精益软件发布管理

乔梁

敏捷中国大会

ThoughtWorks® InfoQueue Enterprise Software Development Community

乔梁
ThoughtWorks资深咨询师
InfoQ敏捷社区特约编辑
YIM: qiaoliang_email
Gmail: sagittatius.qiao
http://blog.csdn.net/tony1130

什么是发布管理(Release Management)

"The definition, support and enforcement of processes for preparing software for deployment to production."

----- Forrester definition

Release Management: http://www.itlibrary.org/index.php?page=Release_Management

我们将讨论

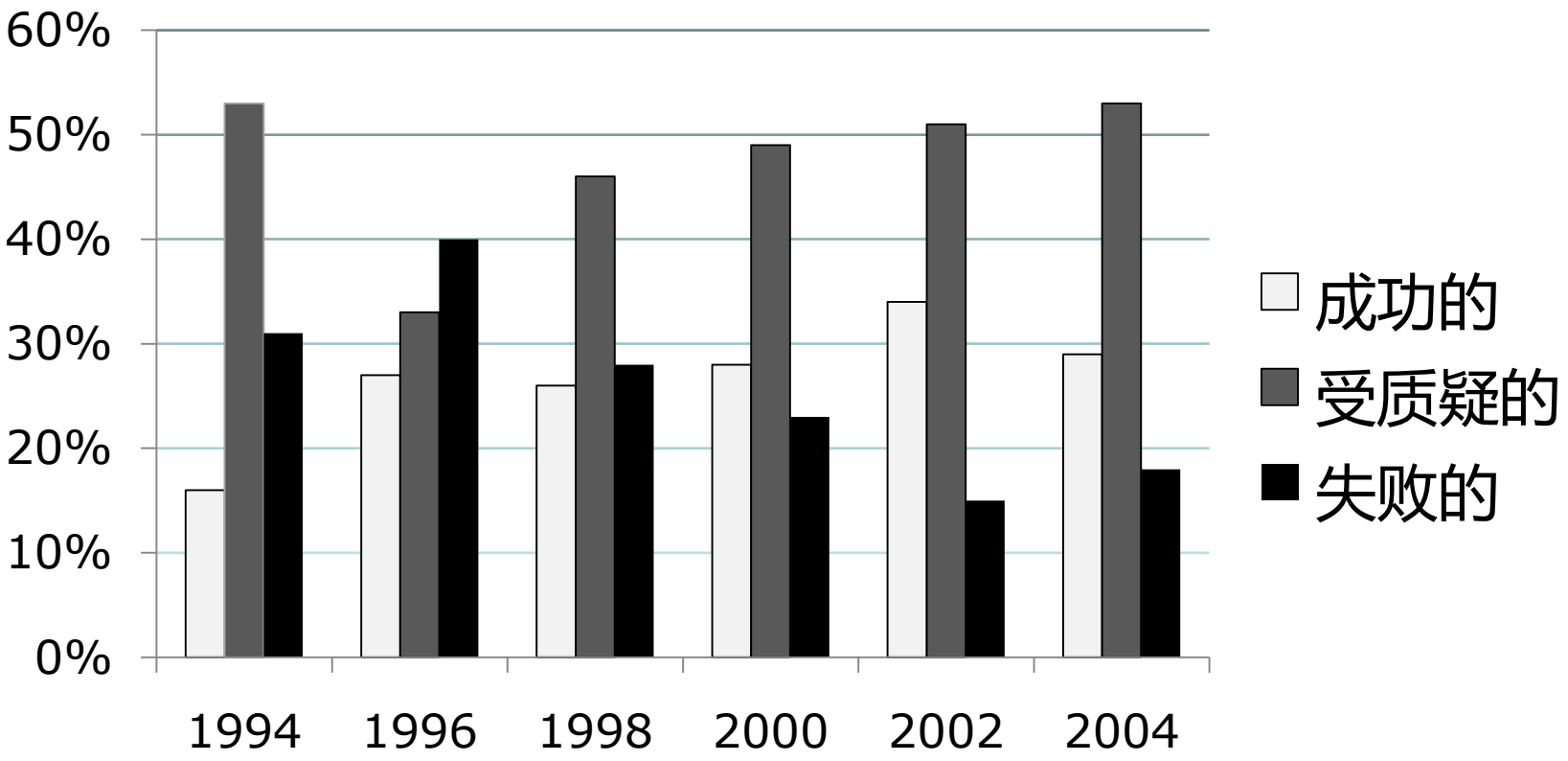
- 发布管理中遇到的问题
- 精益软件开发原则
- 发布管理实践

Out of scope :

