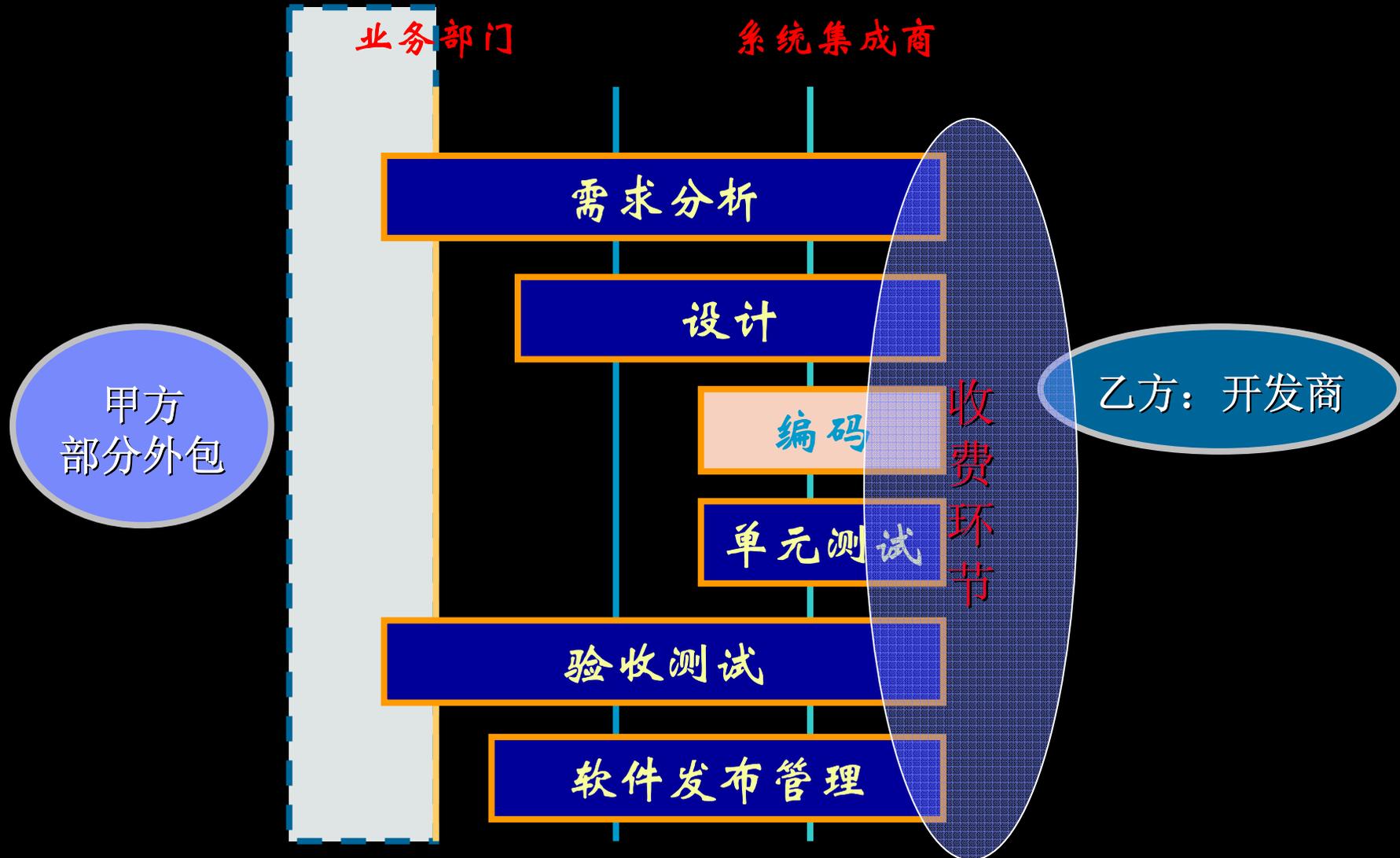
变化的组织结构
部分外包参与项目



系统建设中组织结构模型



国内企业外包主要模型

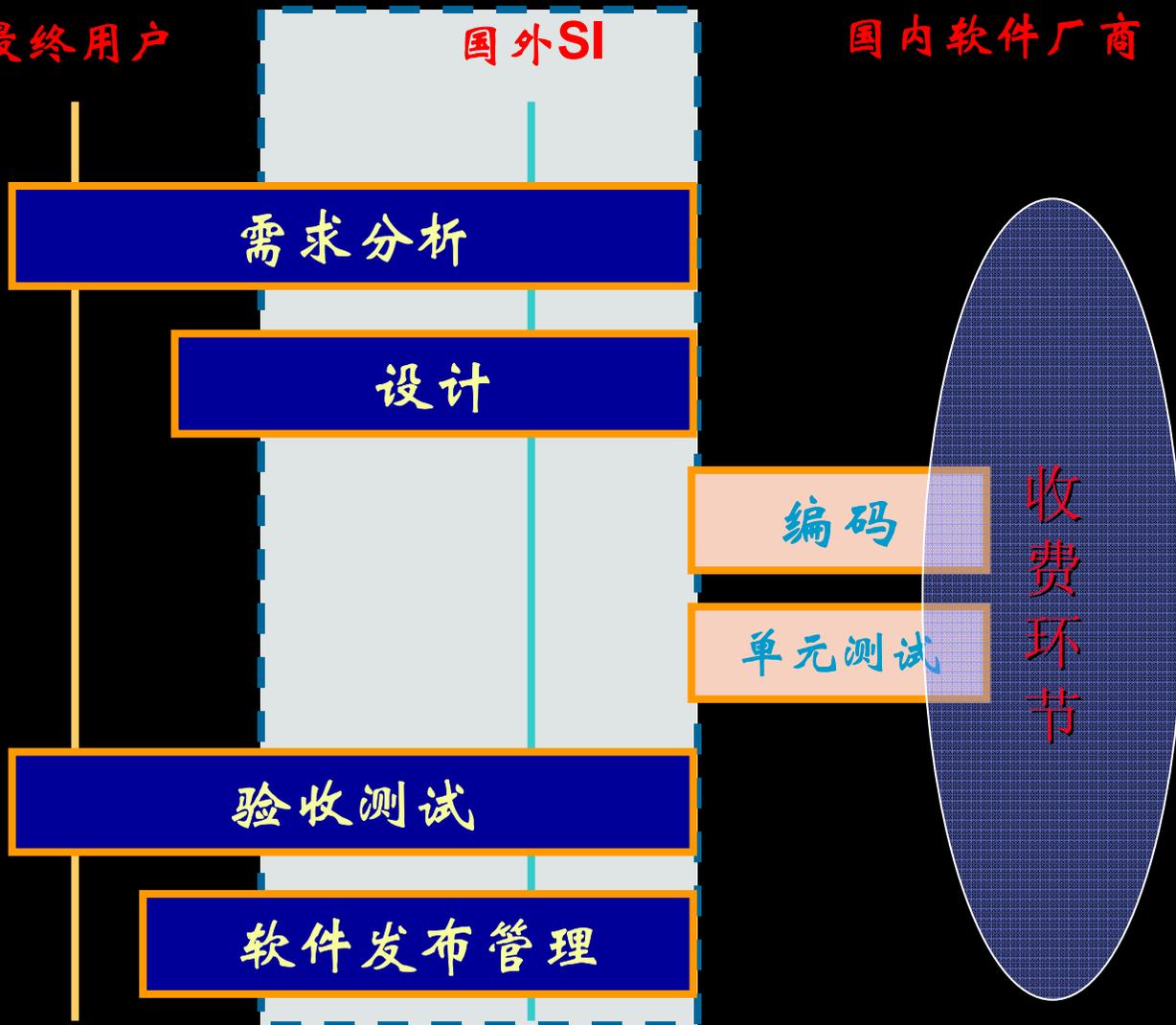


国际软件外包模型现状

最终用户

国外SI

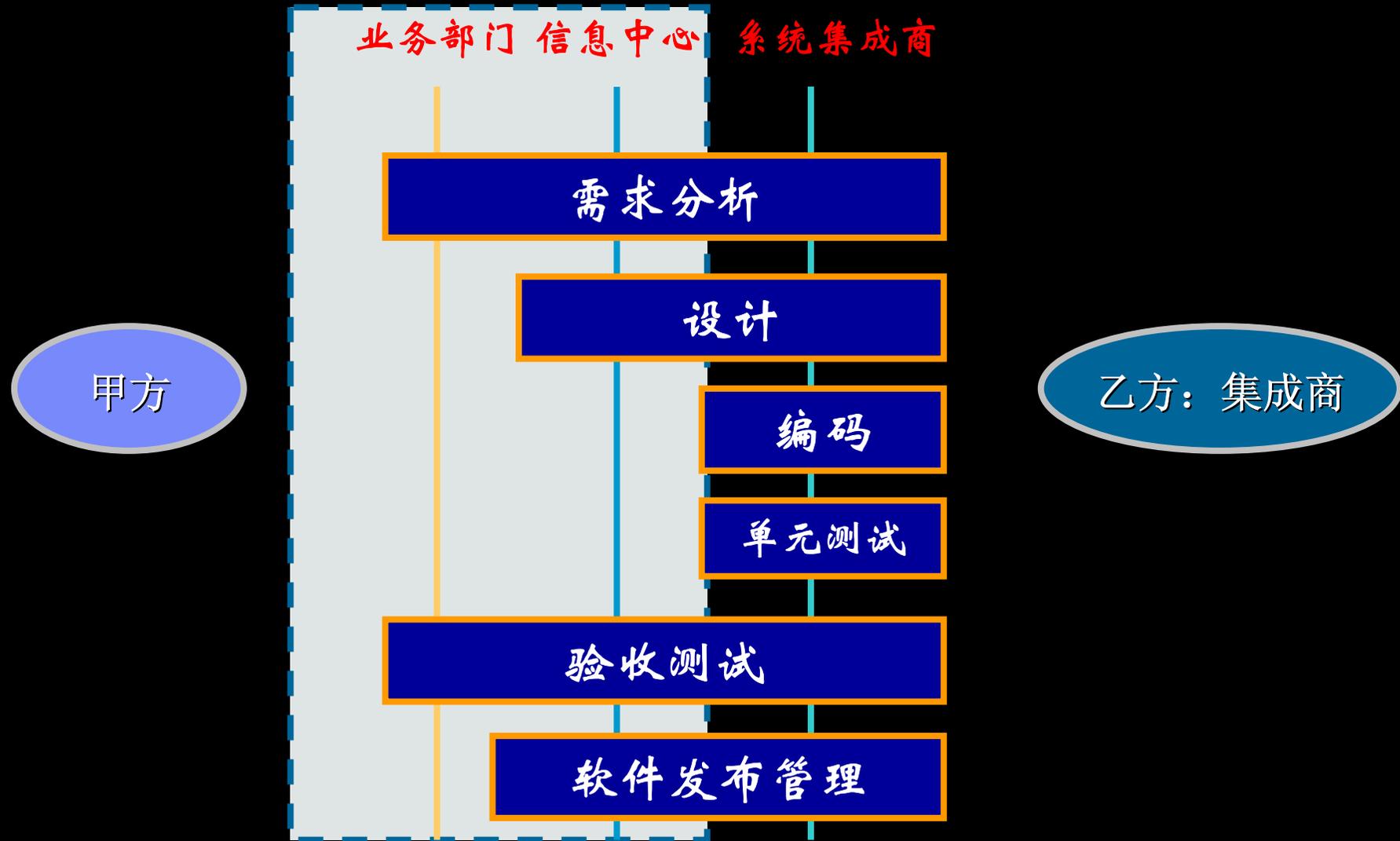
国内软件厂商



什么是最佳组织结构和模型？？？



项目建设最佳组织结构



议程

- IT项目成功的关键
- 如何管理外包软件开发项目
 - ➔ 管理所有的外包项目
 - ▶ 明确软件需求，提高需求质量
 - ▶ 监控项目过程，控制项目风险
 - ▶ 建立验收标准，保证软件质量
 - ▶ 建立变更管理流程，全方位的管理
- 软件项目外包管理最佳实践
- IBM 软件开发平台



企业信息化建设需要项目管理

- 在信息化建设的过程中每年都有几十个甚至是上百个 IT 项目
- 信息部门的职责：
 - ▶ 监控众多 IT 项目的进展情况
 - ▶ 管理项目之间的依赖关系，保证项目进度
 - ▶ 协调不同项目之间的资源（人员、设备、场地等）配置
 - ▶ 控制 IT 项目投资成本，使 IT 投资更好地为业务发展服务



建立企业级的项目管理平台

不仅仅是管理单个的项目，而是管理整个企业内的所有IT项目，并且使企业在IT方面的投入更好地适应业务发展方向和目标。



让IT更好地为业务服务



IBM Rational Portfolio Manager

结合业务发展战略，把所有的工作整合在一个流程和平台上



把握 IT 投资的方向



财务总监

我们在 IT 项目上花了多少钱？
这些项目的投资回报率有多少？

给我一分钟时间，我马上从 RPM
平台中生成一个报告给你。

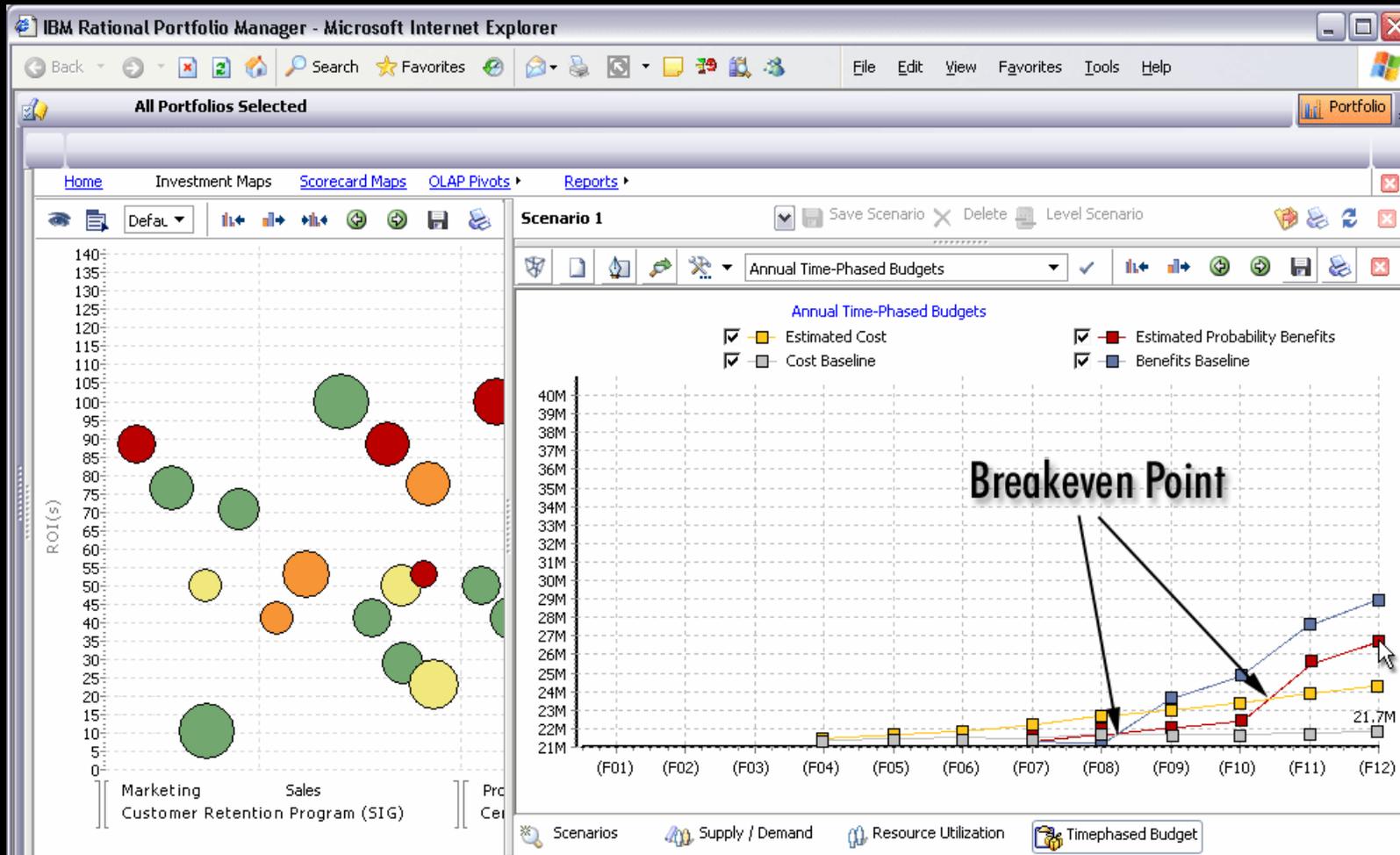


信息中心主任

- RPM 能够帮助统计各 IT 项目的成本开支、报资回报等
- 并生成各种图表和报告来展示这些数据



RPM中的投资回报分析



了解项目进展情况



信息中心主任

我需要了解项目进展状况

所有项目的进展情况都在我们的 RPM 平台上，您可以了解项目的启动日期、完成了多少工作量、剩余多少工作量、目前的进度是否按照计划进行等等



项目经理

- RPM 能够有效地帮助项目经理管理项目的进度
- 企业管理层也可以迅速地获得所有项目组合的最新数据



RPM 监控所有项目的状况

PMODemo6: http://192.168.2.4/webapp/PMODemo6/PMODemo6/中文/~Rene

选中全部项目组合

描述 依赖关系 人员配置 文档管理 范围管理 日历 模板 项目组合 显示

项目组合 提案 可交付成果 工作产品 概要任务 里程碑 任务

[Local] Schedule Dates/Work

快速状态

