



北京航空航天大学
BEIHANG UNIVERSITY

软件过程管理讲座

第二讲：质量管理建模

吴超英

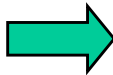
2016年10月

版权所有 请勿翻印



北京航空航天大学
BEIHANG UNIVERSITY

主题

- 
- 质量改进基本原则
 - 一个量化质量改进建模方法
 - 企业实施过程改进实例分享



1. 质量改进的基本原则

- 缺陷在程序中的时间越长，越难被发现
- 缺陷在程序中的时间越长，修复时间花费偏差就越大
- 通过度量建立量化模型可以提高对过程的计划和控制

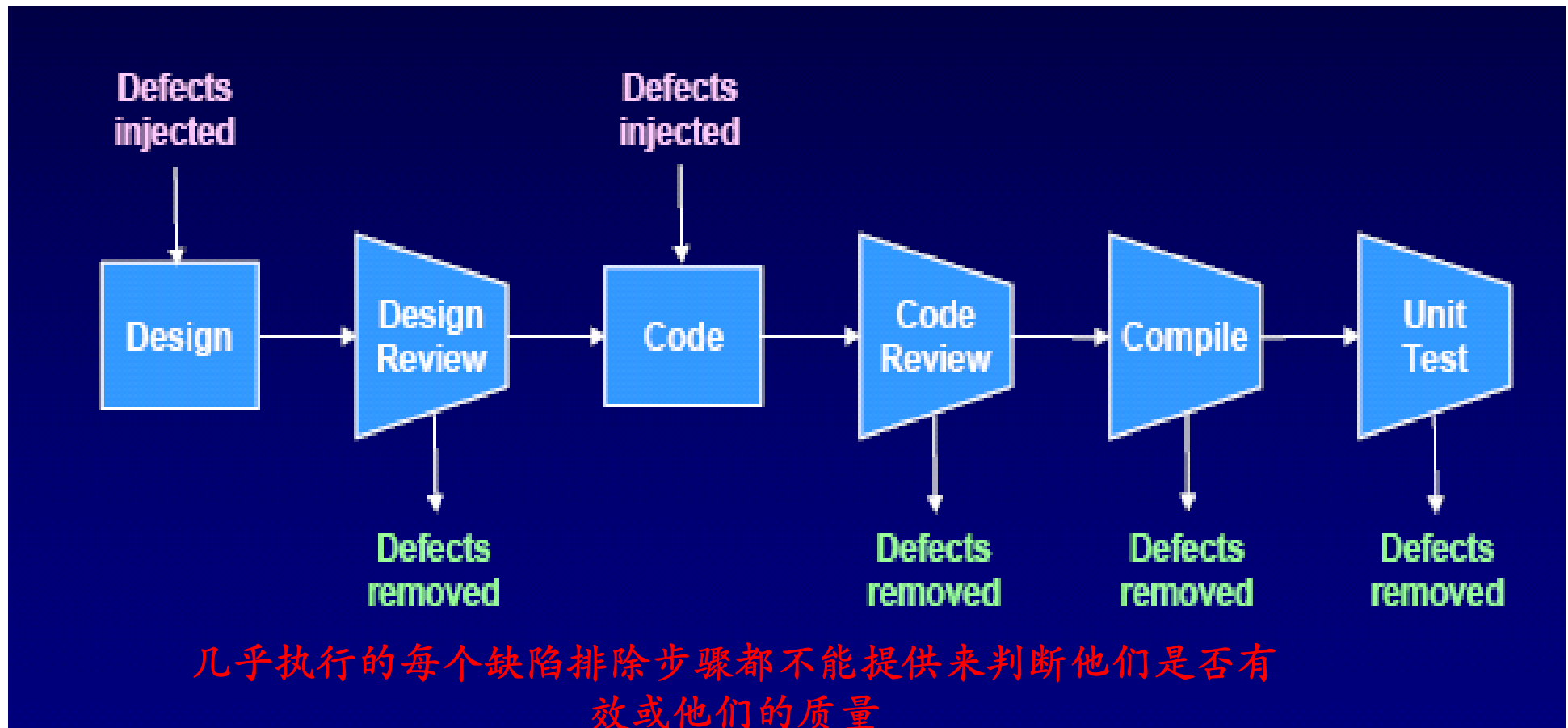


缺陷与质量的度量

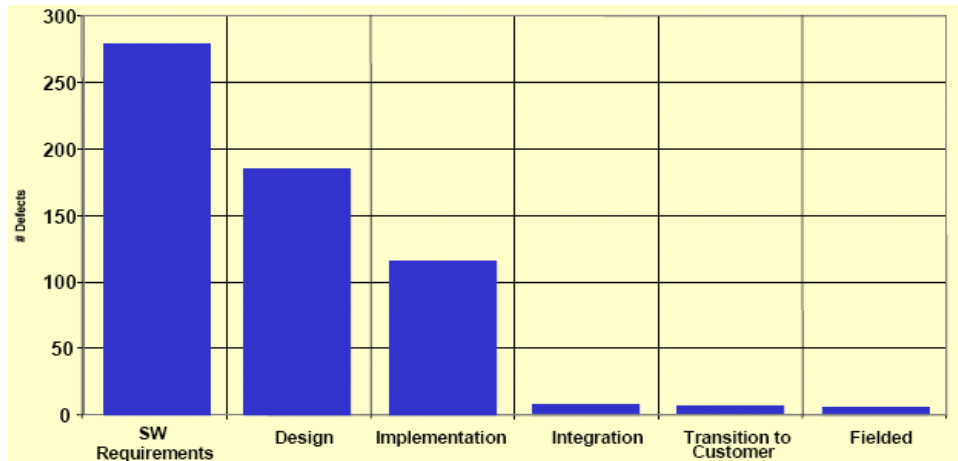
- 目的：了解质量活动效率，从而为提高效率
- **PSP** 有很多有用的质量和过程控制度量
 - 收益（ **yield** ）-即有效性
 - 评审（复查）速率
 - 每规模单位发现的缺陷数
 - 每小时注入和排除的缺陷数
 - 缺陷移动平衡（ **defect-removal leverage** ）
 - 质量成本（**COQ**）