- 发布管理的具体流程
  - ✓ 如何划分业务优先级
  - ✓ 定制发布计划
  - ✓ 发布编码标准
  - ✓ 软件发布记录管理
  - ✓ ……

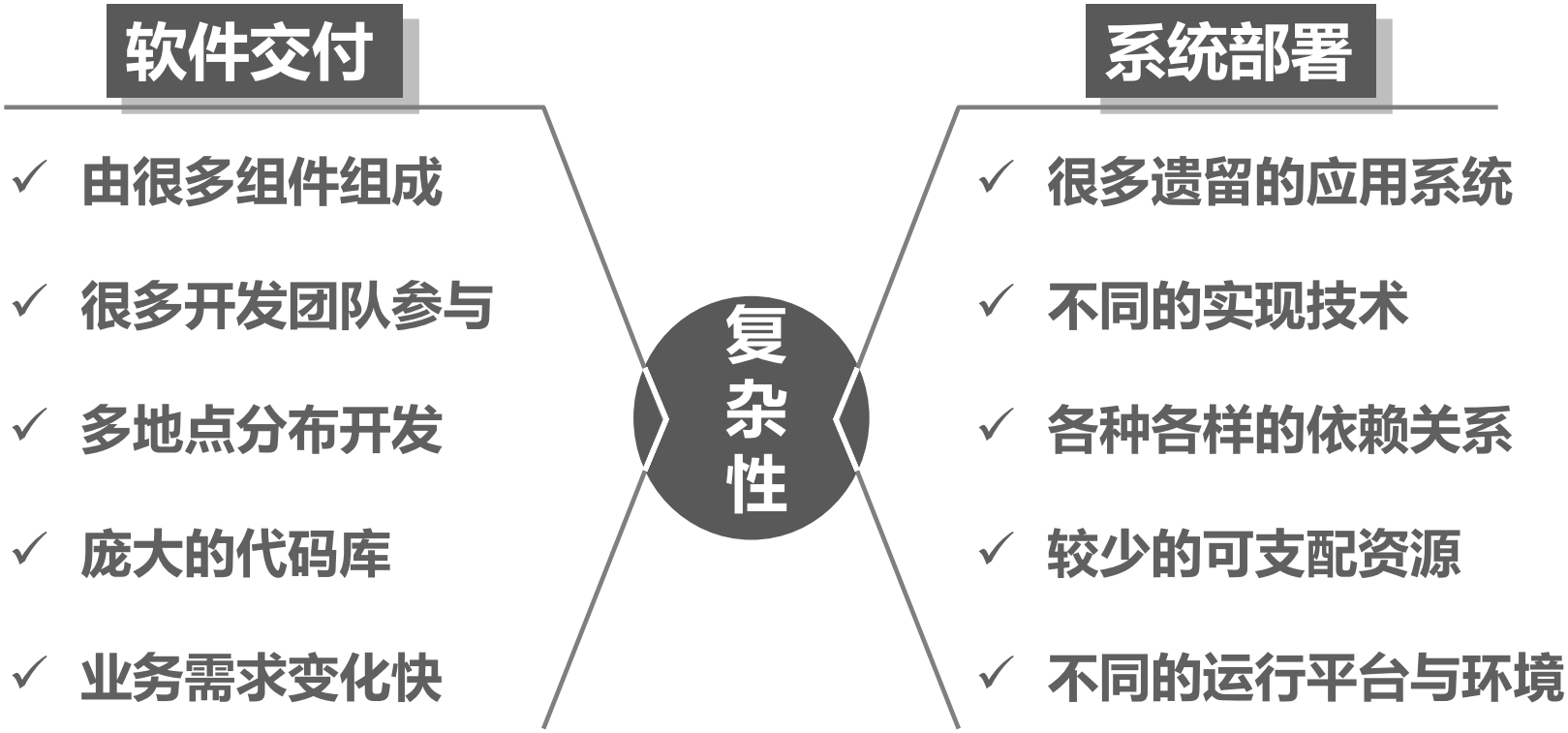
发布管理中遇到的问题

软件项目成功率很低（CHAOS ）



数据来源-：《The Art of Lean Software Development》 - - Curt Hibbs, Steve Jewett & Mike Sullivan

为什么呢


软件交付
由很多组件组成
很多开发团队参与
多地点分布开发
庞大的代码库
业务需求变化快
复杂性
系统部署
很多遗留的应用系统
不同的实现技术
各种各样的依赖关系
较少的可支配资源
不同的运行平台与环境