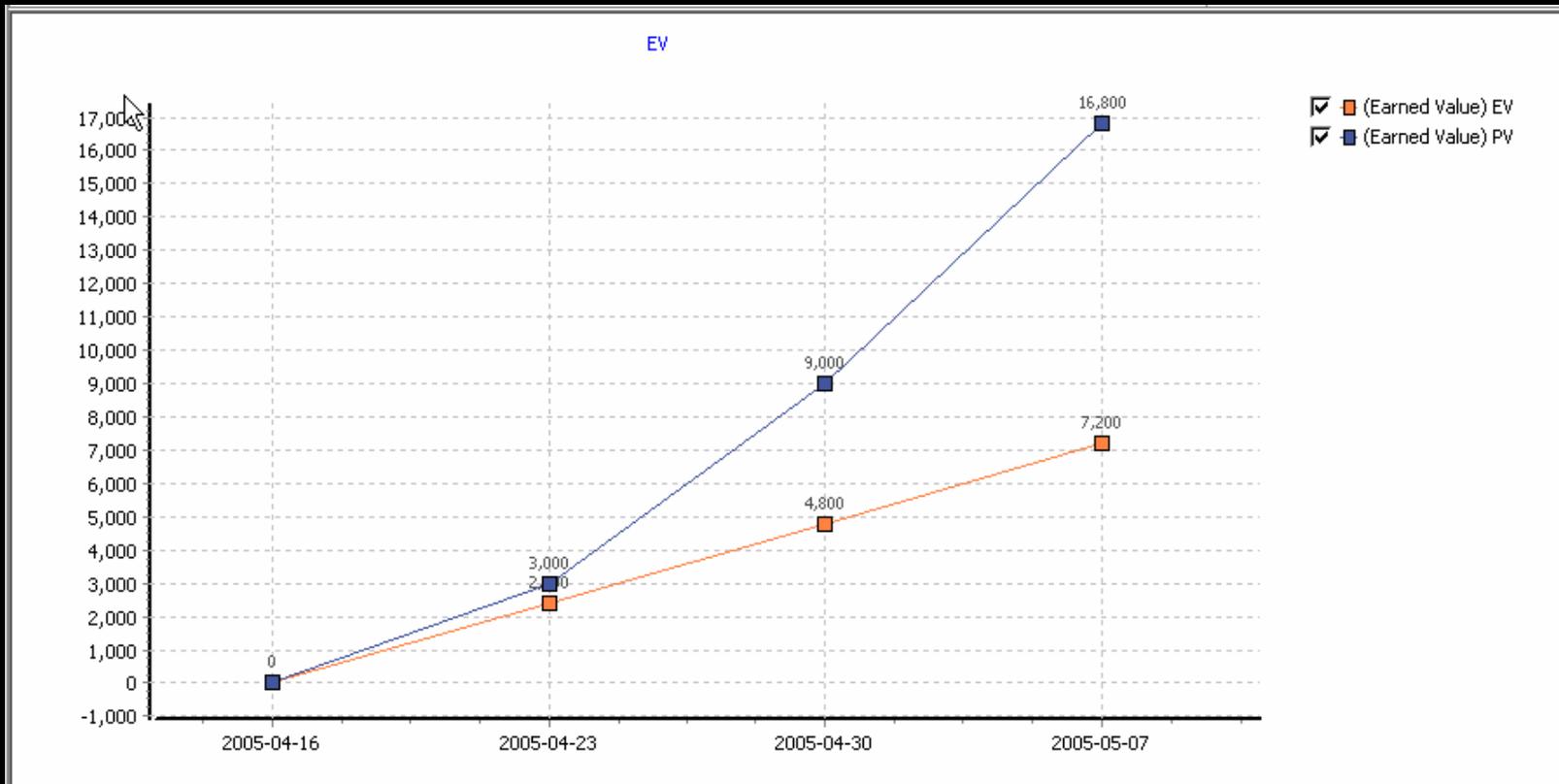
名称	开始	完成	工期	工作	%工期	剩余	实际	%完成	%工作
IT Portfolio									
Proposals									
Projects									
Exchange Virus Scanning So...	2003-04-01	2003-05-09	24.00 d		43.5 %				
QA Test Environment	2003-05-05	2004-11-25	327.13 d	666.00 h	16.7 %	581.00 h	85.00 h	12.8 %	12.8 %
Security Audit	2003-07-11	2003-10-31	66.00 d		80.0 %				
Buy/Sell Indications of In...	2003-04-01	2003-10-16	114.14 d		4.0 %				
Financial Software Develop...	2003-04-29	2004-12-15	341.78 d	1,785.00 h	6.6 %	1,641.65 h	143.35 h	9.3 %	8.0 %
New Software Development	2003-04-14	2003-08-07	67.25 d	1,512.00 h		1,512.00 h			
Financial Decision Support...	2002-10-08	2002-11-21	27.00 d						
Application Support	2003-04-30	2003-04-30	1.00 d						
Reservation System Upgrade	2002-08-06	2004-12-29	501.25 d	4,362.00 h		4,250.00 h	112.00 h	5.9 %	2.6 %
Facility Maintenance	2003-04-30	2003-04-30	1.00 d						
Real Time Program	2003-06-04	2003-11-21	99.00 d		74.2 %				
Hardware Maintenance	2003-04-30	2003-04-30	1.00 d						
Web Site Development Support	2003-06-02	2003-10-30	109.00 d	11,430.00 h		11,430.00 h			
Project R	2003-11-14	2003-12-16	23.00 d	328.00 h		328.00 h			
Network Infrastructure Upg...	2003-12-01	2005-02-15	317.00 d	9,328.00 h	4.9 %	8,547.00 h	781.00 h	8.4 %	8.4 %
Organization									