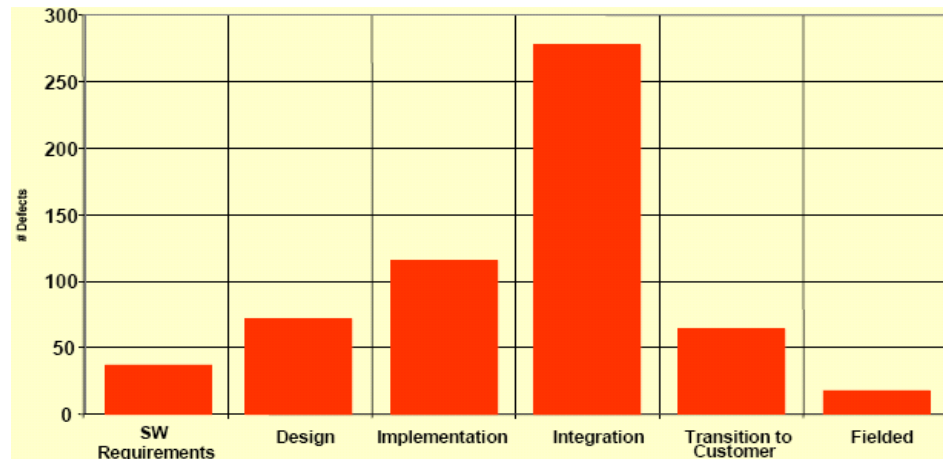
注入和排除缺陷



建立各阶段缺陷引入与诊断模型-示例

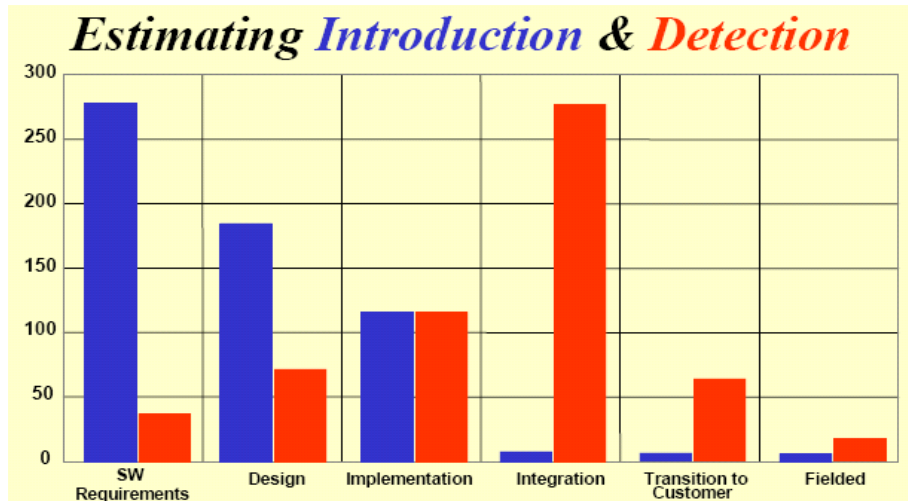


< 缺陷引入模式

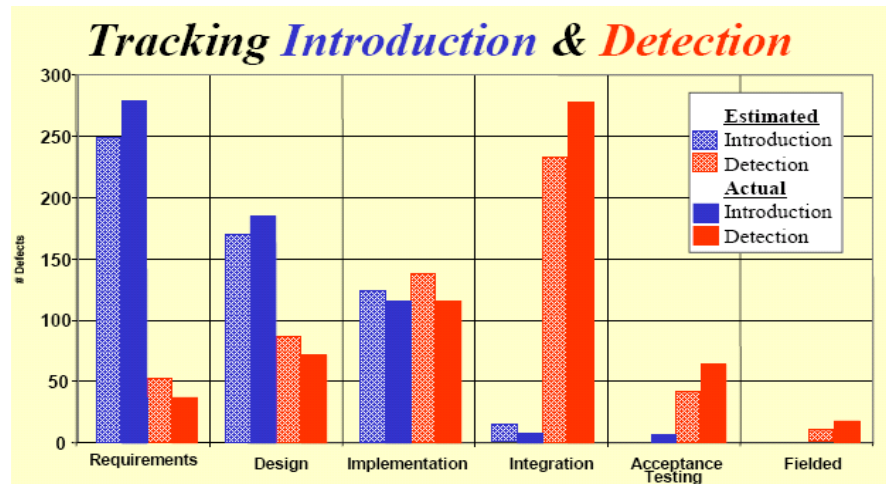


< 缺陷诊断模式

估计与跟踪



< 估计引入和诊断出缺陷数量



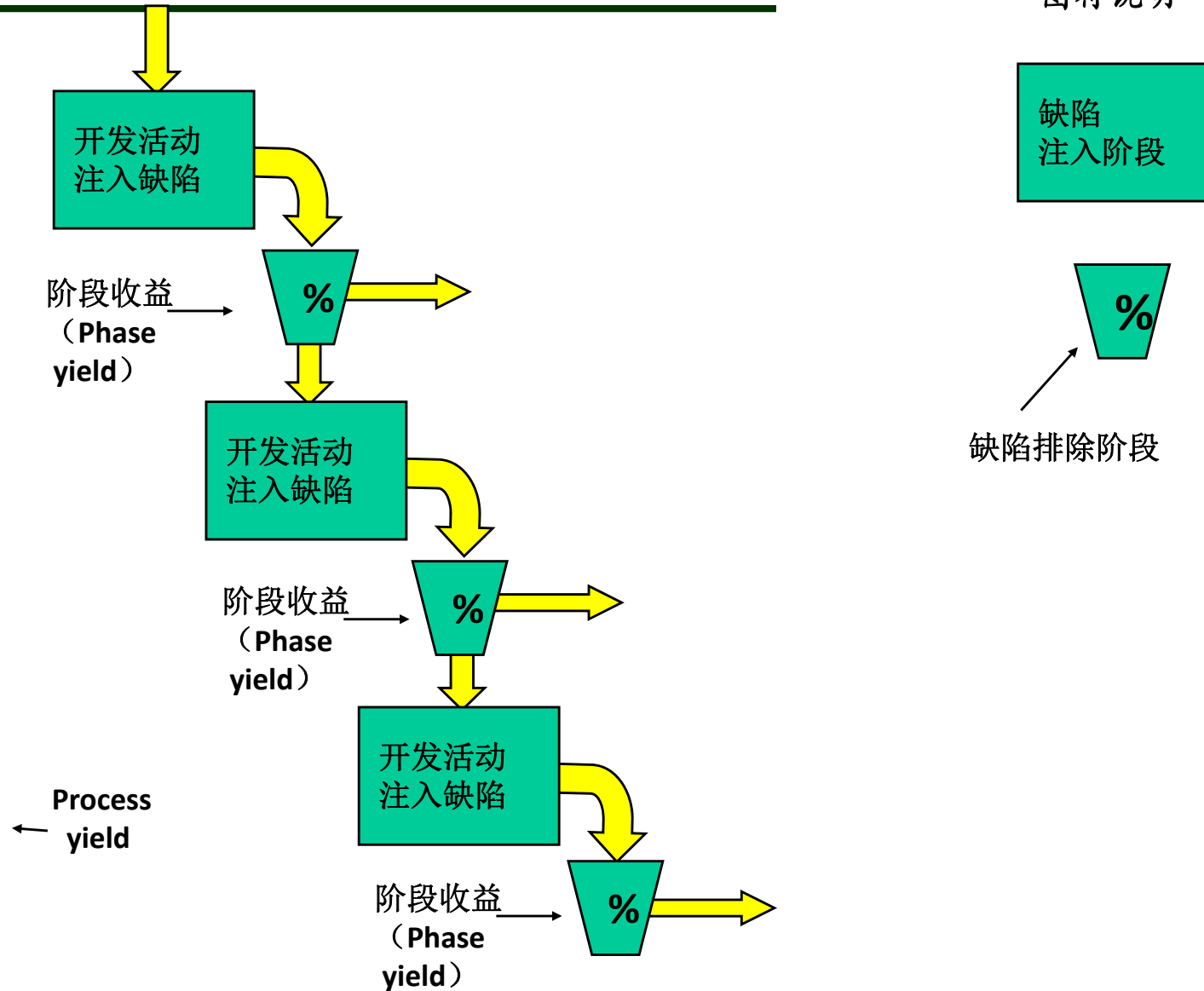
< 跟踪引入与诊断出的缺陷

图中数字是假设的

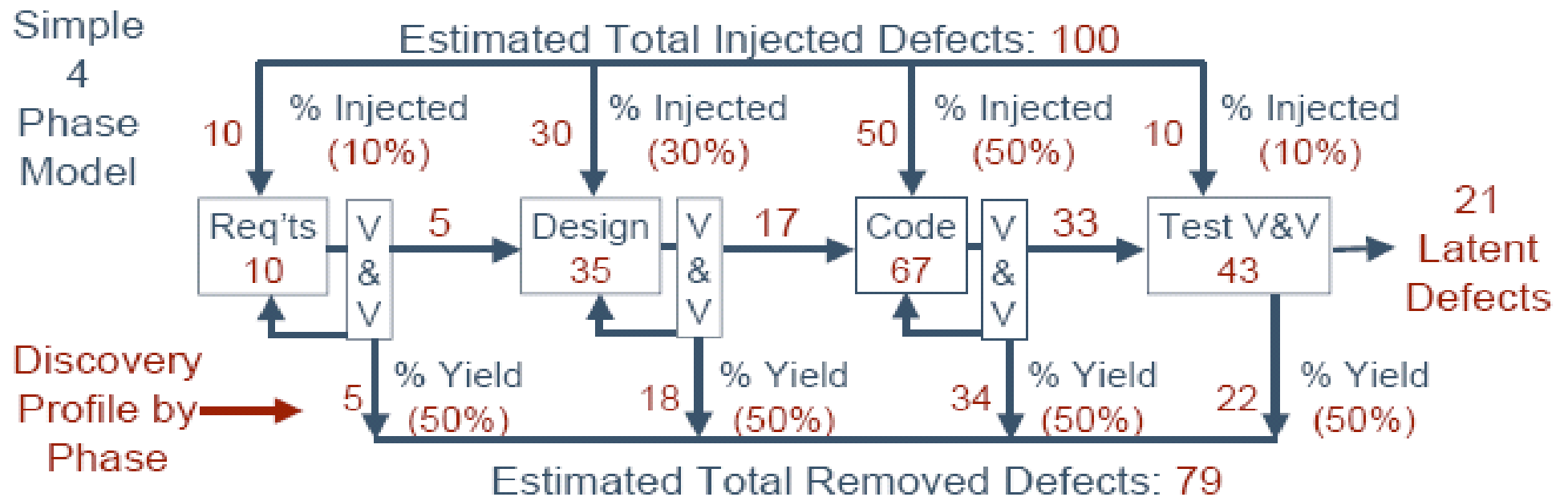


缺陷排除过滤器

图符说明



计划全生命周期缺陷的修复一示例



阶段收益（Phase Yield）

- 阶段收益度量关注
 - 在**该阶段**发现的产品缺陷的百分比
 - 对于该阶段过程（如代码评审）步骤的缺陷排除的效果
- 收益可以用来度量设计、代码评审、检查、编译和测试的效果

收益（对于每一个阶段）=

$100 * (\text{已发现的缺陷}) / (\text{已发现的缺陷} + \text{没有发现的缺陷})$



过程收益 (Process Yield)

- 过程收益是在一个阶段前缺陷注入的百分比，包括在该阶段前发现的所有缺陷
- 举例
 - 项目的过程收益指交付前 所有阶段发现的缺陷的百分比



课堂互动练习一—计算缺陷排除收益

- 从一个公司的历史数据得到，开发代码的平均缺陷率为**5defects/KLOC**，并给出了各测试活动的缺陷发现数。请对该公司已经开发的一个**100KLOC**的程序进行缺陷的预测，计算出每个阶段期望修复的缺陷数；
- Yield (for a phase) = 100 * (defects found) / (defects found + not found)**

开发注入的总缺陷＝			
阶段	发现的缺陷	遗漏数	Yield(阶段收益)%
代码评审（复查）	140		
编译	150		
单元测试	80		
测试	110		
过程收益＝			



最终的缺陷排除收益的估计

- 如何确定有多少缺陷残留在最终产品中？
 - 跟踪产品整个使用生命期间发现的缺陷数。但即使经过若干年的使用也不能发现所有的缺陷，所以永远无法知道确切数值。
 - 如果在缺陷排除的各个阶段发现的缺陷数迅速下降，则所求得的缺陷排除收益值可能相当准确。
 - 在收集缺陷数据之后，就可以求出每个阶段的缺陷排除收益的测量值，然后就能估算可能残留在产品中的缺陷数目。
- 一个有用的拇指规则是
 - 假设残留的缺陷数和最后一个阶段排除的缺陷数相等。这等于假设最后阶段的缺陷排除收益是**50%**。根据**Watts**的经验数据，这个值对认真进行了同行互查和代码评审（复查）的有点低，对编译阶段大约正好，对大多数测试阶段有点偏高。
 - 根据最后一个阶段的缺陷排除收益值计算出可能漏过的缺陷值



质量成本（COQ）-1

- 质量成本(COQ) 包括三个主要元素：过失成本、质检成本和预防成本。
- **过失成本**包括修复产品中缺陷的所有费用。当修复一个缺陷时，就增加过失成本。类似地，当运行调试器来查找有缺陷的语句时，也会增加过失成本。任何与修复缺陷有关的工作都记入过失成本，这甚至包括重新设计、重新编译或者重新测试。
- **质检成本**包括评估产品以确定是否有缺陷的所有工作，但不包括修复缺陷的时间花费。这包括对无缺陷产品的代码评审（复查）时间、编译和测试时间。因此，质检成本不包括修复缺陷的费用。质检成本是确保产品无缺陷的保障费用。
- **预防成本**是标识和解决缺陷原因的费用。例如，包括对理解缺陷所做的分析以及改进需求、设计或编码过程的过程开发工作。重新设计和测试一个新的过程所花费的时间也属于预防花费。
- 其他缺陷预防成本包括
 - 正式规格和设计工作
 - 原型建造
 - 过程分析和改进
 - 缺陷记录



质量成本（COQ）-2

- 质量成本**COQ**是以一种对管理有意义的过程质量度量方式
- 过失成本：在**PSP**中，是指编译和测试时间
- 质检成本：在**PSP**中，是指设计和代码评审时间
- 在**TSP**中，审查（**inspections**）包括在质检成本中



质检/过失比率

- 质检/过失比(**Appraisal/Failure Ratio, A/FR**)

是一种有用的度量方法，其值等于质检成本与过失成本之比。一个更简单的计算**A/FR**的方法，是用评审（复查）时间除以编译和测试时间，用以度量在第一次编译前花在查找缺陷上的时间的相对值。

- 经验表明，**A/FR**的值能很好地指示测试中发现缺陷的可能性。当**A/FR**小于 1 时，测试一般能发现很多错误；而当**A/FR**大于 2 时，每千行源码都只有很少几个缺陷。这说明，**A/FR**大于 2 的过程比**A/FR**小于 1 的过程生产无缺陷产品的可能性更大。因此，应该使**A/FR**的值大于 2。



一个质量准则（团队软件过程）

- ✓ 软件设计时间不小于软件实现时间
- ✓ 设计评审时间至少应占一半的设计时间
- ✓ 代码评审时间至少应占一半的代码编制时间
- ✓ 在编译阶段发现的缺陷不应超过 **10 个/KLOC**
- ✓ 在测试阶段发现的缺陷不应超过 **5 个/KLOC**。