最后一英里

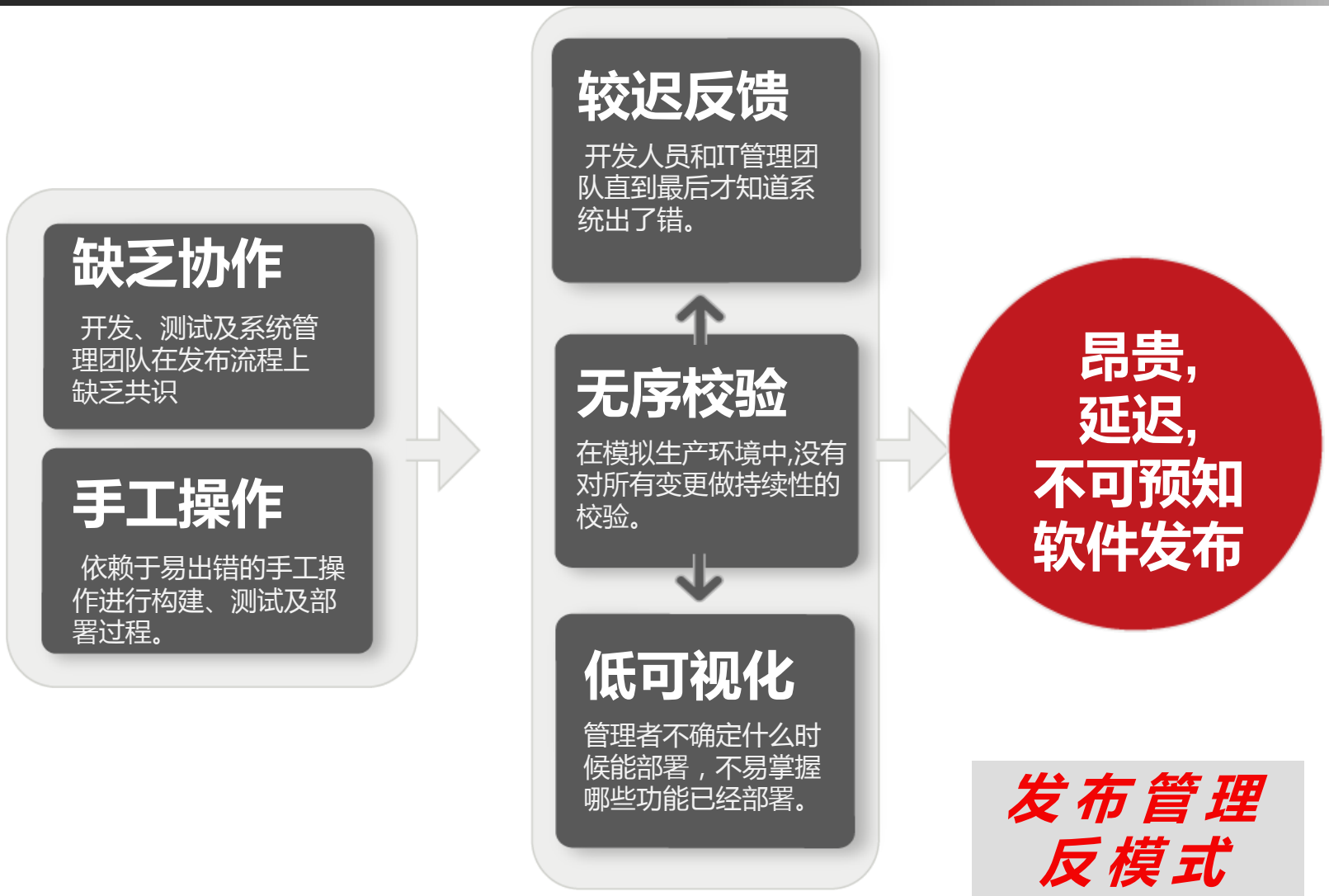
专职部署团队

超人特攻队

响应迟钝

结果……

哪里出了问题

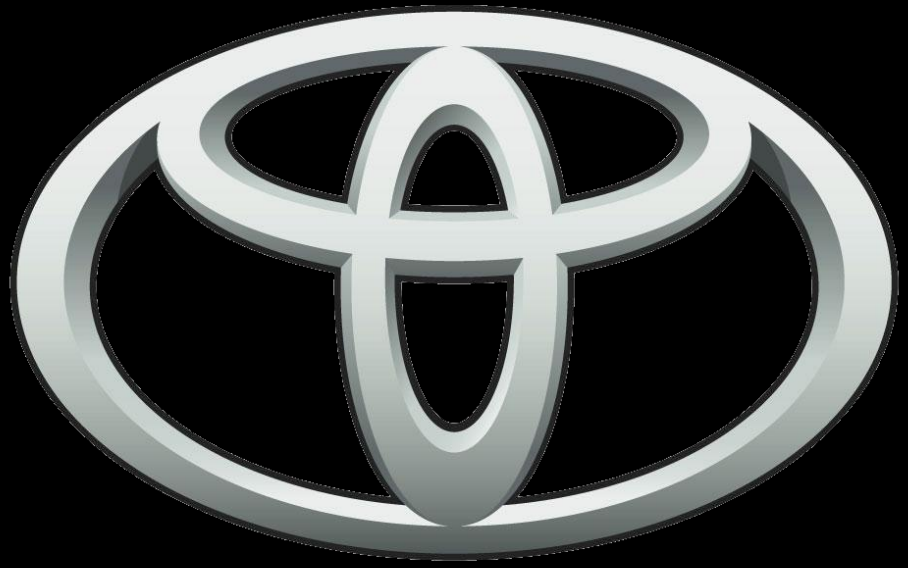

缺乏协作
开发、测试及系统管理团队在发布流程上缺乏共识
手工操作
依赖于易出错的手工操作进行构建、测试及部署过程。
较迟反馈
开发人员和IT管理团队直到最后才知道系统出了错。
无序校验
在模拟生产环境中,没有对所有变更做持续性的校验。
低可视化
管理者不确定什么时候能部署，不易掌握哪些功能已经部署。
昂贵，
延迟，
不可预知
软件发布
发布管理
反模式