欢迎您使用 IBM Rational Portfolio Manager!

帮助 关于



利用挣值曲线来管理项目进度



- 计划完成工作预算成本 (BCWS/PV): 项目的原定计划
- 已执行工作预算成本 (BCWP/EV): 项目执行的实际进度



保证项目进度



供应商

我在 **RPM** 平台上发现我们项目中有几个任务被延误了一周以上的时间，其中有一个还处在关键路径上，你们能尽快解决这个问题吗？



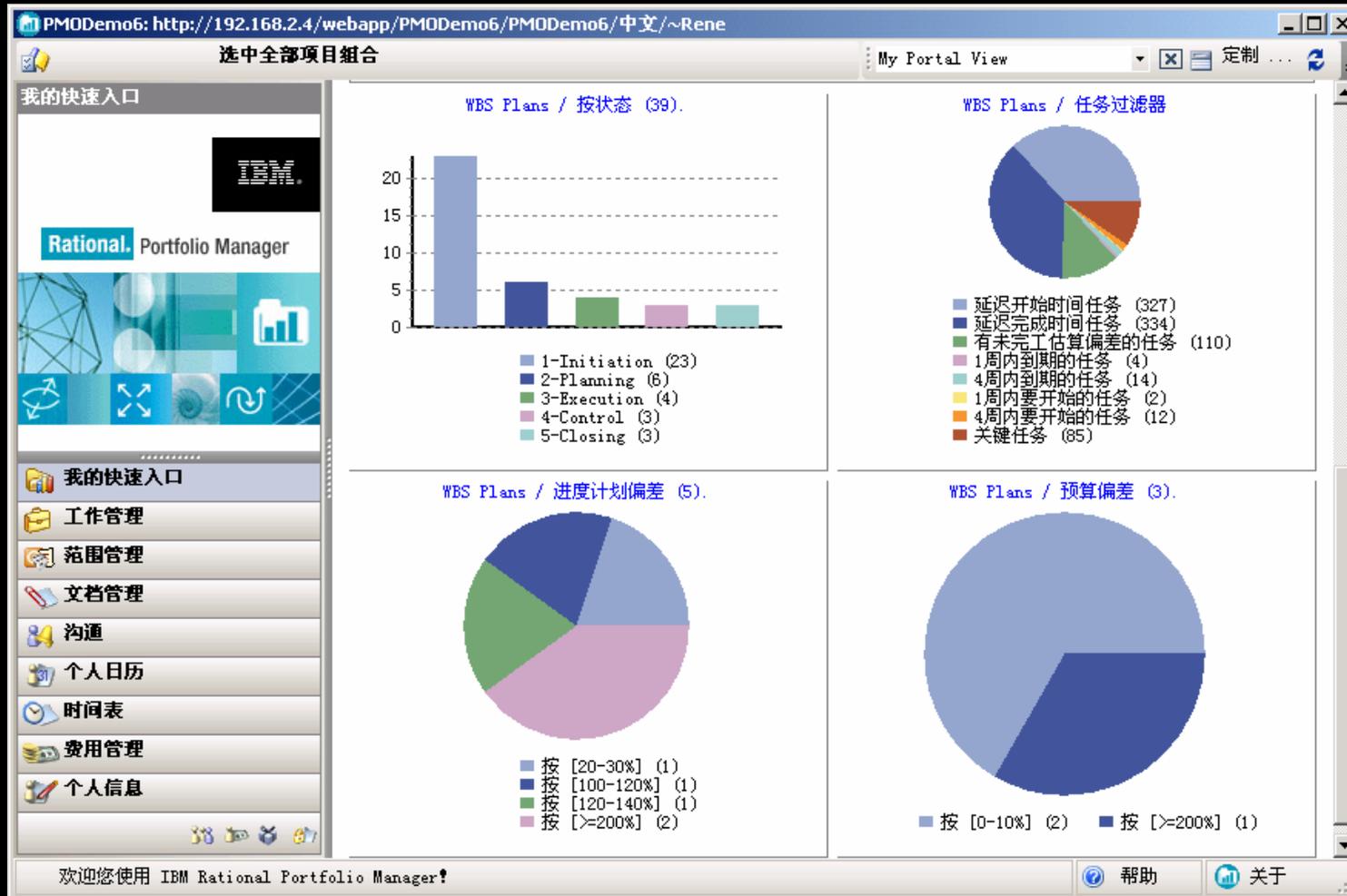
项目经理

是的，我们会尽快增加人手，赶上进度。

- RPM 能够对项目组合进行有效管理，维护任务之间的依赖关系，保证项目进度的正常进行



RPM 实时汇报项目的进度和预算偏差



管理多个项目之间的依赖关系



供应商

下一周我们的新一代业务系统就要在服务器上安装了，可是我从 RPM 系统中看到你们的进度表显示你们的“机房和网络布线”项目有延误，这会影响到我们总体的工作进度。

明白，我们会周末加班把赶上原定计划的。



项目经理

- RPM 中有以下三个层次概念
 - ▶ 项目组合 (portfolio): 一个企业或部门内部所有的项目
 - ▶ 项目群 (program): 一组相互协作的项目组合成一个项目群
 - ▶ 项目 (project): 为了达到一定的业务目标而进行的一组活动
- RPM 可以在多个项目之间建立依赖关系，协调多个项目之间的合作，保证大项目的顺利进行



管理项目风险和问题



项目成员

我们在项目实施过程中遇到了好几个问题，请协调解决。

是的，我已经在 **RPM** 系统中看到这些问题了，我已经安排了厂商的专家来解决这些问题，相关任务已经添加到项目计划中。



项目经理

- RPM 可以有效管理项目风险和项目过程中出现的各种问题
- 项目组成员可以针对这些问题进行沟通 and 讨论，并制定出相应的解决方案
- 解决方案可以被方便地添加到项目计划中去



RPM 管理项目中的风险和问题

PMODemo6: http://192.168.2.4/webapp/PMODemo6/PMODemo6/中文/~Rene

选中全部项目组合

描述 人员配置 文档 模板 显示

范围管理

优先收藏夹

刷新

- 我的范围元素
- 全部范围元素
- 全部范围元素 (90)
- 需求
- 更改请求
- 服务请求

我的快速入口

- 工作管理
- 范围管理
- 文档管理
- 沟通
- 个人日历
- 时间表
- 费用管理

文件夹 需求 更改请求 服务请求 行动 缺陷 问题

名称	版本	更新时间	负责人	月 19	十月 26
Issues			Jaime		
Hardware Malfunction	0.3	2003-05-09 16:40:00	Jaime		
Loss of critical resource	0.3	2003-05-09 16:40:00	Jaime		
Change Requests			Jaime		
New Monthly Report	0.2	2003-05-09 16:36:00	Jaime		
Month end calculation change	0.2	2003-05-09 16:36:00	Jaime		
Risks			Jaime		
Lack of technical resources	0.5	2003-05-09 17:34:00	Jaime		
Actions			Jaime		
Call Meeting	0.1	2003-05-09 17:41:00	Jaime		
Review Minutes	0.1	2003-05-09 17:42:00	Jaime		
Place Orders	0.1	2003-05-09 17:42:00	Jaime		
Issues			Jaime		
Change Requests			Jaime		
Risks			Jaime		
Issues			Jaime		
Change Requests			Jaime		
Risks			Jaime		
Client Requirements			Jaime		

[计数: 51]