TSP侧重于开发项目的组织与协调，并通过采集必要的的数据，使项目组在进入集成和系统测试之前，就能够通过“模块质量剖面图”分析模块的质量。



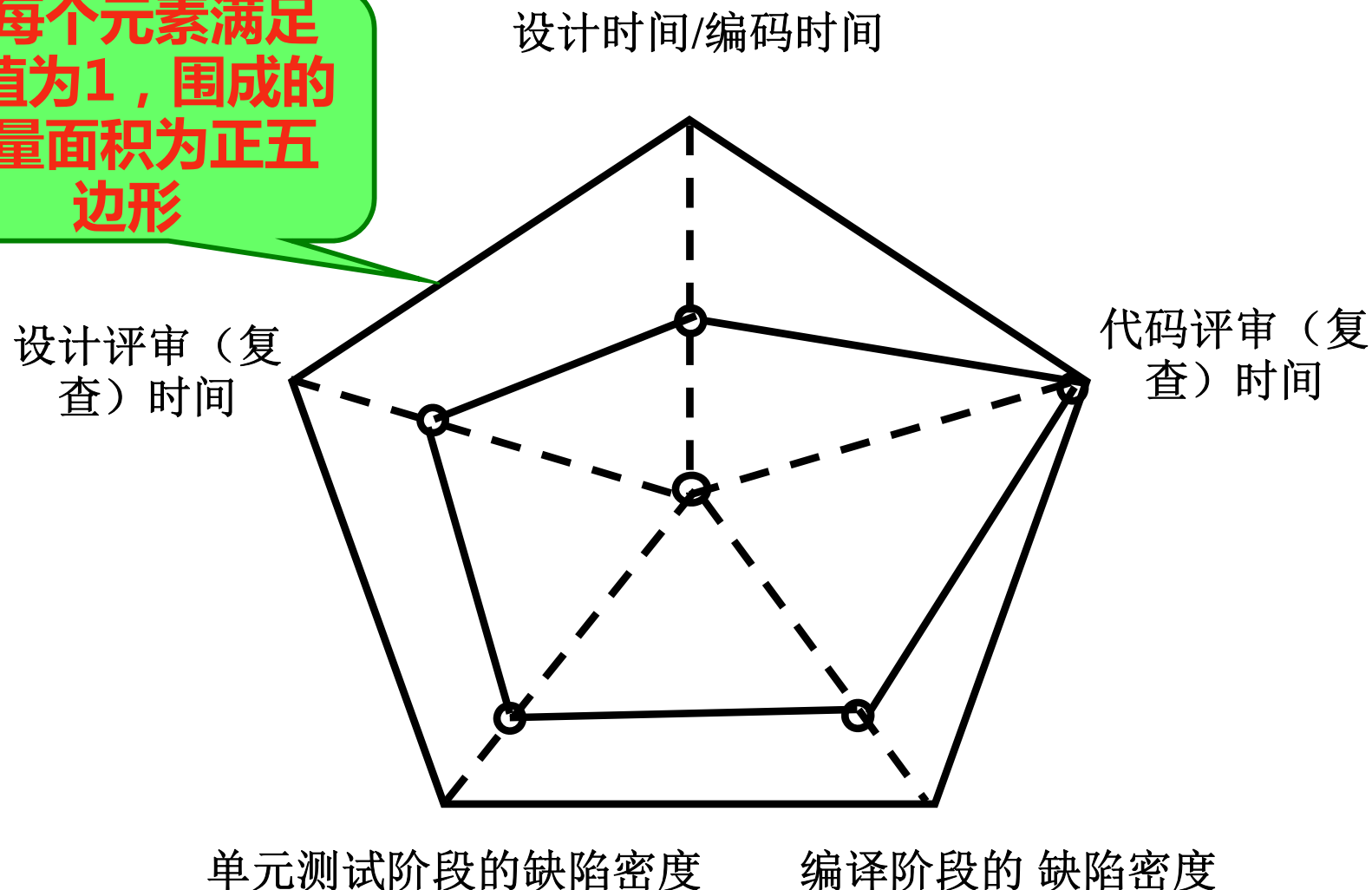
过程质量指数（PQI）

- 设计/编码时间=设计时间/编码时间，范围是**0.0~1.0**
- 设计评审（复查）时间=**2*设计评审（复查）时间/设计时间**，范围是**0.0~1.0**
- 代码评审（复查）时间=**2*代码评审（复查）时间/编码时间**，范围是**0.0~1.0**
- 编译缺陷数/**KLOC=20/（10+编译缺陷数/KLOC）**，范围是**0.0~1.0**
- 单元测试缺陷数/**KLOC=10/（5+单元测试缺陷数/KLOC）**，范围是**0.0~1.0**
- 过程质量指数可通过将这**5**个值相乘得到。然而，目标应该是**1.0**。



模块质量剖面图

当每个元素满足
时值为1，围成的
质量面积为正五
边形

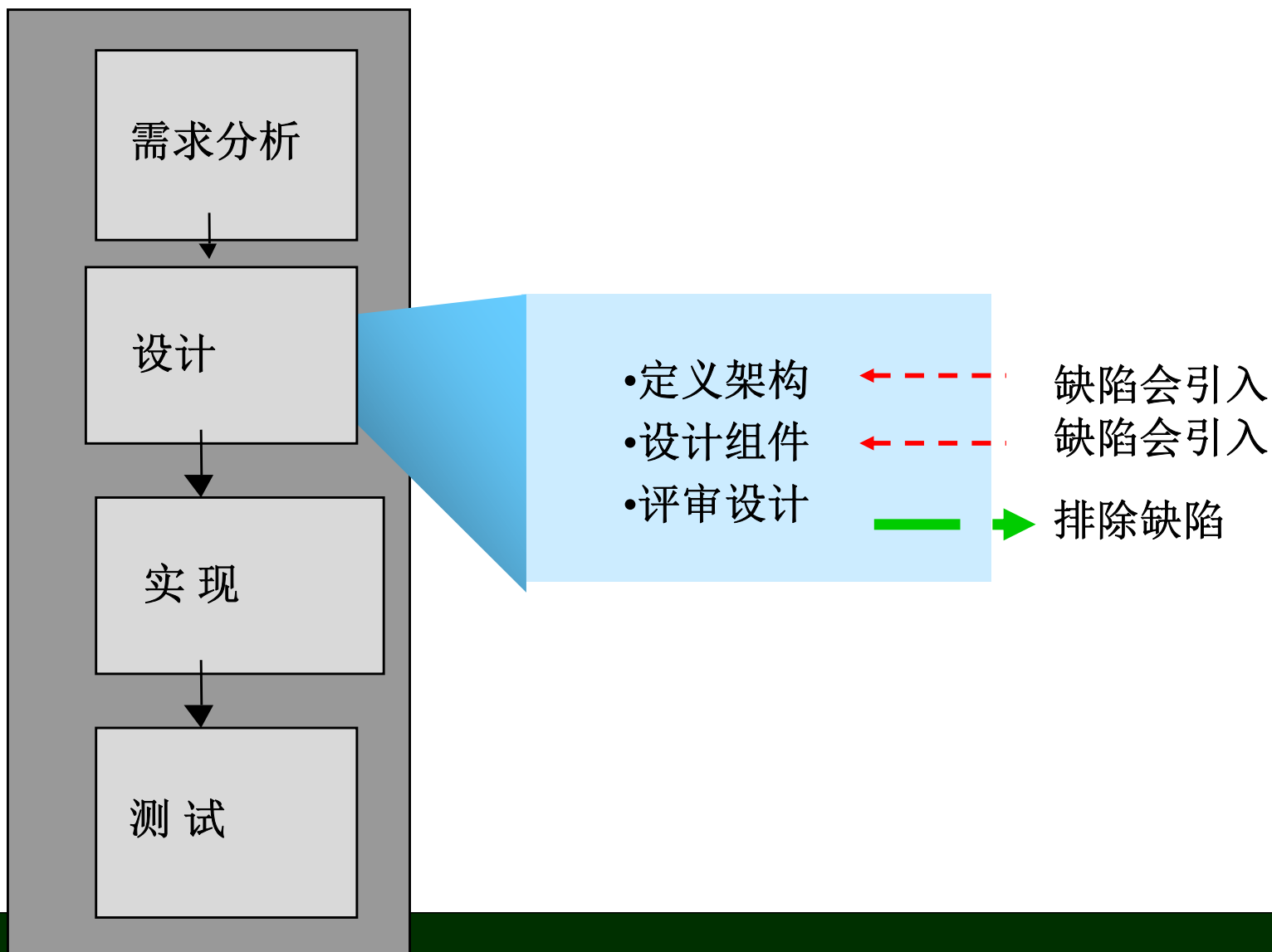


主题

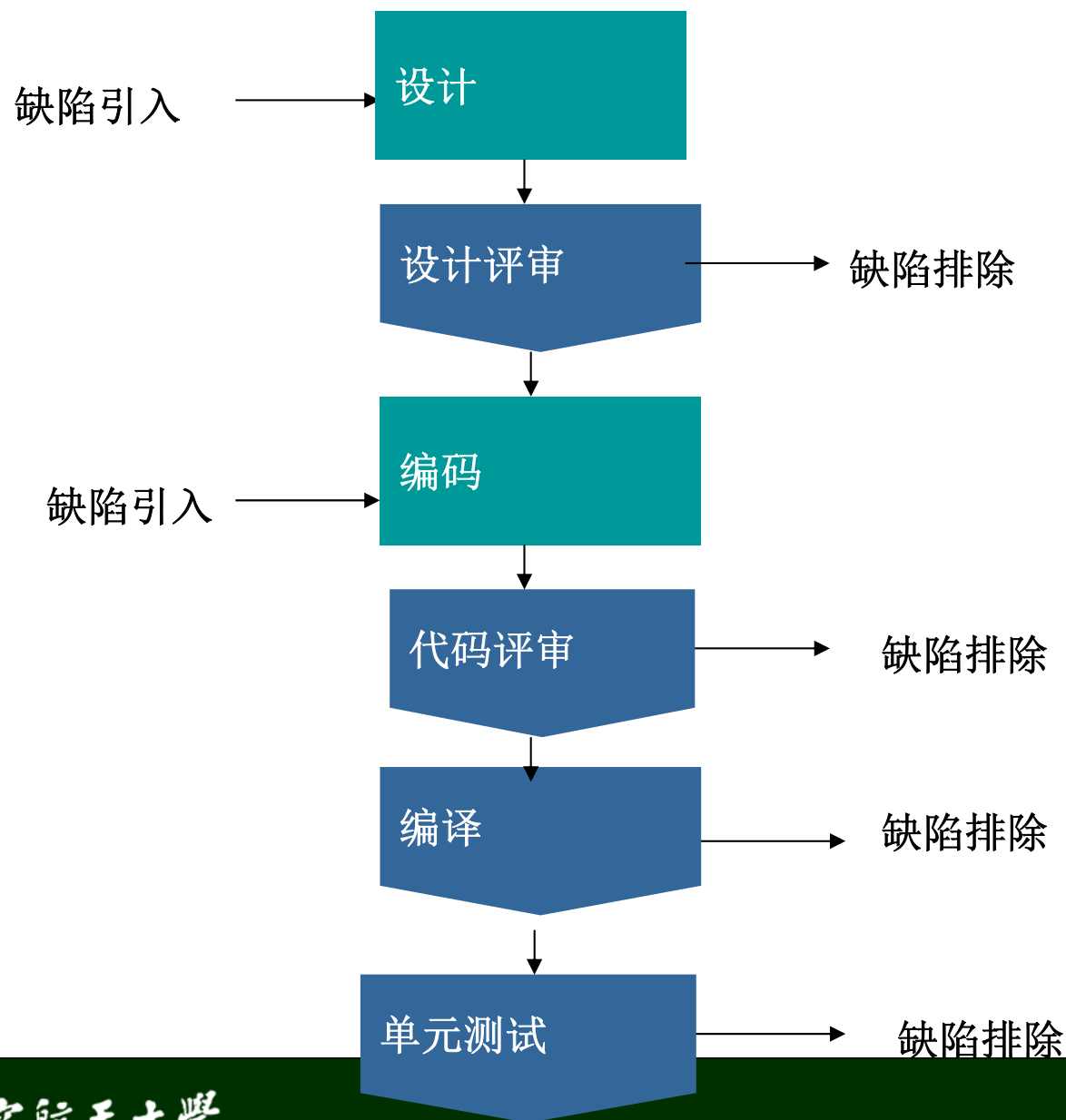
- 质量改进基本原则
- ➡ • 一个量化质量改进建模方法
- 企业实施过程改进实例分享