开发 → 单元测试 → 功能测试

打包部署
测试环境

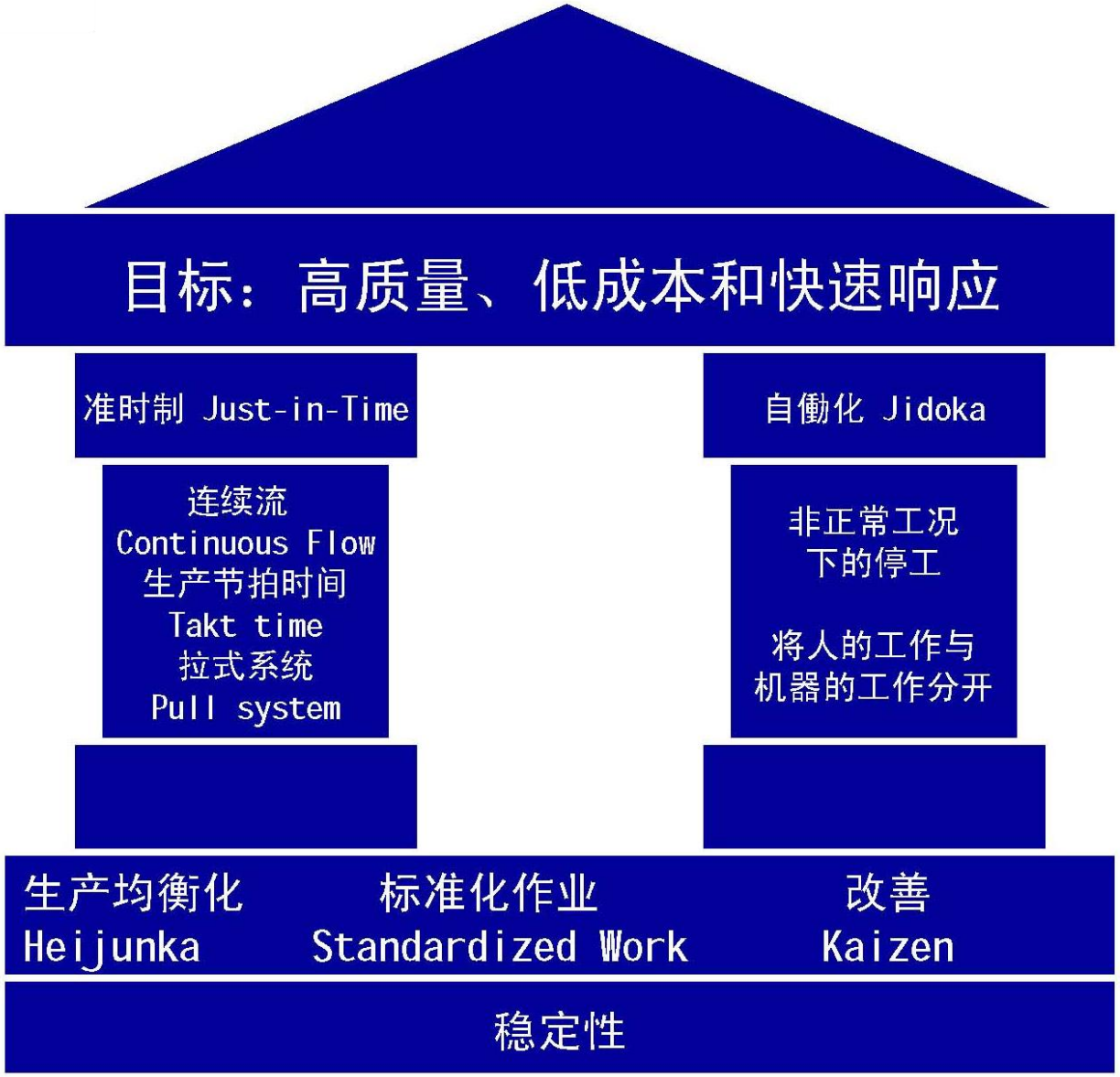
所有相关测试 → 部署试运行环境 → 部署生产环境

精益软件开发原则

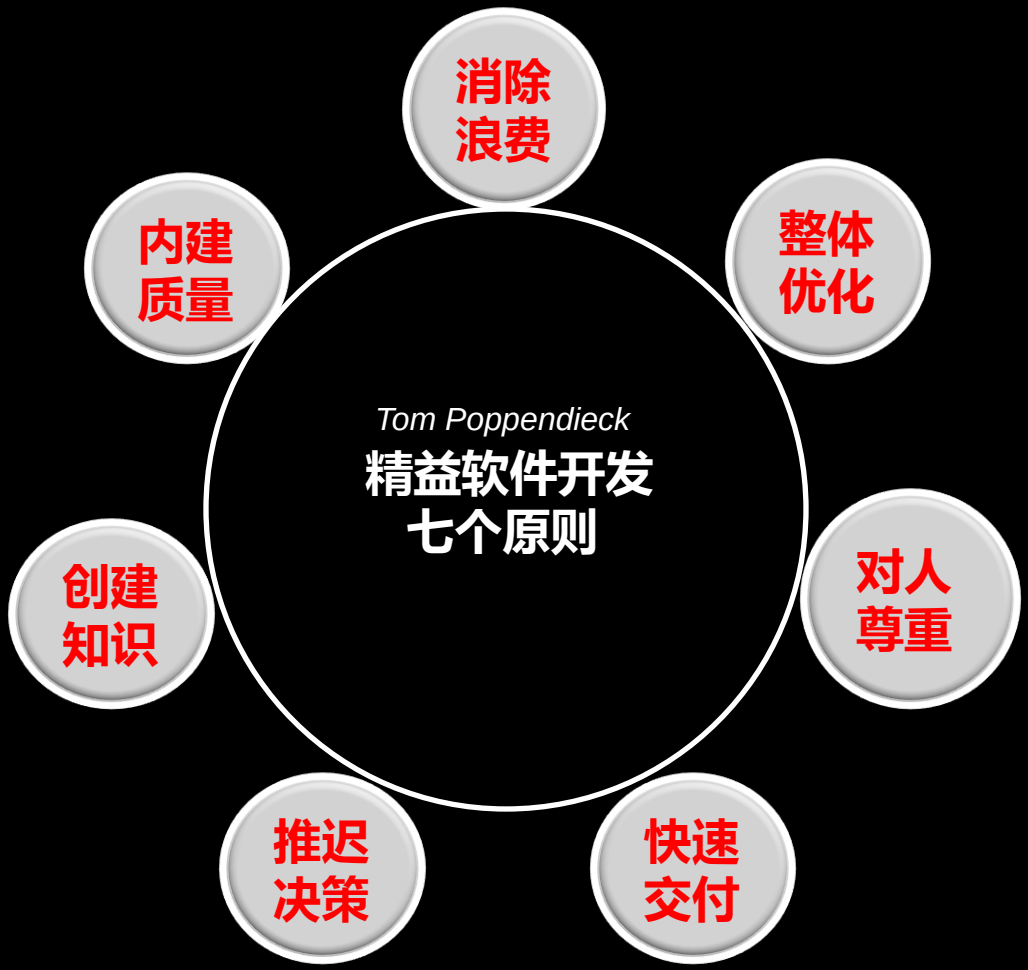


TOYOTA

Toyota Production System



软件开发



Taken from *Lean Software Development: An Agile Toolkit*, Tom Poppendieck (Addison Wesley, 2003)

精益发布管理实践

Mingle Cruise

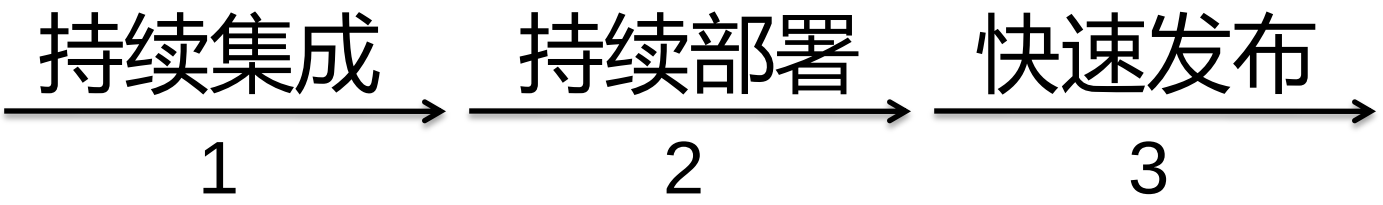
产品情况

	mingle Agile Project Management	cruise Release Management
用途	项目管理和团队协作	持续集成与发布管理
Web应用	√	√
支持 多种平台	√	√
支持 多种浏览器	√	√
支持 多种数据库	PostgreSQL、 MySQL、Oracle	内嵌
版本控制	Subversion, Perforce	Subversion, Perforce, Mercurial, Git,

团队情况

	mingle Agile Project Management	cruise Release Management
地理分布	北京,旧金山	北京,旧金山
人员组成	**20**人（**11 / 3 / 2**）	**11**人（**8 / 1 / 0.5**）
开发语言	RubyOnRails	JAVA
IDE	Textmate	IntellJ 8.0
版本控制	Mercurial	Mercurial
项目管理	Mingle	Mingle
集成部署	Cruise	Cruise
测试工具集	RUnit / Selenium	JUnit / Twist