欢迎您使用 IBM Rational Portfolio Manager! 帮助 关于



团队的协作



项目经理

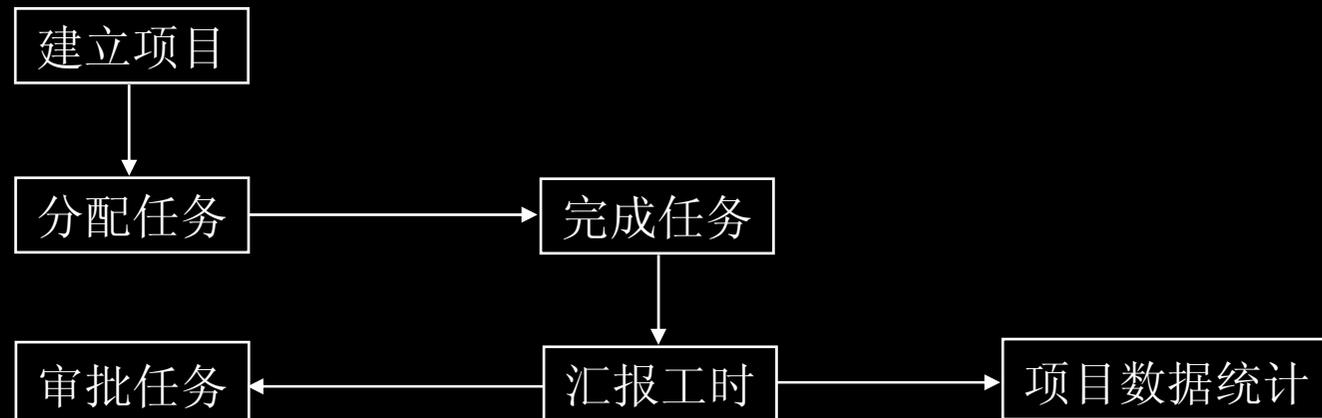


项目成员



企业管理层

RPM 项目
协作流程



- RPM 记录并追踪所花的时间的成本
 - ▶ 汇报个人每周的工作情况 (timesheet)
- 统一平台, 提升协作和沟通效率
 - ▶ 了解分配给自己的任务
 - ▶ 与项目组其他成员进行有效的沟通



项目组成员在RPM中填报个人每周的工作时间

PMODemo6: http://192.168.2.4/webapp/PMODemo6/PMODemo6/中文/~Alex

IBM Rational Portfolio Manager

开始星期: 2005-04-02 到: 2005-04-08 提交申请批准

缺省步骤 个人任务 步骤

名称	预期	周(明细)	概要										
			开始	完成	总计/工作	工作/周	周一-四	周二-五	周三-六	周四-七	周五-八	剩余	未完工估算工作量
我的工作包	✓	2004-11-29 2004-12-15	103.50								103.50	103.50	
管理/个人	✓												
Administrative Tasks	✓												
Organization Meetings	✓												
Personal Time Off	✓												
Travel Time	✓												
General Administration	✓												
Sick Time	✓												
Jury Duty	✓												
Financial Software Development	✓	2004-11-29 2004-12-15	102.00								102.00	102.00	
Development	✓	2004-11-29 2004-12-15	102.00								102.00	102.00	
Developer testing (prim...	✓	2004-11-29 2004-12-15	102.00			0.00					102.00	102.00	

每周合计 常规: 0.00 特殊: 0.00

欢迎您使用 IBM Rational Portfolio Manager! 帮助 关于



有效利用企业内部的资源



项目经理A

我们有个项目下周需要借用一下你们的 DBA

让我上 RPM 去看一下他的日程安排



项目经理B



开发商

下周就要做验收测试了，我们需要两台服务器和五台PC机做为测试平台。

让我用 RPM 里的资源管理器查一下...
你们需要什么配置的服务器？



项目经理



管理企业内部各种资源

- RPM 中的每个资源（包括人、设备或会议室等其它设施）都有一张属于它的时间表，上面记录了该资源被使用的详细情况
- 项目经理可以跨项目地调用资源，并利 RPM 平台的帮助来有效利用各种资源，避免资源使用上的冲突



议程

- IT项目成功的关键
- 如何管理外包软件开发项目
 - ▶ 管理所有的外包项目
 - ▶ 明确软件需求，提高需求质量
 - ▶ 监控项目过程，控制项目风险
 - ▶ 建立验收标准，保证软件质量
 - ▶ 建立变更管理流程，全方位的管理
- 软件项目外包管理最佳实践
- IBM 软件开发平台