2.1 分解软件过程



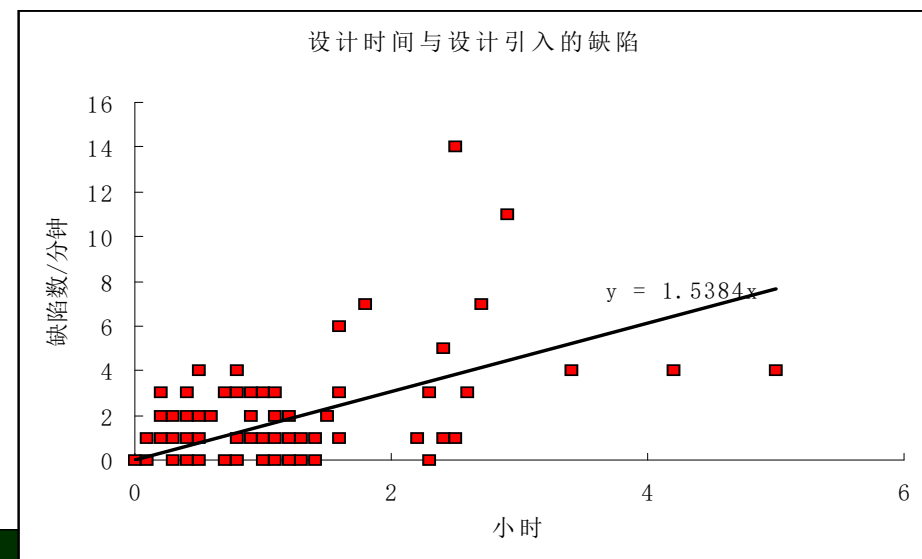
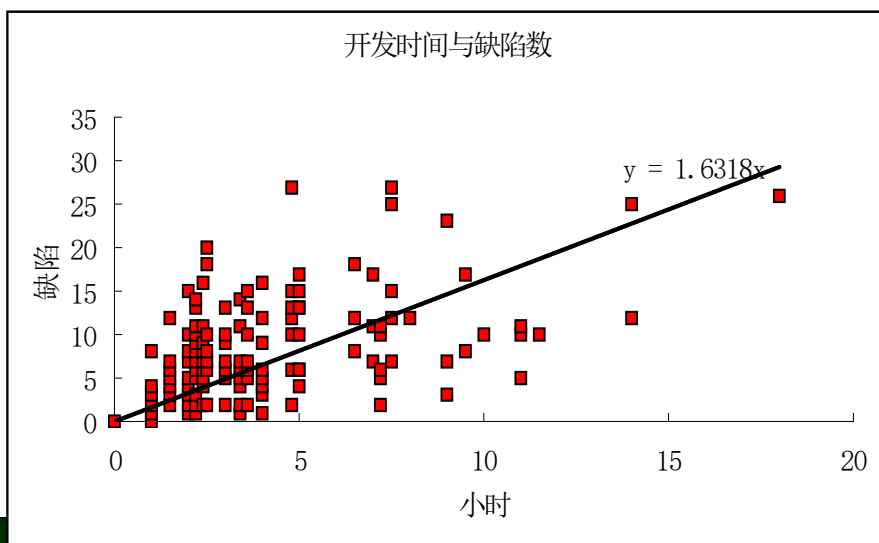
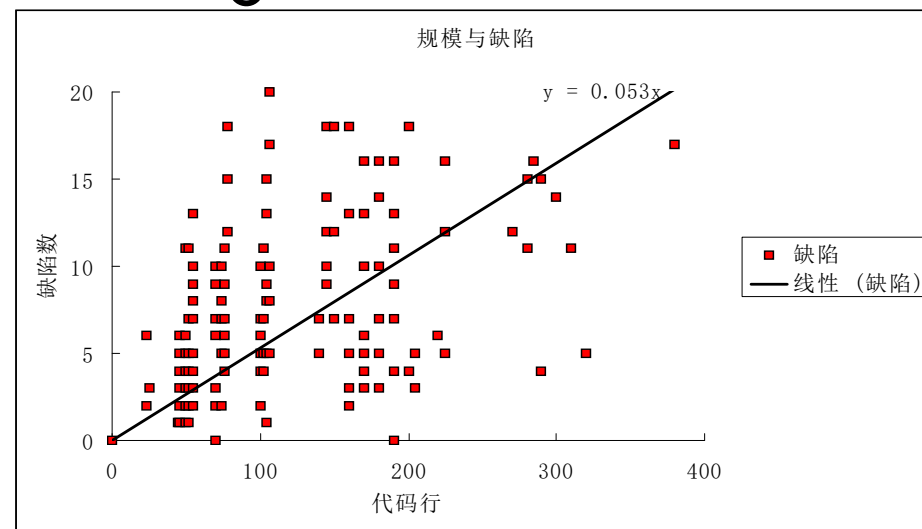
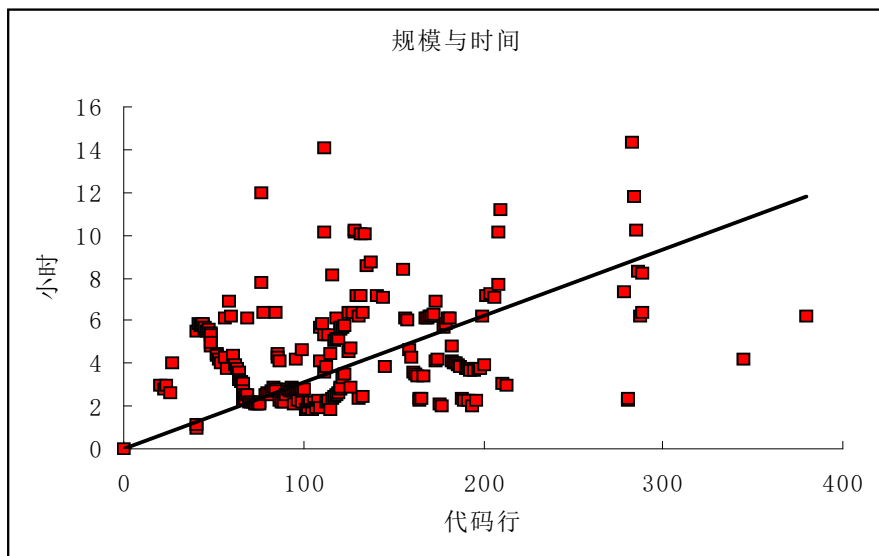
初始过程建模框架



把软件过程分解：
把它视为一组缺陷引入
和缺陷排除的活动

规模和时间与缺陷

观察出什么情况？



2.2 分析时间VS引入缺陷-1

- 规模和时间：其中任何一个都可来对缺陷引入进行建模
- 选择时间来对引入的缺陷进行建模：
 - 可以容易地应用到缺陷引入阶段
 - 得到的结果与活动而言，使用规模得到的结果相似
- 对任何一个缺陷引入活动而言，以设计阶段的为例：
- $$DI_{dsgn} = (\text{引入缺陷率}_{dsgn}) * (\text{时间}_{dsgn})$$
- 上式中，
 - DI_{dsgn} 表示在设计阶段引入的缺陷数目
 - 引入缺陷率 $_{dsgn}$ 表示在设计阶段每小时产生的缺陷数目
 - 时间 $_{dsgn}$ 表示设计阶段花费的时间



时间VS引入缺陷-2

- 相似的，对于编码阶段而言，

$$DI_{code} = (\text{引入缺陷率}_{code}) * (\text{时间}_{code})$$

- 上式中，
 - DI_{code} 表示编码阶段引入的缺陷数目
 - 引入缺陷率 $_{code}$ 表示编码阶段每小时引入缺陷的数目
 - 时间 $_{code}$ 表示编码时间



2.3 量化分析缺陷排除活动

- 缺陷排除活动包括：
 - 测试
 - 同行评审/审查
 - 个人评审
 - 编译

阶段排除缺陷收益

阶段	假设前提	有关的度量	阶段缺陷排除收益计算
测试	<ul style="list-style-type: none">有适当的测试时间缺陷随机分布	<ul style="list-style-type: none">产品的缺陷数目测试过程及方法的有效性	<ul style="list-style-type: none">测试运行的代码中发现的缺陷所占代码的百分比
评审	<ul style="list-style-type: none">有充足的时间对产品进行评审评审人具有良好的评审工作状态	<ul style="list-style-type: none">产品的缺陷数目评审过程及方法的有效性	<ul style="list-style-type: none">评审代码时发现的缺陷所占代码中的百分比
编译	<ul style="list-style-type: none">编译器可以100%的发现编译过程中的语法缺陷	<ul style="list-style-type: none">代码中语法类型缺陷的数目	<ul style="list-style-type: none">编译发现语法缺陷所占进入到编译阶段的缺陷数目的百分比



代码评审阶段的缺陷排除计算

- 代码评审阶段排除的缺陷数目为：

$$DR_{CR} = (\text{收益率}_{CR}) * (\text{缺陷数}_{CR})$$

- 上式中，
 - DR_{CR} 表示代码评审阶段排除的缺陷数目
 - 收益率 $_{CR}$ 表示代码评审阶段收益
 - DE_{CR} 表示代码进入到代码评审阶段时的缺陷数目



编译阶段的缺陷排除计算

- 编译阶段排除的缺陷数目为：

$$DR_{comp} = (PS_{comp}) * (缺陷数_{comp})$$

- 上式中，
 - DR_{comp} 表示编译阶段过程中排除的缺陷数目
 - 缺陷数_{comp} 表示代码进入到编译阶段的缺陷数目
 - PS_{comp} 表示 DE_{comp} 中语法缺陷所占的百分比