软件产品发布实践



软件产品发布管理实践

第一部分

持续集成

摘　　要

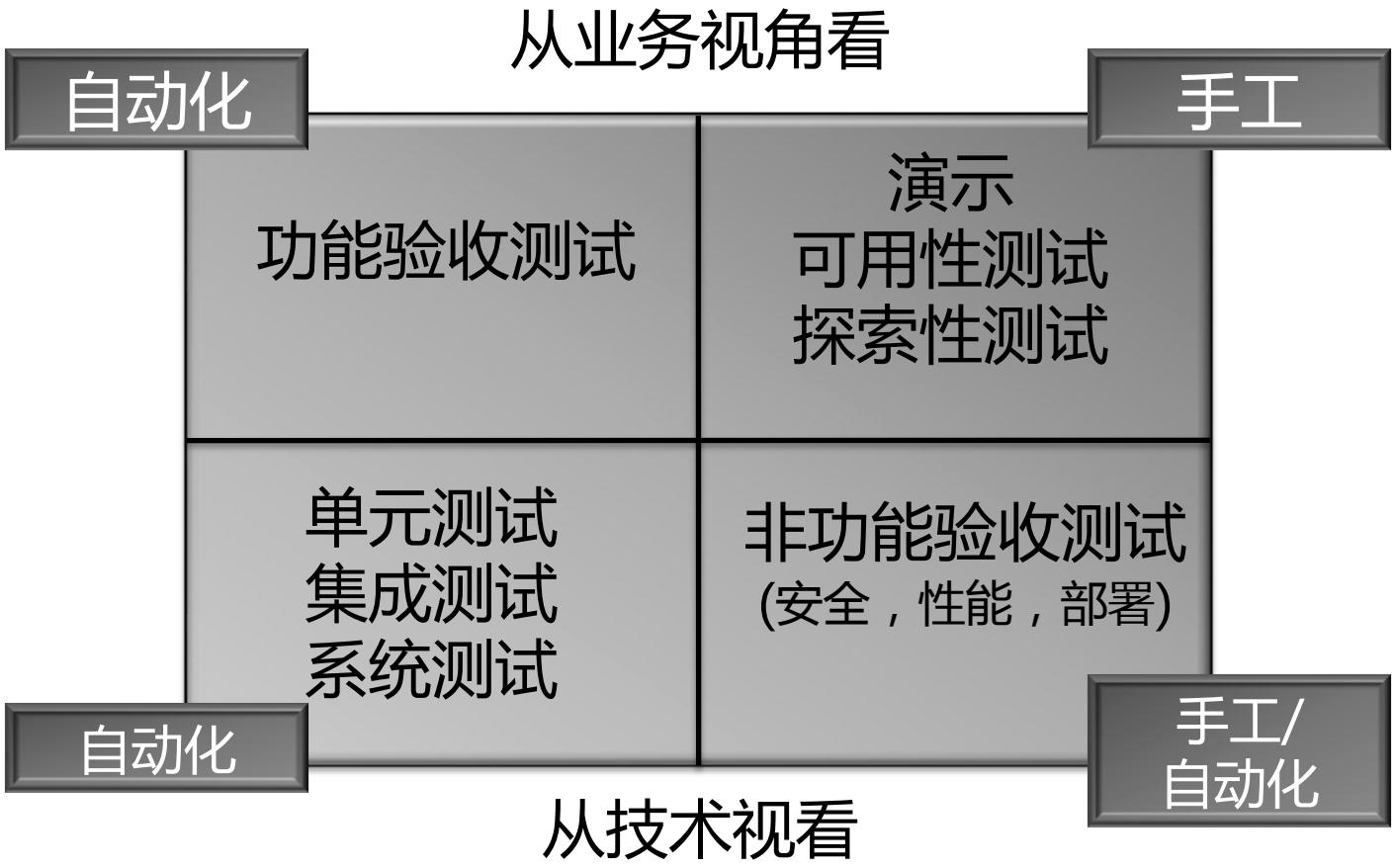
- ✓ 什么是持续集成
- ✓ 为什么做持续集成
- ✓ 相关实践
- ✓ 可能的问题和困难

什么是持续集成

- 是一个软件开发实践
- 软件开发团队成员频繁集成他们的工作成果(通常每人每天至少一次)
- 每次集成通过运行自动化构建与测试套件尽早发现集成问题.

----*Martin Fowler*

测试



From Brian Marick http://www.exampler.com/old-blog/2003/08/21/#agile-testing-project-1, via Poppendieck, *Implementing Lean Software Development* (Addison Wesley, 2006), ch. 8.