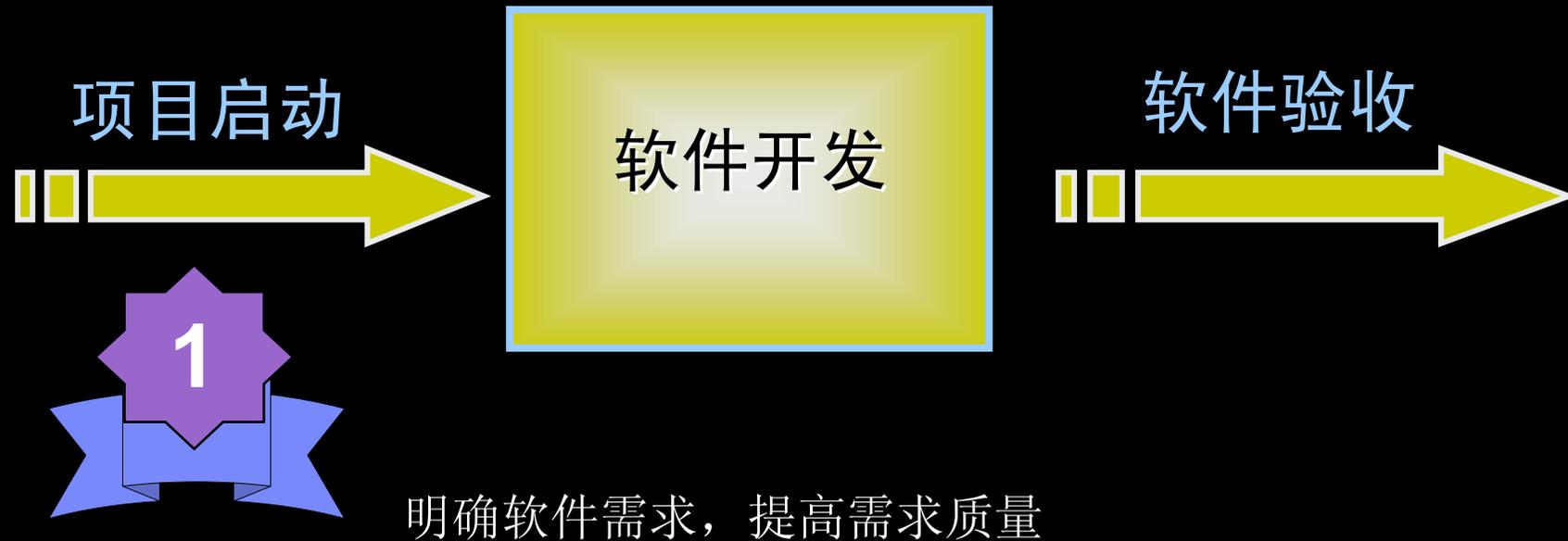
管理软件项目外包的全过程

- 把握好外包项目的三个关键阶段



管理软件项目外包的全过程

- 把握好外包项目的三个关键阶段

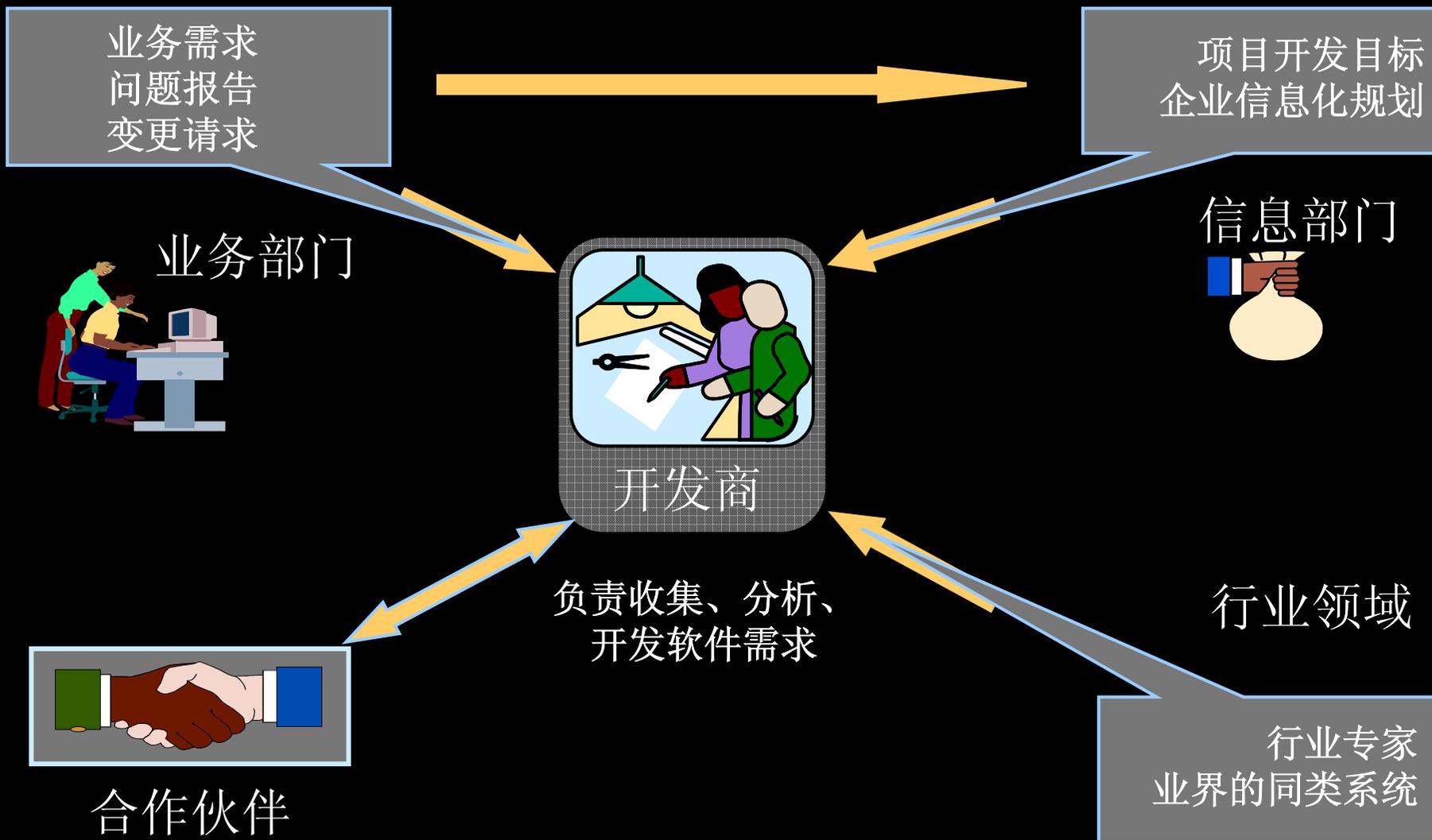


关于软件需求的常见问题

- 开发商没有完全理解企业的业务需求
- 需求会随着项目的进展而发生变化
- 开发商提供的需求文档难于被业务部门所理解



需求从哪里来



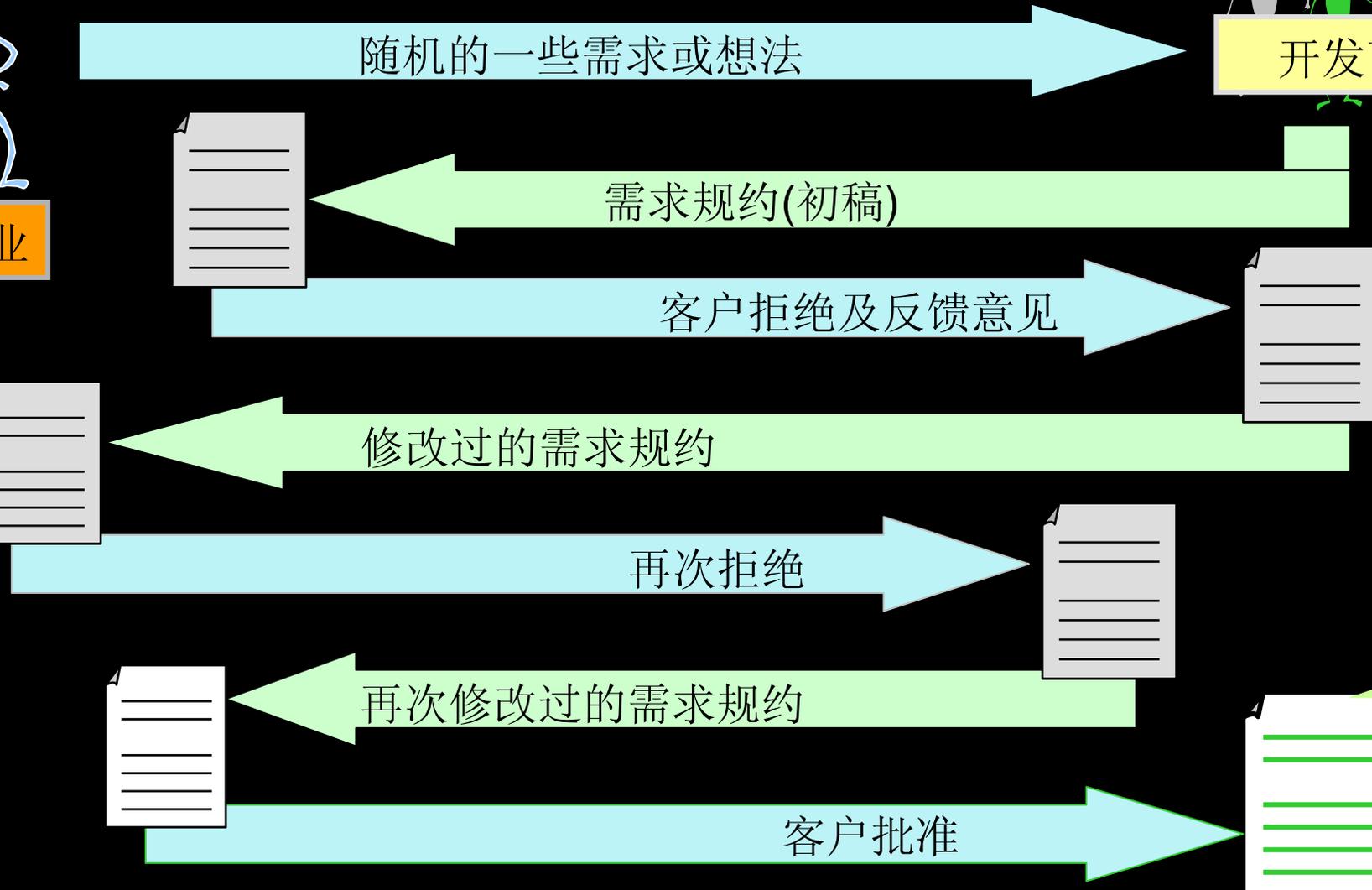
需求开发是一个不断反复的过程



企业



开发商



需求的质量维度

- 正确性：必须能够支撑和满足业务部门的业务需求
- 完备性：没有遗漏任何来自于用户的重要需求
- 一致性：在需求文档中对于需求内容的描述要前后一致
- 无二义性：不同的人对于同一需求阐述应该具有相同的理解
- 易于理解：需求应该易于被业务部门所理解



保证需求质量的手段

- 保证开发商完全理解业务流程
 - ▶ 通过业务建模来了解现有的业务流程，理清业务实体之间的关系
- 站在用户的立场上来描述需求
 - ▶ 用例建模技术完全是从最终用户的角度来描述系统功能
- 采用统一的需求描述手段
 - ▶ 增强需求的可理解性，便于对需求内容进行审核
- 辅助以其他的需求开发技术
 - ▶ 用户界面原型：及早收集业务部门的反馈
 - ▶ 需求研讨会：向业务部门解释软件是如何实现业务流程的
- 提交业务需求和软件需求之间的追踪矩阵报告
 - ▶ 确保所有的业务需求都被软件需求覆盖了



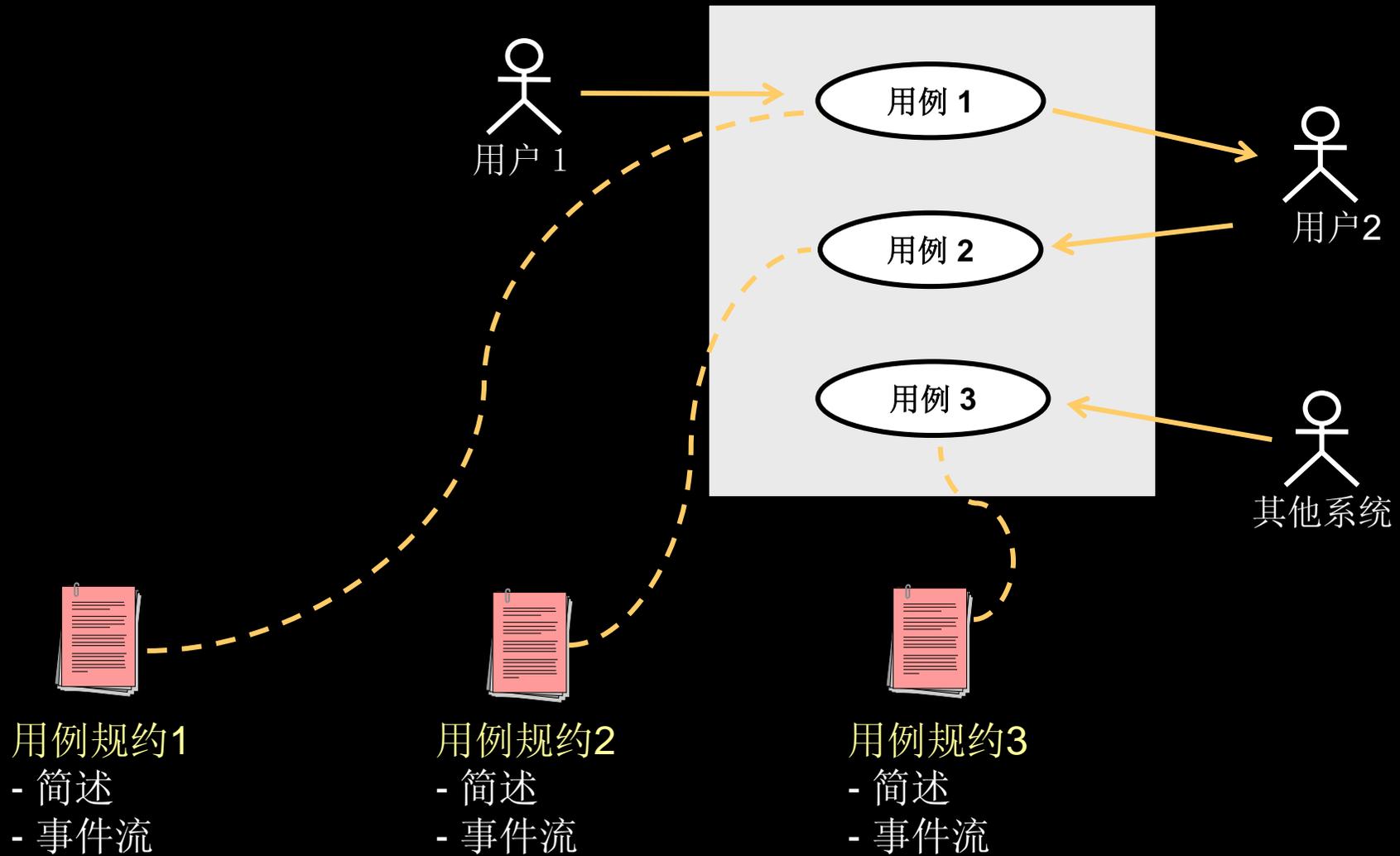
业务建模的作用

- 整理企业现有的业务
- 描述企业内部的组织架构
- 定义不同的角色分工及他们之间的协作
- 理清业务实体之间的关系（领域建模）
- 记录企业为实现业务运作而制定的内部协作流程
- 帮助业务部门制定流程改进的方向
- 帮助开发商了解软件系统的应用环境



用例模型的主要内容

用例图



业务需求和软件需求之间的追踪矩阵

- 保证所有的业务需求都由相应的软件需求所满足

The screenshot displays the Rational RequisitePro interface with a traceability matrix. The left pane shows a hierarchy of Use Cases (UC) and Features (FEAT). The main area shows a grid where arrows indicate the relationships between UCs and FEATs.

Relationships: - direct only

UC	FEAT1	FEAT2	FEAT3	FEAT4	FEAT5	FEAT6	FEAT7	FEAT8	FEAT9	FEAT10	FEAT11	FEAT12	FEAT13	FEAT14	FEAT15	FEAT16	FEAT17	FEAT18	FEAT19	FEAT20	FEAT21	FEAT22	FEAT23	FEAT24
UC1: Apply for...																								
UC2: Execute Trade																								
UC3: Get Quote																								
UC4: Manage...																								
UC5: Distribute News																								
UC6: Review Account																								
UC7: Maintain...																								
UC8: Report Tax...																								
UC9: Maintain Web...																								
UC10: Resolve...																								
UC11: Learn System...																								