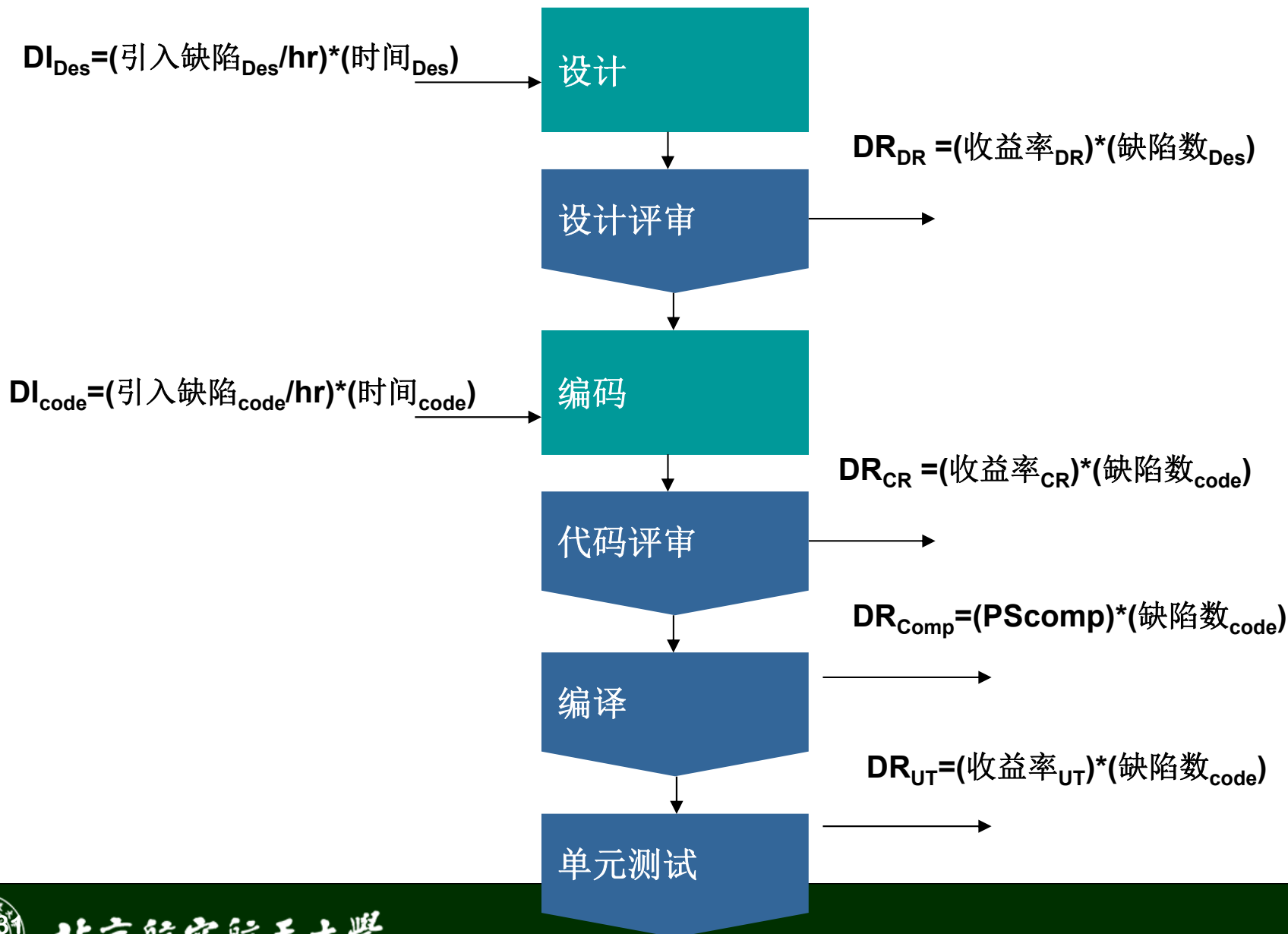
单元测试阶段的缺陷排除计算

- 单元测试阶段的缺陷排除数目为：

$$DR_{UT} = (\text{收益率}_{UT}) * (\text{缺陷数}_{UT})$$

- 上式中，
 - DR_{UT} 表示单元测试阶段的缺陷排除数目
 - 收益率 $_{UT}$ 表示单元测试阶段收益
 - 缺陷数 $_{UT}$ 表示程序进入单元测试阶段的缺陷数目

2.4 建立缺陷引入与排除模型

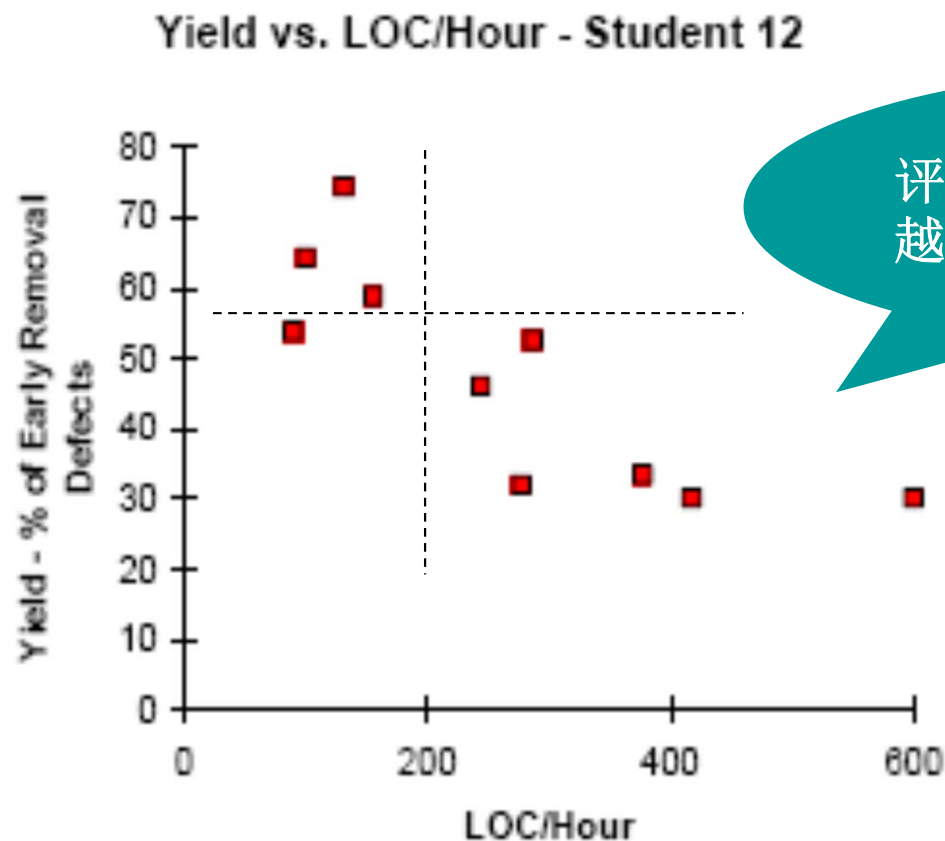


考虑时间关系

- 目前模型包含了：
 - 引入的缺陷数目
 - 排除的缺陷数目
 - 引入缺陷活动花费的时间
- 将时间的概念添加到排除缺陷的活动中
 - 同行评审/审查
 - 个人评审
 - 编译
 - 测试



评审速率与评审收益



评审速率越高，评审收益越低。

- 高质量评审过程：低于**200LOC/小时**的个人评审率，收益达到**60%~80%**

评审时间

- 个人花费在代码评审的时间可以用以下公式来表示:

$$\text{时间}_{\text{CR}} = (\text{规模}_{\text{编码}}) / (\text{RR}_{\text{CR}})$$

- 上式中,
 - 时间_{CR}表示代码评审的时间
 - 规模_{code}表示评审的代码行数
 - **RR_{CR}**表示每小时评审的代码行数



编译时间和测试时间

- 缺陷排除活动花费的时间通常都是查找并修复缺陷所用的时间。
- 编译阶段(或测试阶段)就可以用以下公式来计算:

$$\text{时间}_{\text{comp}} = (\text{DR}_{\text{comp}}) * (\text{FT}_{\text{comp}})$$

- 上式中,
 - 时间_{comp}表示编译阶段的时间
 - **DR_{comp}**表示编译阶段排除的缺陷数目
 - **FT_{comp}**表示编译中查找和修复缺陷的平均时间



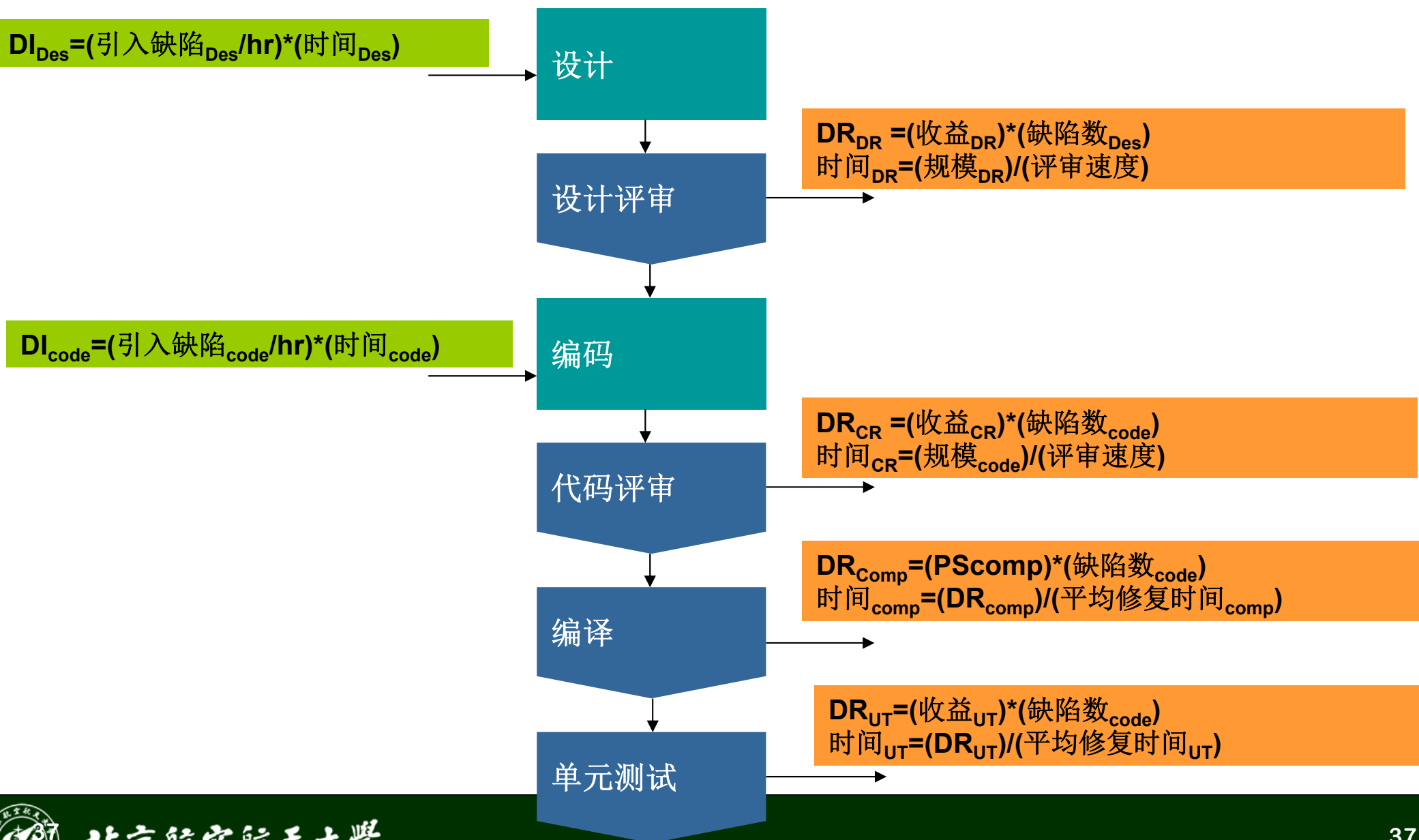
各种典型的收益

- 一些典型的阶段收益和平均查找和修复的时间

阶段	收益	平均查找和修复
设计评审	50%~80%	
编译		2~4 分钟
代码评审	50%~80%	
单元测试	50%~70%	15~40 分钟
集成测试	40%~50%	2~6 小时
系统测试	35%~50%	4~16 小时



缺陷引入与排除度量模型进化



2.4 过程建模举例

- 如一个过程模型，从设计阶段一直到系统测试，产生了一个**1000**代码行的程序。
- 使用前面提到的一些典型数据作为输入
- 过程模型计算如下：

阶段	收益	平均查找和修复
设计评审	50%~80%	
编译		2~4 分钟
代码评审	50%~80%	
单元测试	50%~70%	15~40 分钟
集成测试	40%~50%	2~6 小时
系统测试	35%~50%	4~16 小时

阶段	时间 (小时)	缺陷数/小时	阶段收益 (Yield)	引入的缺陷	排除的缺陷	遗留的缺陷
设计	4.00	2		8		8
编码	20.00	4.2		84		92
编译	3.07	15	50%		46	46
单元测试	15.33	1.5	50%		23	23
集成测试	23.00	0.4	40%		9.2	13.8
系统测试	38.64	0.125	35%		4.83	8.97
总计	104.04			92.00	83.03	8.97