为什么要做持续集成

- 尽早发现问题
- 降低缺陷进入下一环节的机率
- 节约成本

持续集成相关实践

项目一开始就做

持续集成相关实践

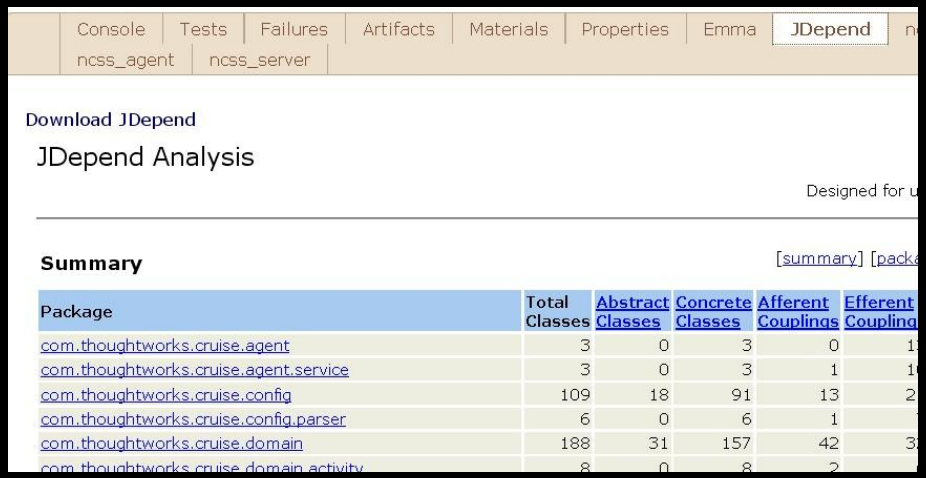
多角色协作

持续集成相关实践

慎用Mock技术

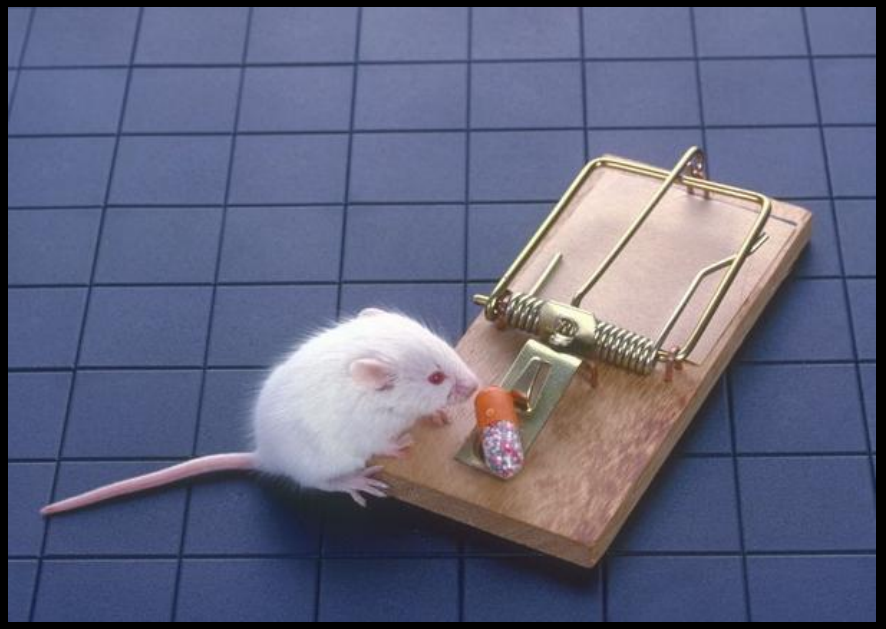
持续集成相关实践

自动收集有效度量



持续集成相关实践

“早死早托生”



持续集成相关实践

有问题时停止生产线

持续集成相关实践

手工测试一样重要

持续集成相关实践

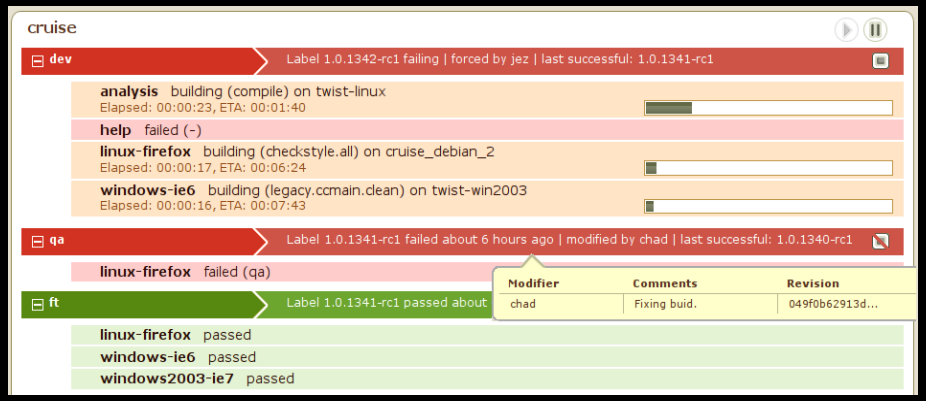
“干净”的集成环境

持续集成相关实践

尽可能多的集成环境

持续集成相关实践

及时且可视化



可能遇到的问题

- 测试代码也需要测试吗?
- 测试在本地运行太慢
- 集成耗时太长
- 持续失败/随机成功
- 大团队持续集成

软件产品发布管理实践

第二部分

持续部署

摘　　要

- ✓ 什么是持续部署
- ✓ 为什么做持续部署
- ✓ 相关实践
- ✓ 可能的问题和困难

什么是持续部署(Continuous Deployment)

- 软件发布管理中的一个实践
- 部署领域的持续集成
- 持续有规律地自动构建代码并将其部署到测试环境
- 通过一系列的测试后，选择适当的版本部署到预演环境中试运行
- 最后选择稳定的版本部署到生产环境

为什么要做持续部署

- 部署时，问题多多
- 采用“保守策略”后风险更大

一个手工部署过程

1. 登录到某个构建机器
2. 从版本控制系统下载最新的可用的版本
3. 打开构建机器上的IDE, 编译应用程序
4. 将得到的编译结果上传到生产环境中web服务器的某个备份文件夹中
5. 停止Web服务器
6. 在生产数据库上运行新的SQL升级脚本文件
7. 把编译后的所有文件拷贝到指定位置
8. 重新启动web服务器

手工部署后……

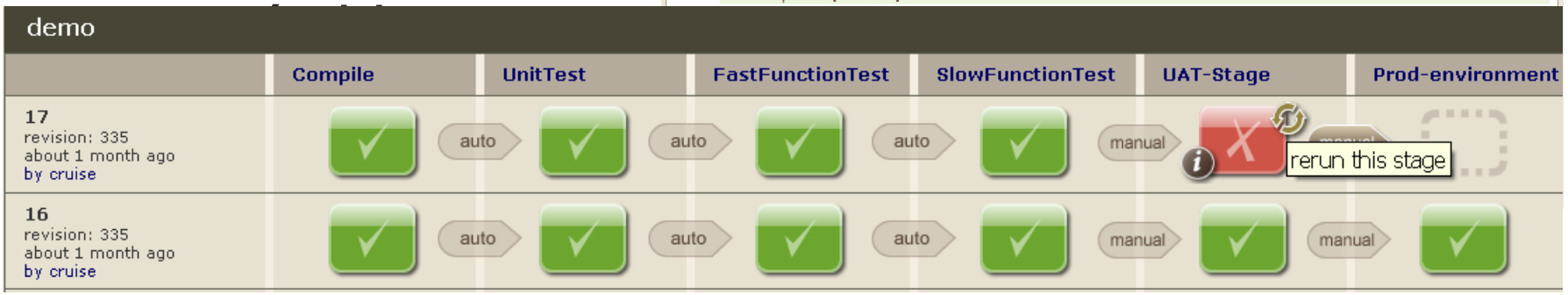
- GOD!真主啊! 我的神啊!保佑我吧！！”
- 咦，怎么服务没起来?
- 两小时后……
- 噢！测试中的配置文件怎么与生产环境中的不一样？是谁把配置文件的路径改啦！

Cruise配置

```
1. <job name= “Prod-deploy” >
2.     <tasks>
3.         <ant target= “deploy_to_production” />
4.         <ant target= “rollback” >
5.             <runif status= “failed” />
6.         </ant>
7.     <tasks>
8. </job>
```

持续部署相关实践

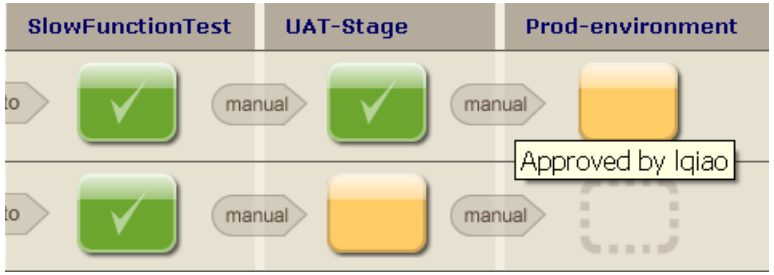
- 自动化
- 项目一开始就做



- 愈真实愈好
- 用版本工具对配置文件进行管理

可能的疑问和困难

- 某些步骤需要人工干预（？？）
- 自动部署失败,怎么办?
- 资源太少?