View saved as STRQ Hierarchy | 11 requirements

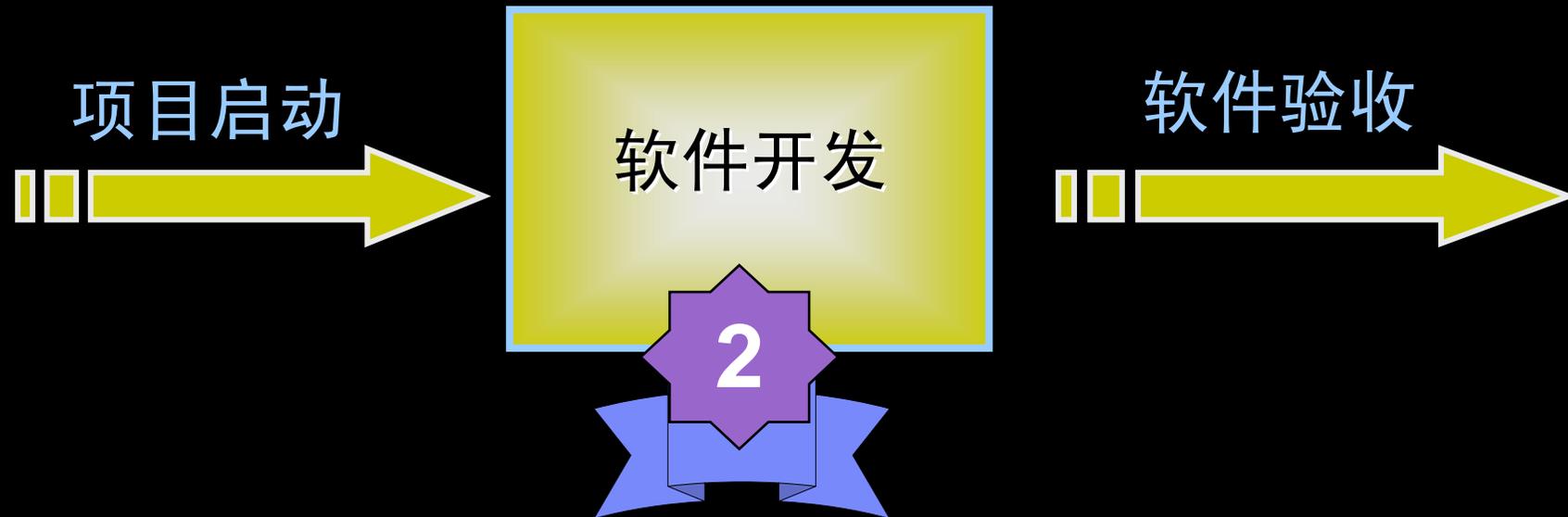
项目启动阶段主要交付件

- 业务模型
- 项目前景文档(初稿)
- 用例模型(初稿)
- 软件项目开发计划
- 风险管理计划



管理软件项目外包的全过程

- 把握好外包项目的三个关键阶段



监控项目过程，控制项目风险



传统的开发模式：一次性交付系统

- 很多问题要到验收阶段才会暴露出来
- 难于控制项目的进度，经常导致项目延期
- 开发过程不透明，很难监控开发的进展情况



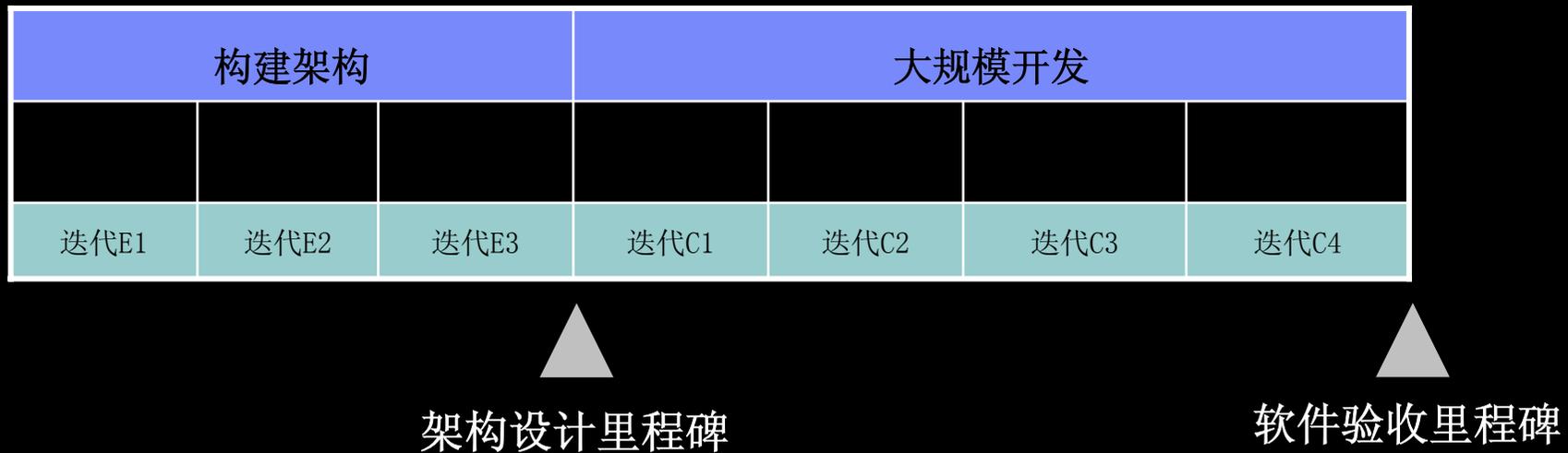
加强开发中间过程的监控

- 变一次验收为多次检验
- 在每个里程碑处设立检验点



软件开发的两个主要阶段

- 构建架构
 - ▶ 定义所有的需求：所有的用例及用例场景
 - ▶ 选择一些关键用例场景，搭建系统架构
 - ▶ 提交系统架构原型



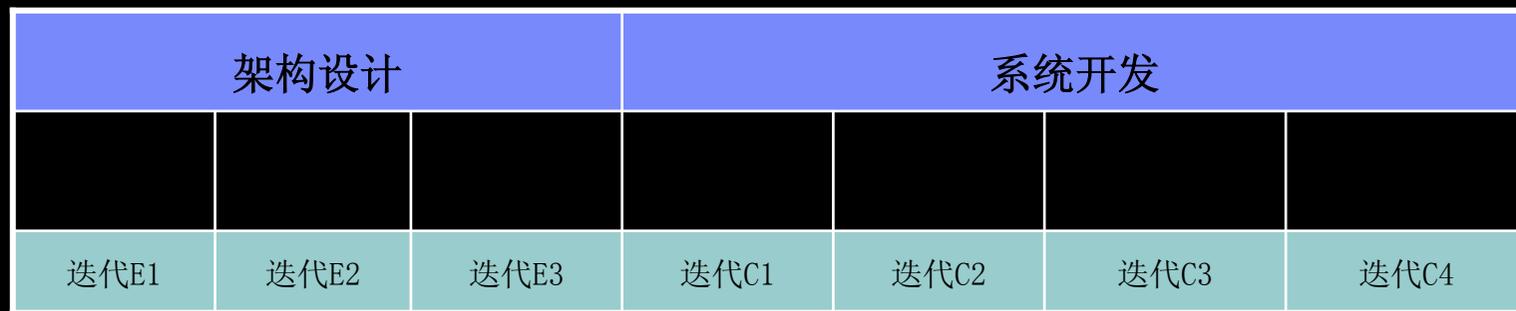
软件开发的两个主要阶段

- 大规模开发
 - ▶ 尽快完成软件产品的开发
 - ▶ 尽可能降低开发成本，优化开发资源和避免不必要的返工
 - ▶ 在保证开发进度的同时达到足够的软件质量
 - ▶ 获得一些有用的版本（alpha, beta等）



设置中期检验点

- 在架构设计里程碑处设置中期检验点
- 对系统原型进行中期验收
 - ▶ 考查系统功能、性能指标
 - ▶ 确认系统是否能与生产环境中正常工作
 - ▶ 收集业务部门的反馈意见



每一次迭代结束都是一个检验点



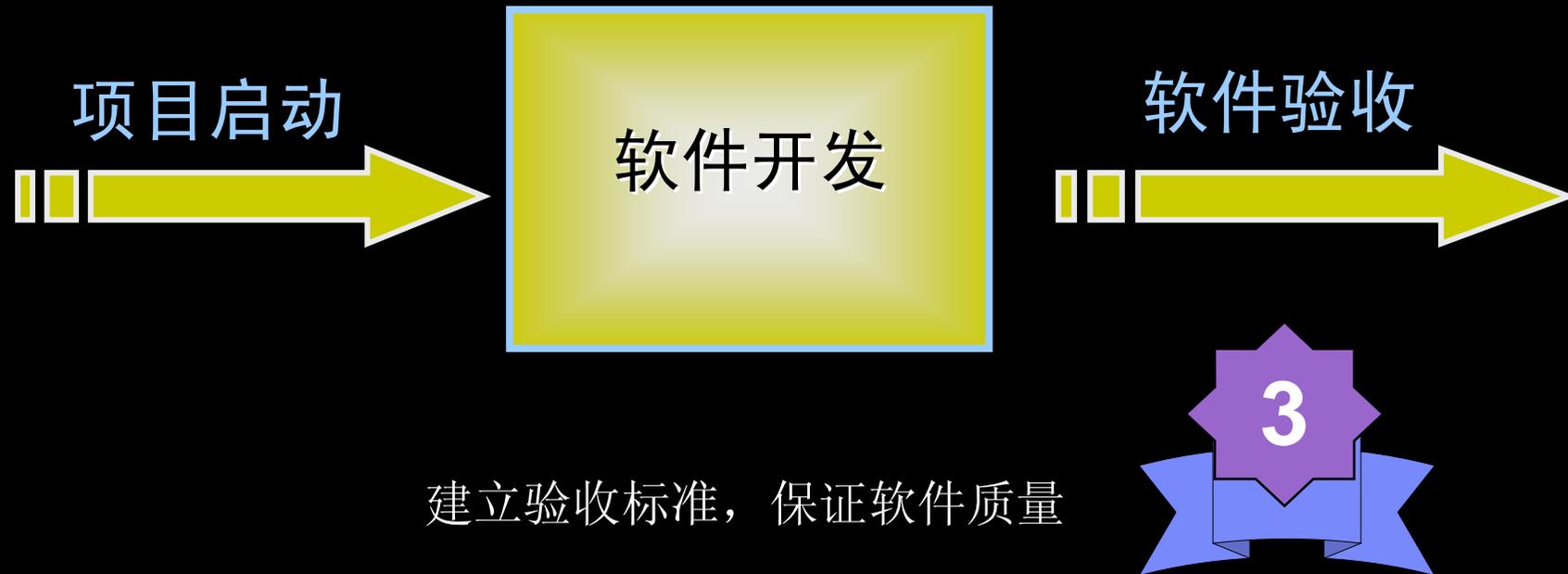
中期验收主要交付件

- 软件项目开发计划
 - ▶ 在项目开发过程中不断调整，包括下一阶段的工作计划
- 风险管理计划(调整过的)
- 项目前景文档(终稿)
- 用例模型(终稿)
- 系统设计模型
- 系统架构原型
 - ▶ 可运行的系统，实现了一些关键的用例场景
- 中期验收测试计划及报告