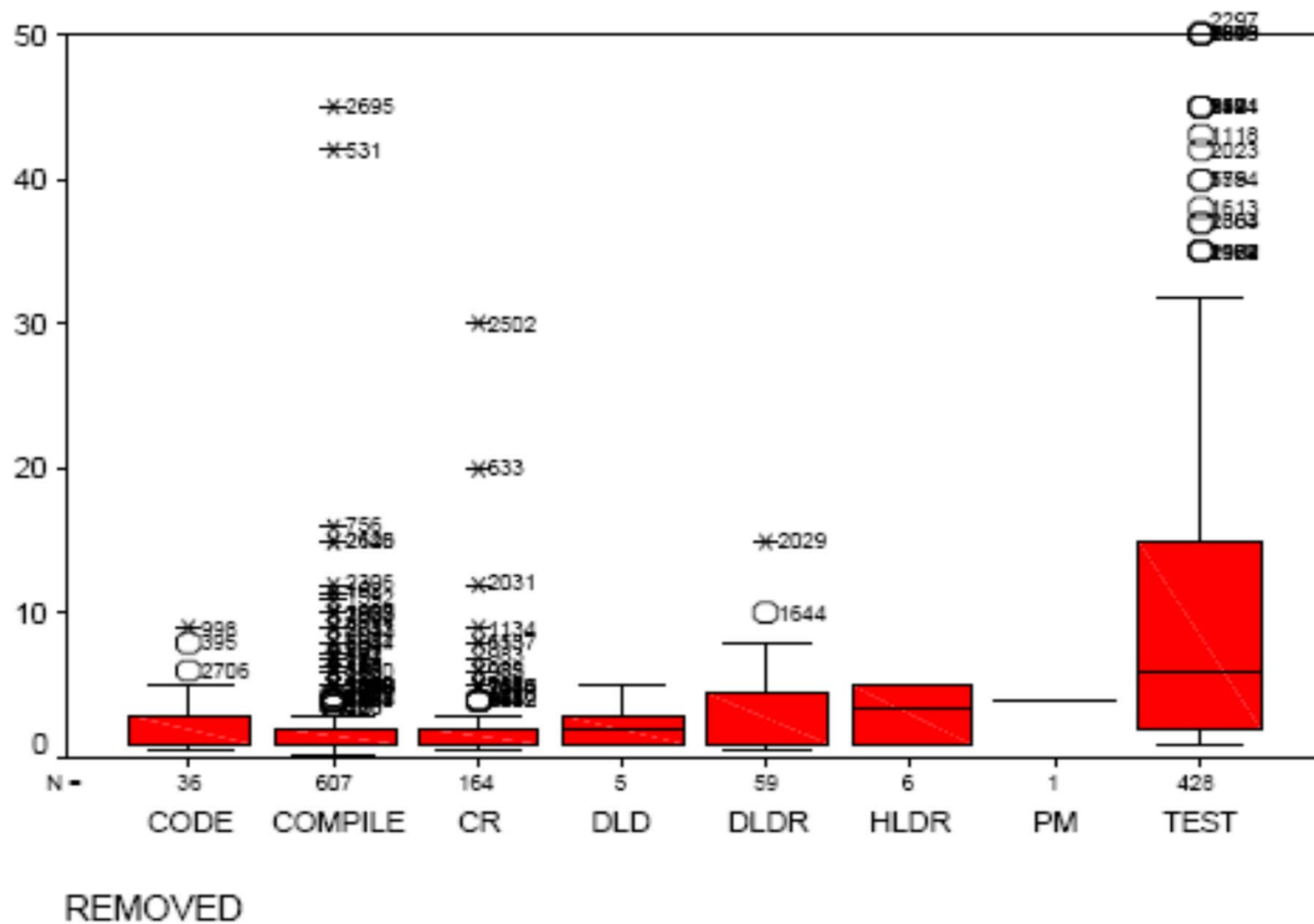
	输入的数据
	输出的数据

缺陷排除时间的预测

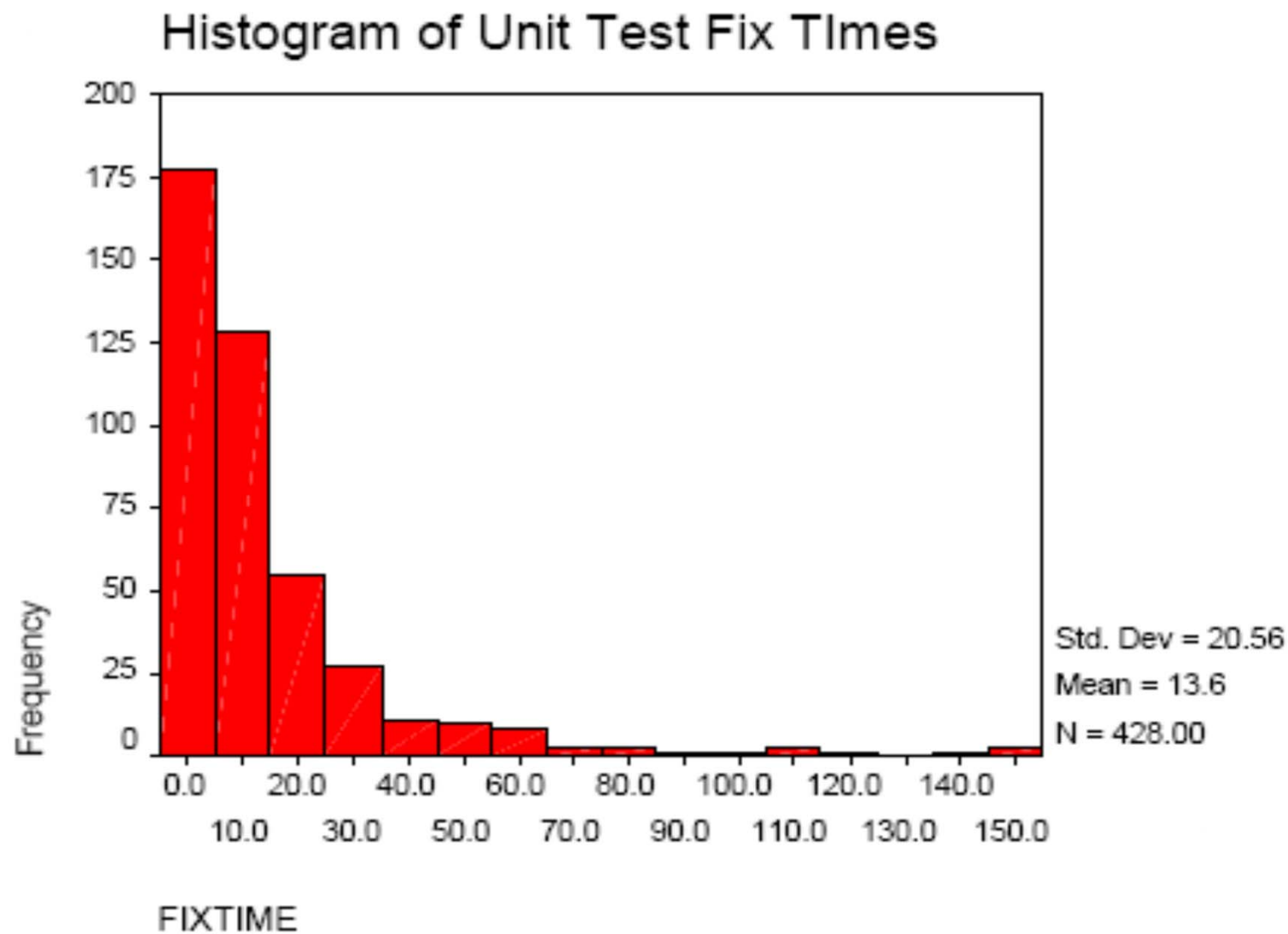
- 在模型上加上对预期的缺陷排除时间的范围进行预测的能力。
- 对于测试阶段来说，单个缺陷查找和修复的时间范围非常大。
- 提出的问题：假如我们在测试阶段有**N**个缺陷要排除，那么找到这**N**个缺陷并且修复它们的时间范围是多少？



修复时间的范围

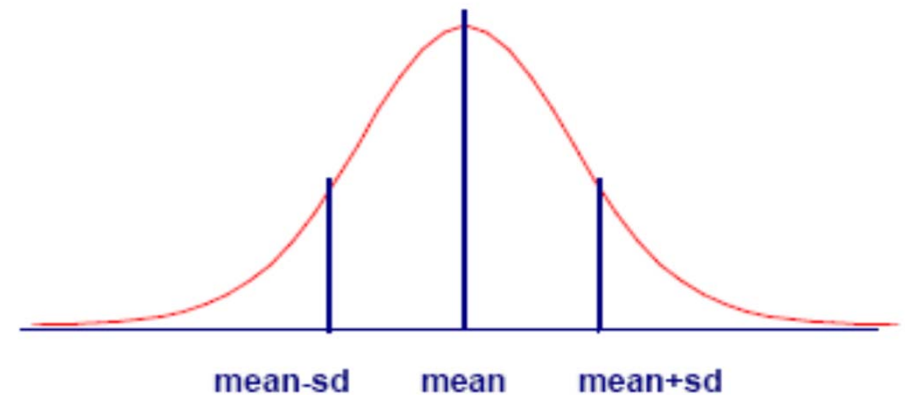


缺陷排除时间的分布



预期修复时间的范围—1

- 如果这个分布和正态分布类似，那么最可能的值就是这个分布的均值（**mean**）。
- 然后我们可以用均值±方差来预测范围，大约**70%**的实际缺陷发现和修复时间都落在这个范围内。



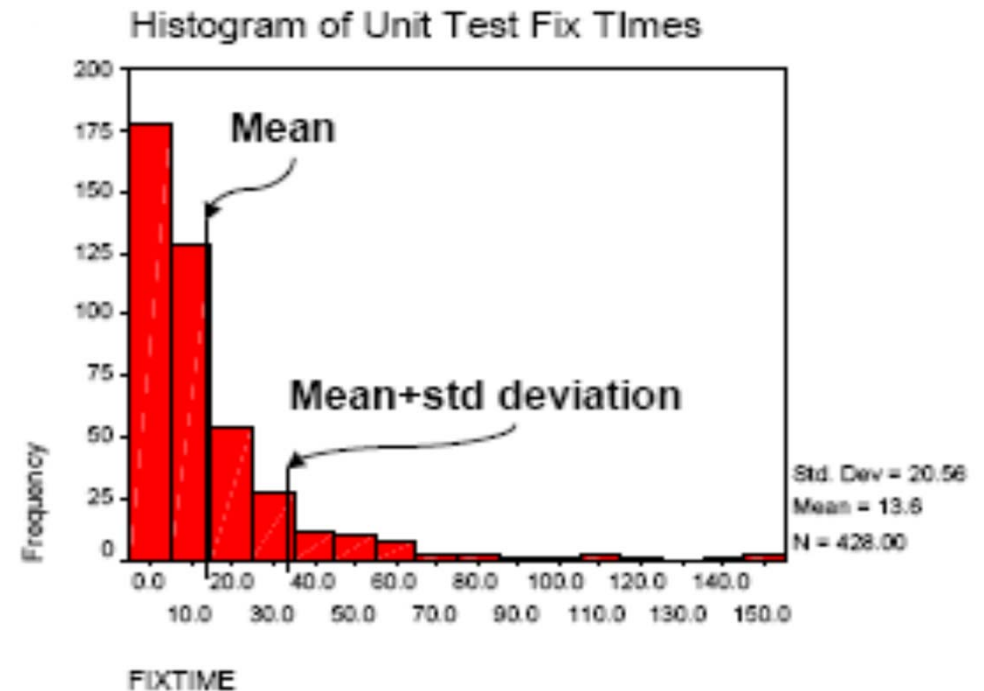
预期修复时间详解-1

- 用排除一个缺陷的均值和标准方差以及中心极限定律所给出的公式，来定义排除**N**个缺陷的排除时间的分布。
- 当**N**很大时，就会表现为一个正态分布。
 - 均值是最可能的值
 - **70%**的分布落在[均值±标准方差]之内
- 当**N**很小时
 - 均值大于最可能的值
 - 超过**70%**的分布落在[均值±标准方差]之内