自动化全过程


CCTray Feed | Server Details | Help
cruise
CURRENT ACTIVITY
PIPELINES
AGENTS
ADMINISTRATION
Pipelines >
Pipelines
cruise Configuration
auto
UnitTest
auto
package
auto
FunctionalTest
auto
installers
manual
UAT-deploy
manual
Prod-deploy
manual
Publish

软件产品发布管理实践

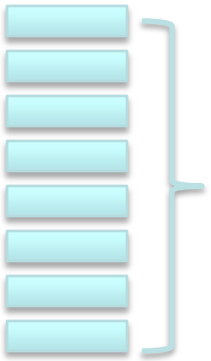
第三部分

摘　　要

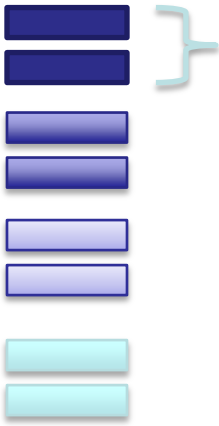
快速发布

- ✓ 快速发布对客户价值产生的影响
- ✓ 快速发布对承包方的益处
- ✓ 快速发布相关实践

不同发布频率的对比



8组需求 X 10天/组 = 80天 = 4个月 一次发布



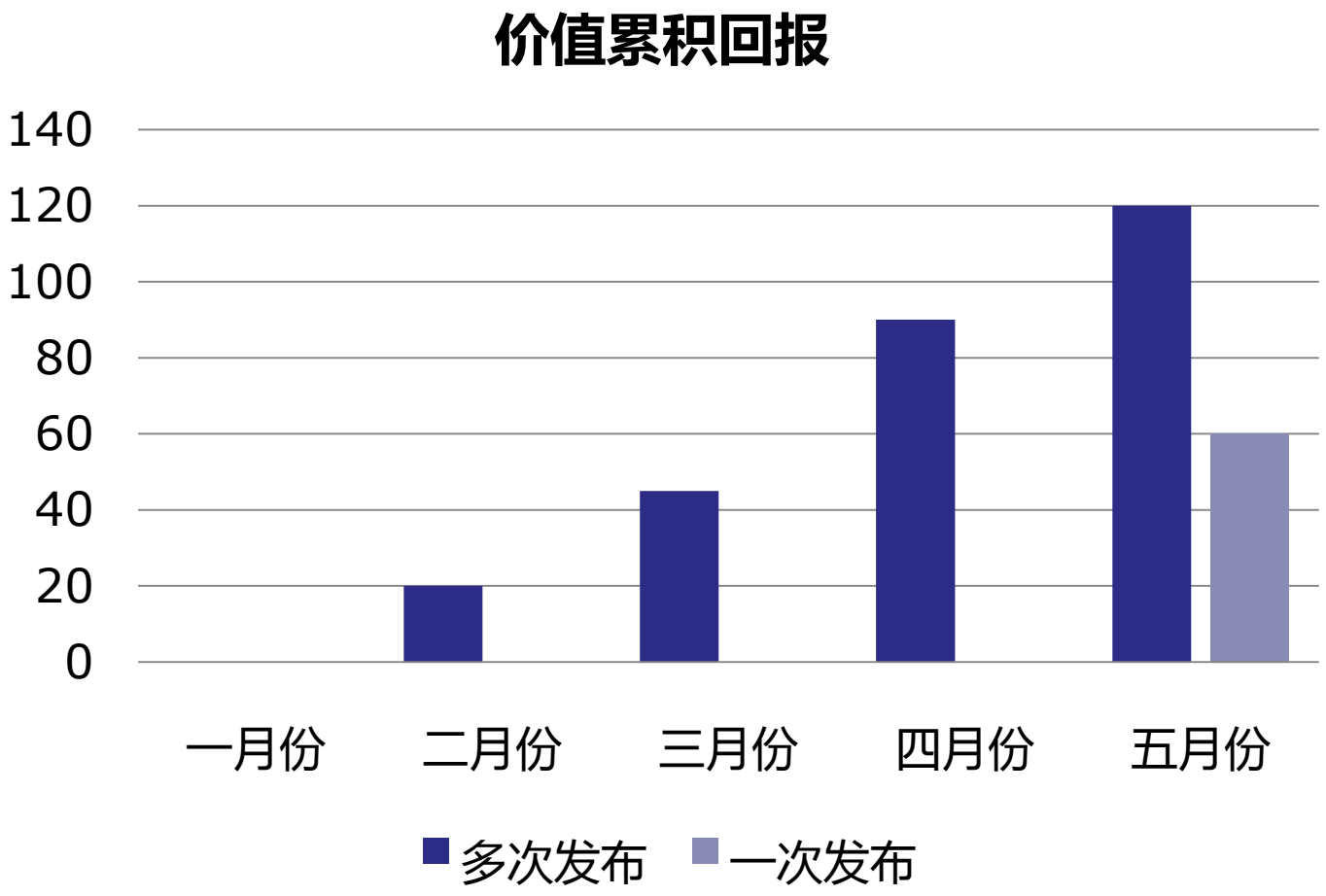
2组需求 X 10天/组 = 20天 = 1个月 第一次发布

= 20天 第二次发布

= 20天 第三次发布

= 20天 第四次发布

不同发布频率的对比

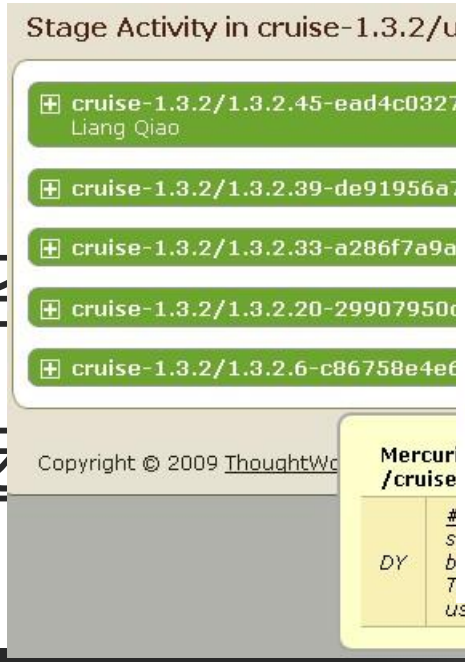


快速发布——收益

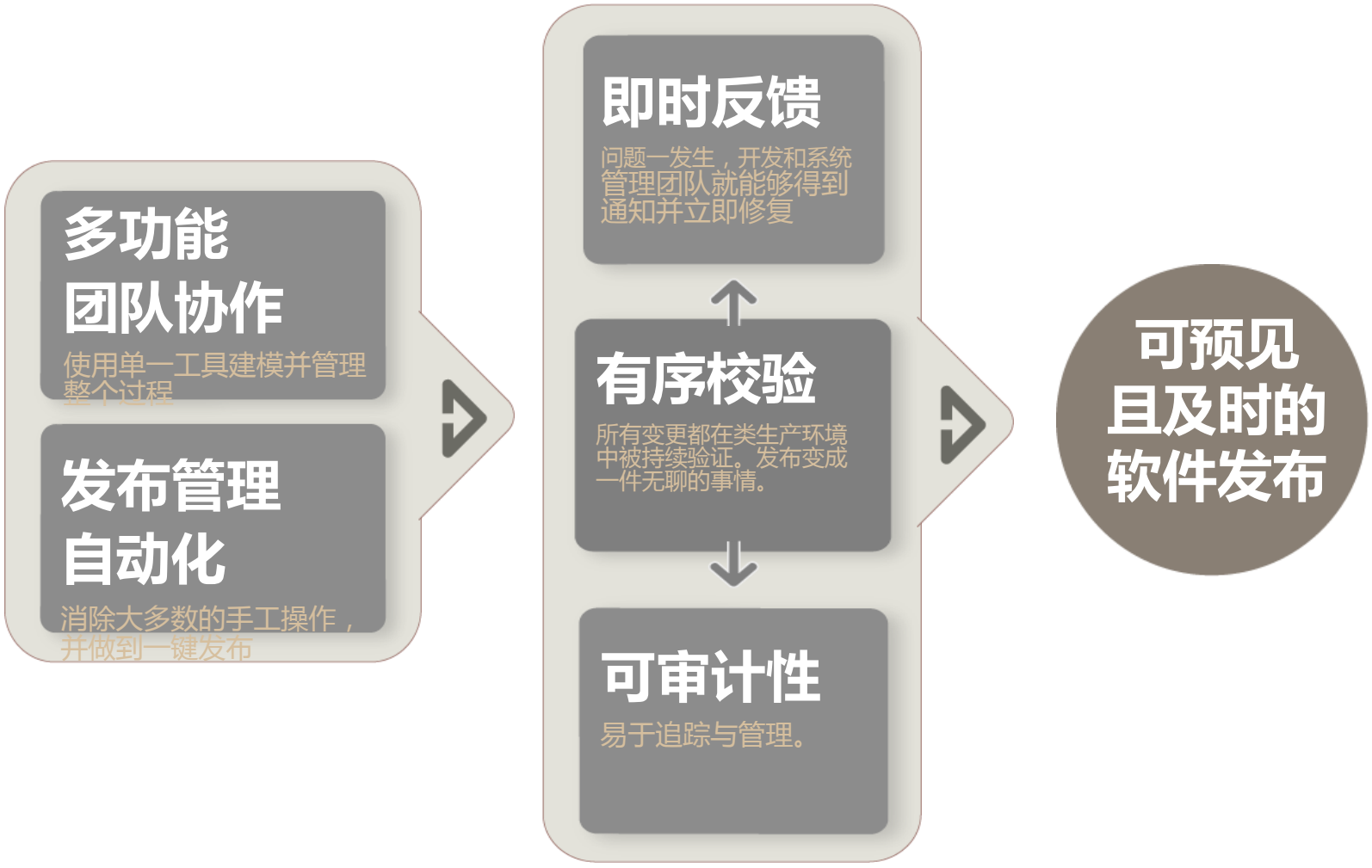
- 发布成功机会增加
- 发布越早，成本越小
- 一旦失败，回滚成本小
- 容易排定业务需求优先级,制定发布计划

发布实践

- “原子功能”提交,频繁提交
- 短迭代,小发布
- 易审计跟踪
- 不要走“捷径
- 发布管理是团



精益软件发布模式


多功能
团队协作
使用单一工具建模并管理整个过程
发布管理
自动化
消除大多数的手工操作，并做到一键发布
即时反馈
问题一发生，开发和系统管理团队就能够得到通知并立即修复
有序校验
所有变更都在类生产环境中被持续验证。发布变成一件无聊的事情。
可审计性
易于追踪与管理。
可预见
且及时的
软件发布

- 可预见且及时地发布经过全面测试的软件
- 通过有效利用硬件和人员时间来节省成本
- 交付高质量的软件：通过更好地测试，并遵循编码标准
- 各功能团队更好的协作
- 安全性
- 可审计性