管理软件项目外包的全过程

- 把握好外包项目的三个关键阶段



验收的两个方面

- 验证 (Verification)
 - ▶ 评估系统在功能上是否和需求定义相一致
 - ▶ 检验系统在技术上是否达到一定质量标准 (可靠性、性能等)
- 确认 (Validation)
 - ▶ 评估系统是否真正满足了生产环境和业务运作的需要
 - ▶ 需要业务部门的参与
 - ▶ 需要将系统部署到实际生产环境中去进行检验



软件质量的多个维度



验收的原则

- 质量标准应该是可验证的
- 尽量采用一些量化指标
- 验收的不仅是系统本身，还应该包括项目的所有工件



软件验收

- 功能：
 - ▶ 开发商应提交其测试计划和测试报告
 - ▶ 提交需求-测试用例覆盖率报告
 - ▶ 组织业务部门进行业务功能确认测试
- 性能：
 - ▶ 利用测试工具对系统进行压力测试
- 可靠性：
 - ▶ 开发商应提交代码覆盖率报告并达到相应的覆盖率指标
 - ▶ 利用测试工具来找出内存泄漏、系统运行不稳定方面的原因



系统交付件(1)

RUP workflow	交付件	说明
业务流程	业务模型	描述系统所处的业务背景
需求	前景文档	项目开发的目标、范围及系统的主要特性
	用例模型	定义系统的功能性需求
	补充规约	定义系统的非功能性需求(如性能、可靠性等)
	词汇表	定义项目开发中所用到的专业词汇
分析	设计模型	采用UML语言记录的系统设计结果
	数据模型	采用UML语言记录的数据表结构
编码	源代码	所有的源代码文件
	单元测试报告	所有模块的单元测试报告, 包括测试用例报告、代码覆盖率报告等



系统交付件(2)

RUP workflow	交付件	说明
测试	测试计划	该系统是如何被测试的
	测试报告	系统测试的结果, 包括通过的测试用例数、未通过的测试用例数及相关分析
部署	安装手册	软件产品安装步骤
	发布说明	该发布版本主要增加了哪些功能、改正了哪些错误等
	用户手册	用户使用手册
	部署计划	如何将系统部署到生产环境中去
	材料清单	所有交付件的列表
	培训教材	针对最终用户的培训教材
项目管理	项目开发计划	该系统是如何被开发的
	风险管理计划	在开发过程中是如何管理风险的
	迭代计划	每一个迭代的详细开发活动计划

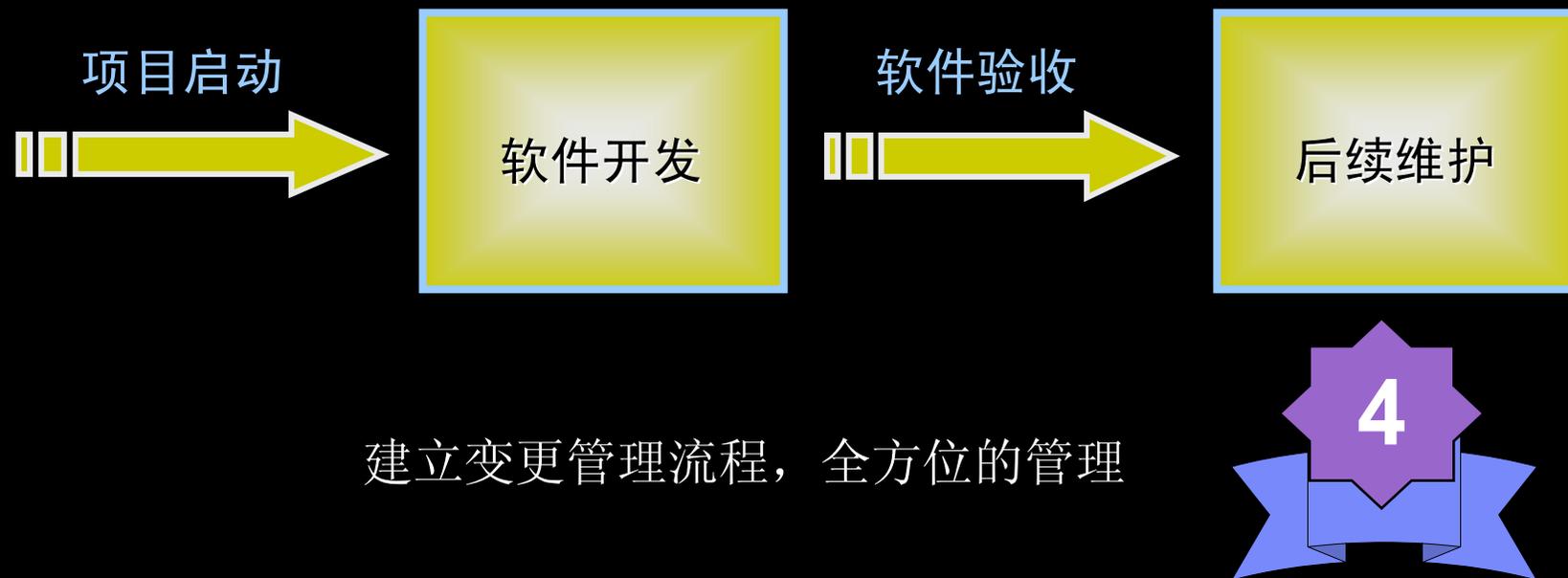


项目已经结束了， 我们可以休假吗？



软件项目外包的第四个阶段

- 把握好外包项目的第四个关键阶段



维护阶段的常见问题

- 太多的变更请求
 - ▶ 业务部门提出很多新的业务需求
 - ▶ 软件缺陷需要得到迅速修正
- 需要同时维护多个软件版本，经常发生版本错误
 - ▶ 不同地区的特殊业务要求系统有多个版本
 - ▶ 不同的运行平台也要求系统支持多个运行版本
- 开发周期太短
 - ▶ 业务部门要求系统快速实现某些新功能
 - ▶ 没有足够的时间来对系统进行完备的测试



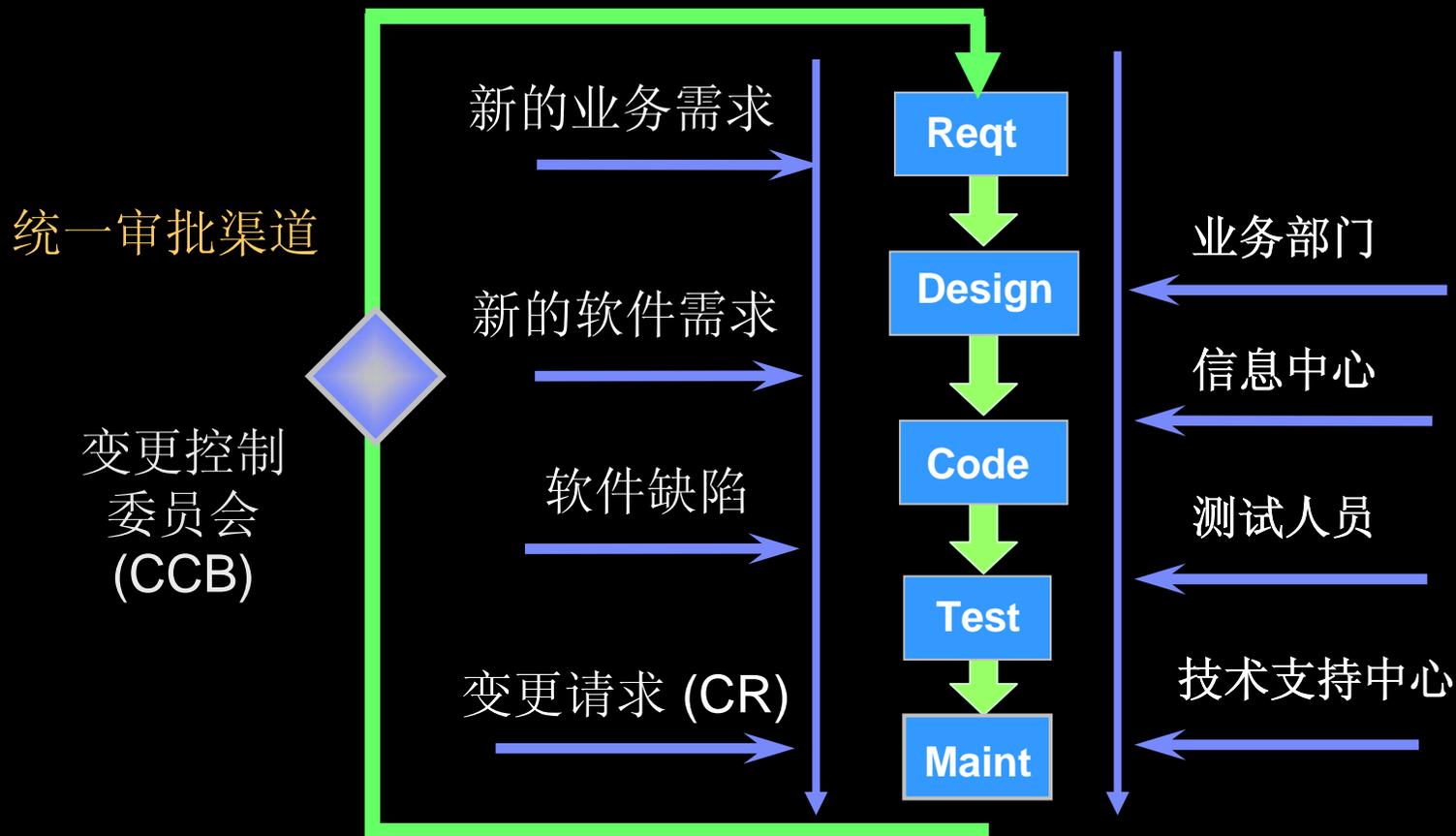
解决方案

- 建立软件变更管理流程
- 做好软件配置管理，管理好软件版本发布
- 建立基于构件的系统架构，增加系统的灵活性
- 采用测试工具进行自动化的回归测试



软件变更的统一管理流程

所有的变更请求都必须经过统一的审批渠道。





议程

- IT项目成功的关键
- 如何管理外包软件开发项目
- ➔ ■ 软件项目外包管理最佳实践
- IBM 软件开发平台



软件项目外包管理最佳实践

最佳实践

统一交付件

监控项目过程

建立验收标准

管理版本和变更



最佳实践 – 统一交付件

- 统一所有开发商的交付件
 - ▶ 统一交付件类型和模板及其质量标准
 - ▶ 使用统一的开发指南，统一交付件风格
- 有助于交付件的审核，提高产品质量
- 提高交付件的可理解性，增强团队沟通
- 增强系统的可维护性