预期修复时间详解-2

- 以前面的分布为例
 - 均值=13.6
 - 标准方差=20.6
 - 均值+标准方差=34.2
- 在正态分布中，分布的**70%**的点都落在[均值±标准方差]之内。
- 在这个分布中，**70%**的点都落在**0到15**之间，稍微超过**13.6**的均值



修复时间的预测

- 每个阶段的缺陷排除时间的范围，可以用下述单个缺陷排除时间的分布的参数计算得出。

缺陷排除时间范围		
阶段	均值	标准方差
编译(分钟)	4	6.99
单元测试(分钟)	40	60.4
集成测试(小时)	2.5	3.76
系统测试(小时)	8	12.08

总范围：

不是单个范围的和

是的单个范围的平方的和的平方根

当单个估计是无偏的时候，上述论断成立

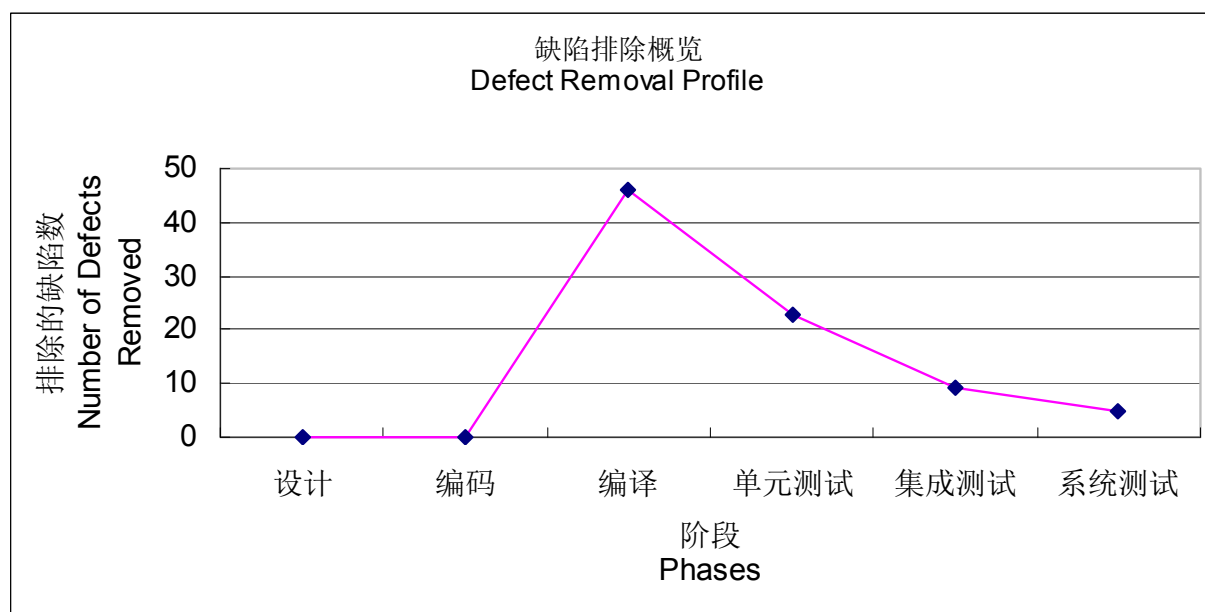
加入修复时间范围

阶段	时间 (小时)	70%范围 (小时)	缺陷数/小时	阶段收益 (Yield)	引入的缺陷	排除的缺陷	遗留的缺陷
设计	4.00		2		8	0	8
编码	20.00		4.2		84	0	92
编译	3.07	0.79	15	50%		46	46
单元测试	15.33	4.83	1.5	50%		23	23
集成测试	23.00	11.40	0.4	40%		9.2	13.8
系统测试	38.64	26.55	0.125	35%		4.83	8.97
总计	104.04	29.31			92.00	83.03	8.97

	输入的数据
	输出的数据

基本过程模型结果

从缺陷排除来评价过程，用一个缺陷排除的概览图



2.5 模型用于改进的应用

- 改进过程

增加更早一些的缺陷排除阶段
可以对哪些有改变？

- 时间（进度）变化？
- 缺陷排除时间的范围（进度预测性）变化？
- 缺陷（质量）变化？

- 提高早期缺陷排除率，是缺陷排除前移，措施如下：

- ✓ 增加代码评审
- ✓ 增加代码审查
- ✓ 强调设计投入的工作量
- ✓ 增加设计评审

个人代码评审的例子

加一个代码评审阶段，来看看模型预测的结果。收益为65%，评审速率是100LOC/小时

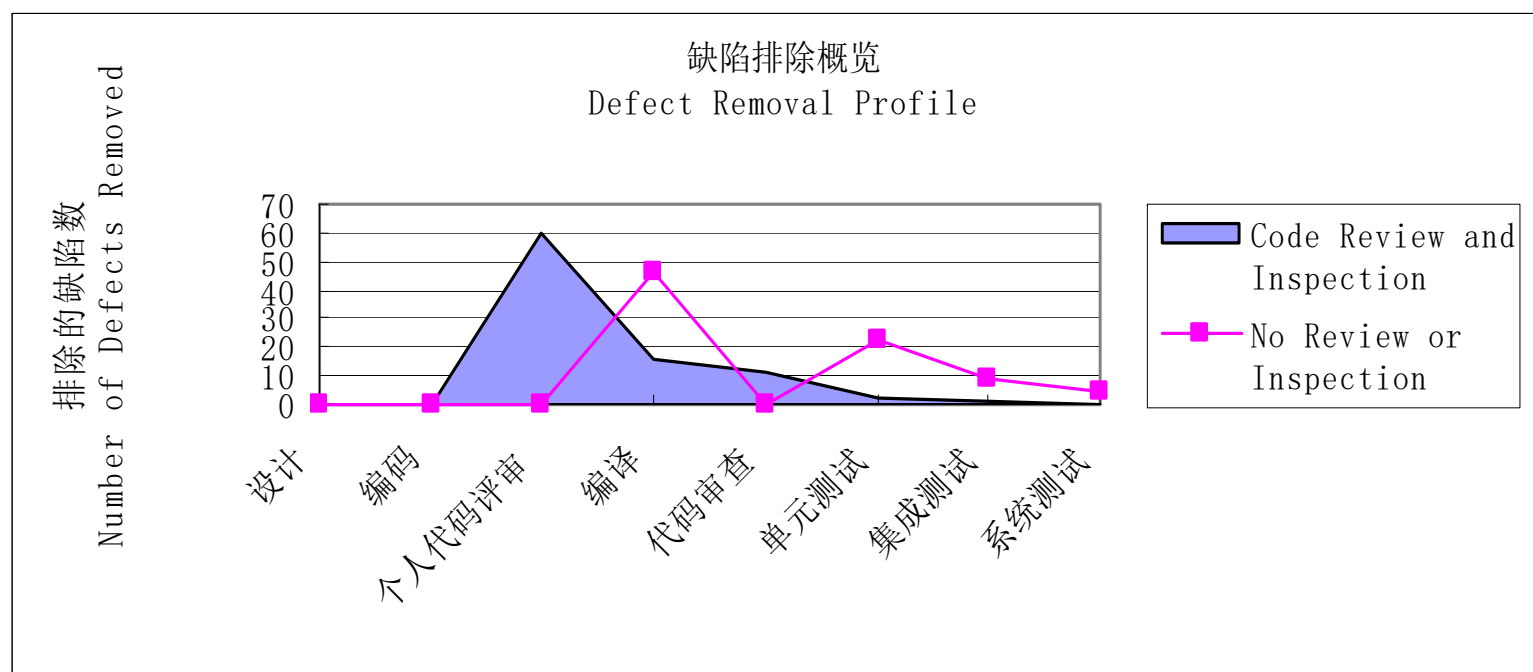
阶段	时间 (小时)	70%范围 (小时)	缺陷数/小时	阶段收益 (Yield)	引入的缺陷	排除的缺陷	遗留的缺陷
设计	4.00		2		8	0	8
编码	20.00		4.2		84	0	92
个人代码评审	10.00		5.98	65%		59.80	32.20
编译	1.07	0.47	15	50%		16.10	16.10
单元测试	5.37	2.86	1.5	50%		8.05	8.05
集成测试	8.05	6.75	0.4	40%		3.22	4.83
系统测试	13.52	15.71	0.125	35%		1.69	3.14
总计	62.01	17.34			92.00	88.86	3.14

加一个同级的代码审查，来看看模型预测的结果。3个评审员，收益为70%，评审速率是150LOC/小时

阶段	时间 (小时)	70%范围 (小时)	缺陷数/小时	阶段收益 (Yield)	引入的缺陷	排除的缺陷	遗留的缺陷
设计	4.00		2		8	0	8
编码	20.00		4.2		84	0	92
个人代码评审	10.00		5.98	65%		59.80	32.20
编译	1.07	0.47	15	50%		16.10	16.10
代码审查	20.94		0.54	70%		11.27	4.83
单元测试	1.61	1.56	1.5	50%		2.42	2.42
集成测试	2.42	3.70	0.4	40%		0.97	1.45
系统测试	4.06	8.60	0.125	35%		0.51	0.94
总计	64.10	9.50			92.00	91.06	0.94