Lean

Useful Links

http://blog.csdn.net/tony1130
http://studios.thoughtworks.com
http://www.infoq.com/
http://martinfowler.com/
http://en.wikipedia.org/wiki/Toyota_Production_System
http://en.wikipedia.org/wiki/Lean_software_development

Q&A

乔梁
YIM: qiaoliang_email
Gmail: sagittatius.qiao
http://blog.csdn.net/tony1130

开发情况

	mingle Agile Project Management	cruise Release Management
单元测试	5000个	5000个
功能测试	5000个	150个
集成环境	个人(1)/Team(60)	个人(4)/Team(20)
提交频率	——	平均 9次/天
构建频率	平均 20次/天	平均12次/天
部署频率	___ / 1周 / 1月	<1天 / <1周 / 1周
发布频率	<3个月(自2008年)	<3个月

Maximize your business-responsiveness with Mingle.
Provide your global development team a shared space that adapts to the way they work.

Build and evolve long-lasting test suites with Twist.
Ensure your team delivers intended, tested and complete business value.

Make release anxiety history with Cruise.
Gain control and visibility from ‘commit’ to production.

For true agility you need both

Management and Engineering practices

mingle™ + twist™ + cruise™ = **Truly Agile**

Tell me more

冒烟测试环境　　模拟生产环境　　真实