最佳实践 – 监控项目过程

- 监控开发商项目开发的全过程
 - ▶ 了解开发商的项目开发计划并监控其执行
 - ▶ 确保项目开发按预定进度进行
- 多次分阶段验收，掌控项目进度，降低项目风险
- 提前收集业务部门反馈意见，保证产品质量



最佳实践 – 建立验收标准

- 建立软件验收标准体系
 - ▶ 统一规定交付件集合及其质量标准
 - ▶ 建立量化质量指标，保证验收的客观性和可操作性
- 把软件质量作为项目需求的一部分
- 有效执行验收标准
 - ▶ 该质量标准是可验证的(在有限的时间和代价下)
 - ▶ 在项目开发的各检验点(里程碑)确保验收标准的执行



最佳实践 – 管理版本和变更

- 开发商提交的所有交付件是企业的重要软件资产
- 必须对所有的交付件进行集中管理
- 控制管理对于交付件的变更
- 管理同一软件的不同定制版本
 - ▶ 针对不同地区的定制版本
 - ▶ 针对不同运行平台的运行版本



议程

- IT项目成功的关键
- 如何管理外包软件开发项目
- 软件项目外包管理最佳实践
- ➔ ■ IBM 软件开发平台



基于角色的软件开发平台



系统分析员

建模、模拟
组合、监视
业务流程



架构设计师

对应用和数
据进行可视
化建模



开发人员

快速构建
变换、集成
生成代码



测试人员

设计、创建
和执行测试
用例



部署人员

按需配置
实施调试
解决应用
系统问题



项目经理

- 遵循公共流程
- 管理和度量项目和项目组合
- 管理需求
- 管理变更和软件资产
- 管理软件质量



企业管理层

- 使 IT 投资与业务发展目标相适应
- 分析和监控项目和项目组合



IBM 软件开发平台



系统分析员

WBI
Modeler
& Monitor

Rational
Software
Modeler



架构设计师

Rational
Software
Architect



开发人员

Rational
Application
Developer

Rational
PurifyPlus



测试人员

Rational
Functional
& Manual
Tester

Rational
Performance
Tester



部署人员

Tivoli
Configuration
Manager

Tivoli
Monitoring

Customer
Extensions

ECLIPSE

3rd Party
ISV Tools



项目经理

(RUP, RequisitePro, TestManager, SoDA, ClearQuest, ClearCaseLT)

Rational Team Unifying Platform

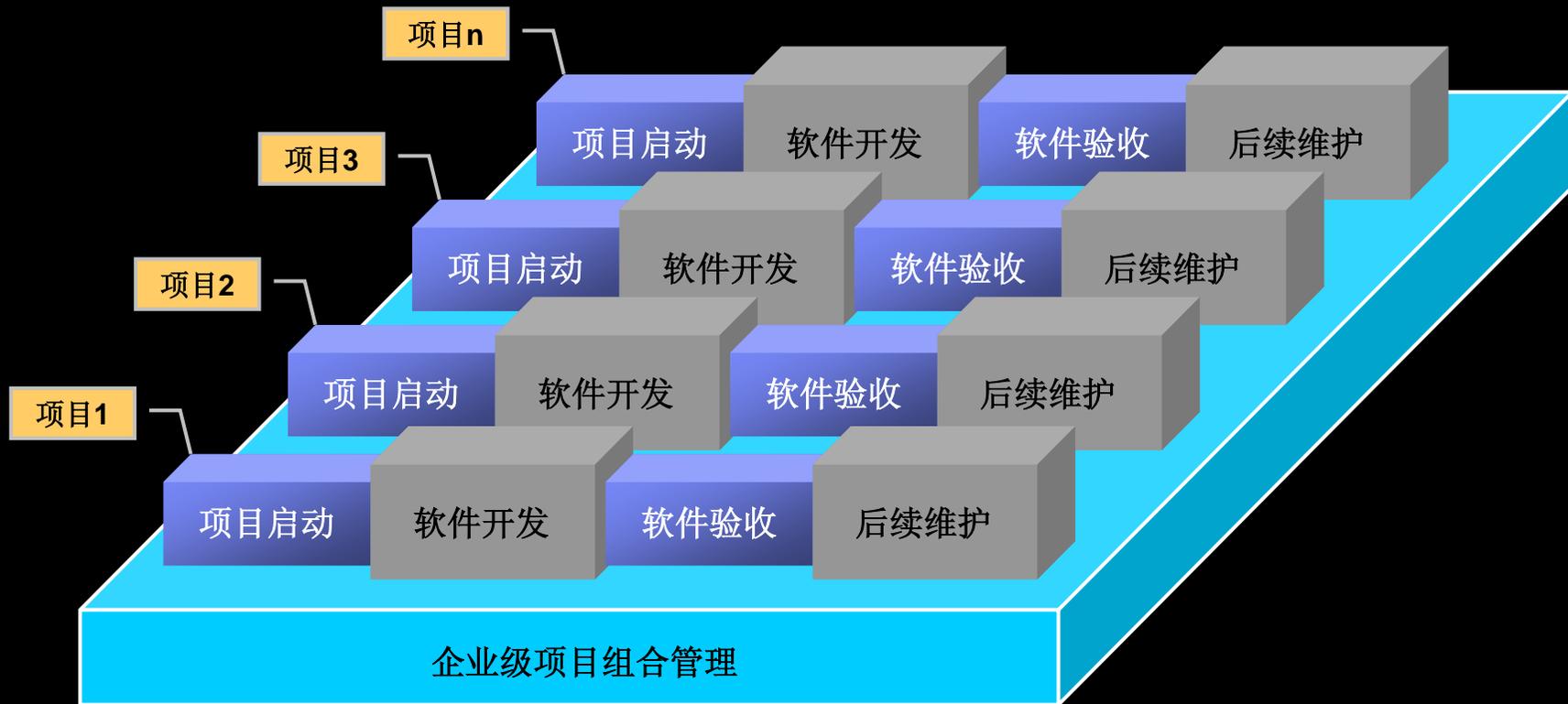


企业管理层

Rational Portfolio Manager



企业项目外包模型



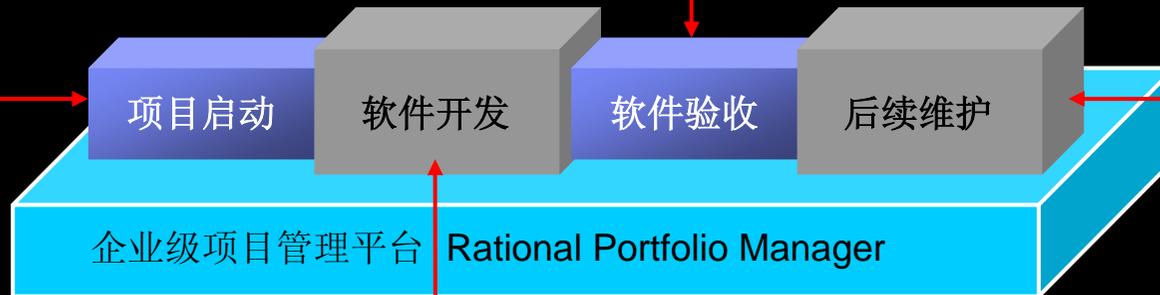
IBM 软件开发平台为外包管理提供全面支持

项目需求管理解决方案

- 明确软件需求，提高需求质量
- 需求管理工具 *RequisitePro*
- 可视化建模工具 *Rose/RSM*
- 需求变更管理工具 *ClearQuest*
- 文档自动生成工具 *SoDA*

软件验收管理解决方案

- 建立验收标准，保证软件质量
- 可靠性测试工具 *PurifyPlus*
- 自动化回归测试工具 *Functional Tester*
- 自动化性能测试工具 *Performance Tester*
- 缺陷跟踪及变更管理工具 *ClearQuest*
- 文档自动生成工具 *SoDA*



开发过程监控解决方案

- 监控项目过程，控制项目风险
- 需求管理工具 *RequisitePro*
- 系统构建工具 *Rose/XDE/RSA/RAD*
- 配置管理工具 *ClearCase*
- 变更管理工具 *ClearQuest*

软件维护开发解决方案

- 建立变更管理流程，全方位的管理
- 缺陷跟踪及变更管理工具 *ClearQuest*
- 配置管理工具 *ClearCase*
- 可视化建模工具 *Rose/XDE*
- 自动化回归测试工具 *Functional Tester*



IBM 软件开发平台: 客户的投资回报

Merrill Lynch

- 300% productivity increase

Ericsson

- 80% fewer bugs; 100% productivity increase

Lockheed Martin Canada

- \$409K Net Quantifiable Benefit; 222% ROI

Credence Systems Corporation

- 1200% increase in developer productivity
- 90% reduction in bug backlog

Covarity

- Reduced development lifecycle by 25%

Choice Hotels International

- 1440% ROI; 9.5M Net Quantifiable Benefit

Alltel

- 66% reduction of development cycle time
- 2 month manual test reduced to 2 days



THANK
YOU