与之前的过程的比较-2



强调设计和设计评审

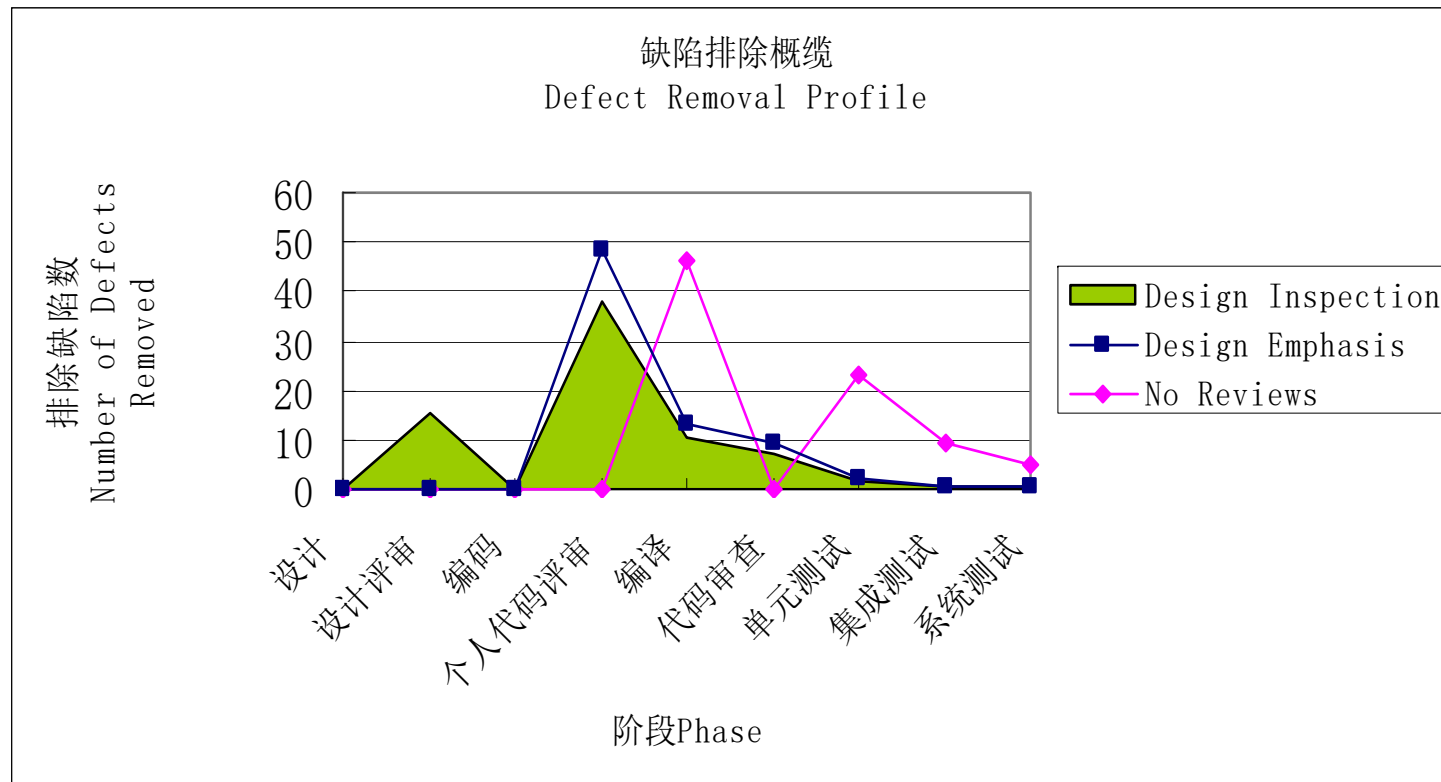
阶段	时间 (小时)	70%范围 (小时)	缺陷数/小时	阶段收益 (Yield)	引入的 缺陷	排除的 缺陷	遗留的缺陷
设计	12.00		2		24	0	24
设计评审	6.06		2.57	65%		15.6	8.4
编码	12.00		4.2		50.4	0	58.8
个人代码评审	6.06		6.31	65%		38.22	20.58
编译	0.69	0.37	15	50%		10.29	10.29
代码审查	20.60		0.35	70%		7.20	3.09
单元测试	1.03	1.25	1.5	50%		1.54	1.54
集成测试	1.54	2.95	0.4	40%		0.62	0.93
系统测试	2.59	6.88	0.125	35%		0.32	0.60
共计	62.57	7.60			74.40	73.80	0.60

设计评审时间为设计时间的
1/2。且等于编码时间

代码评审时间为编码时间的
1/2

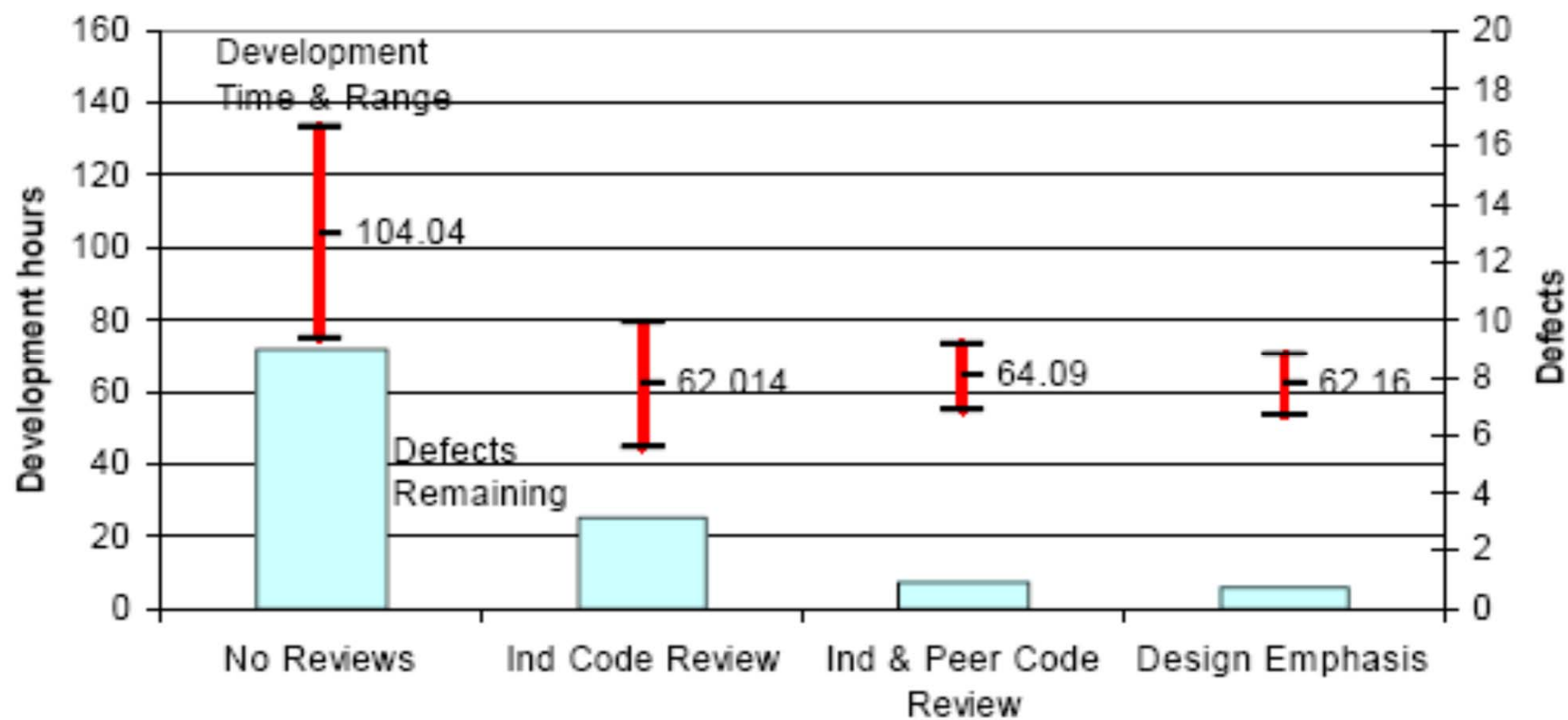


改进后的结果1

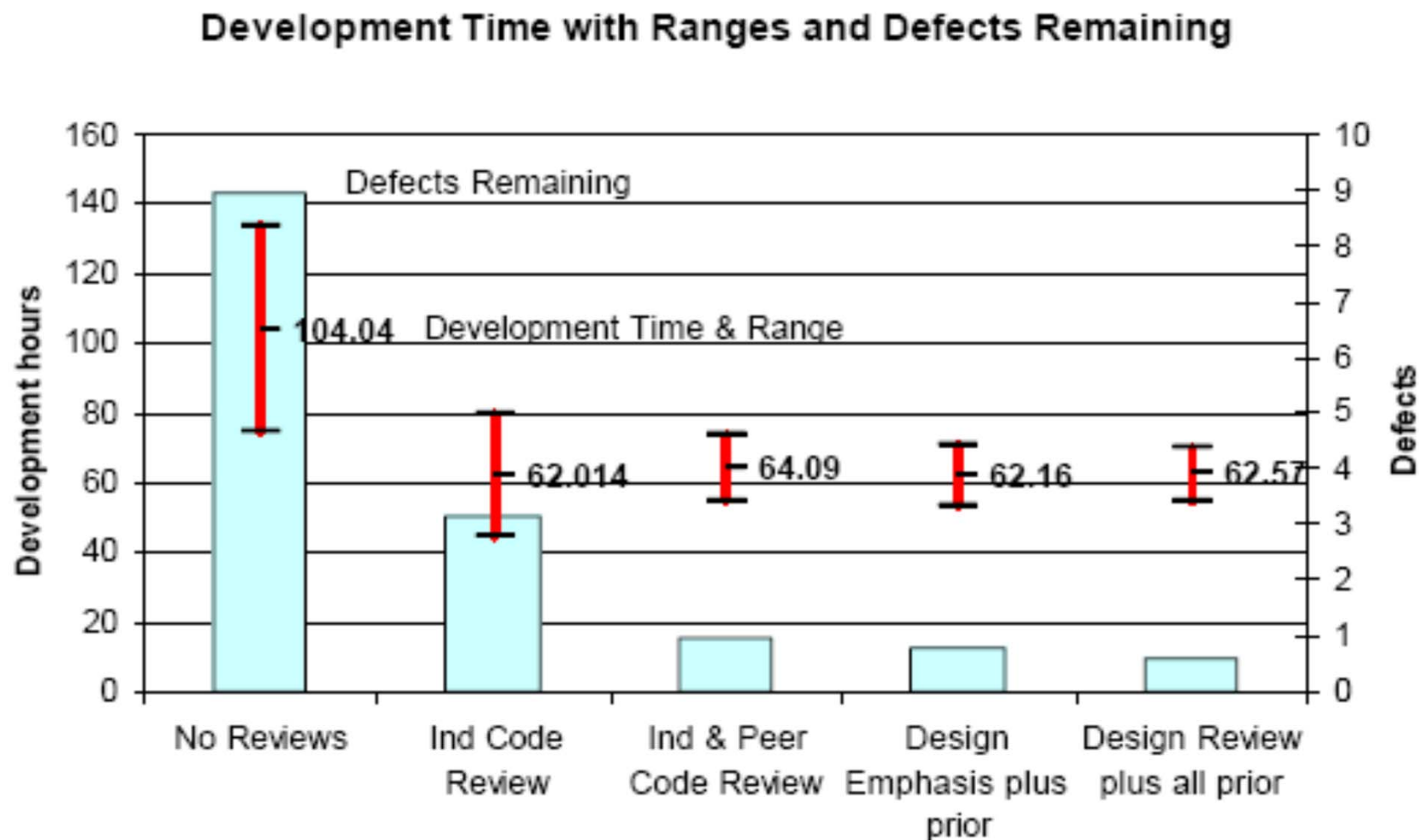


强调设计

- 开发时间和范围以及残留的缺陷



加一个设计评审



改进后的结果2

改变过程	没有评审	个人评审后	代码审查后	强调设计后	强调设计评审后	总下降百分比
时间 (H)	104.04	62.01	64.10	62.16	62.57	39.9%
70% 范围	29.31	17.34	9.50	8.55	7.60	74.1%
总缺陷数	92.00	92.00	92.00	74.40	74.40	19.1%
遗留缺陷	8.97	3.14	0.94	0.76	0.60	93.3%

主题

- 质量改进基本原则
- 一个量化质量改进建模方法
- • 企业实施过程改进实例分享



小结1：缺陷排除效益

- 缺陷在程序中的时间越长，越难被发现
- 阶段效益是：
 - 发现缺陷是阶段有效性的度量
 - 给定阶段缺陷的排除率
- 早期排除缺陷效益会更高
 - 评审和审查：效益值为**60%-80%**
 - 单元测试：效益值为**50%**
 - 集成测试和系统测试：效益值在**50%**以下

小结2：修复时间偏差

- 缺陷在程序中的时间越长，修复时间花费偏差就越大
- 随着平均修复时间的增加，最大和最小修复时间的范围也会增加
- 成本和进度估计的准确性也会受到影响

小结：早期移除缺陷的好处

- 减少周期时间和开发费用
- 提高生产率和质量
- 提高估计的准确性
- 更进一步的好处是提高过程控制
- 阶段效益和修复时间数据能用来决定
 - 阶段出口准则
 - 剩余的工作量、花费和进度

参考材料

- 材料主要参考
 - **A Quantitative Approach to Software Quality Management, Dan Burton and James Over, SEI**
 - **《PSP—软件工程师自我改进过程》, PSP: A self-improvement Process for Software Engineering. Watts S. Humphrey. 吴超英等译, 人民邮电出版社, 2006**



