Measures

代码走查最佳实践及案例

麦哲思科技（北京）有限公司 关钦杰

议题

- 代码评审的最佳实践

内容

Measures

- 代码评审的意义
- 代码评审的方法
- 代码评审的最佳实践
- 代码评审的要点
- 代码评审度量数据的分析
- 代码评审案例分析

代码评审的意义

Measures

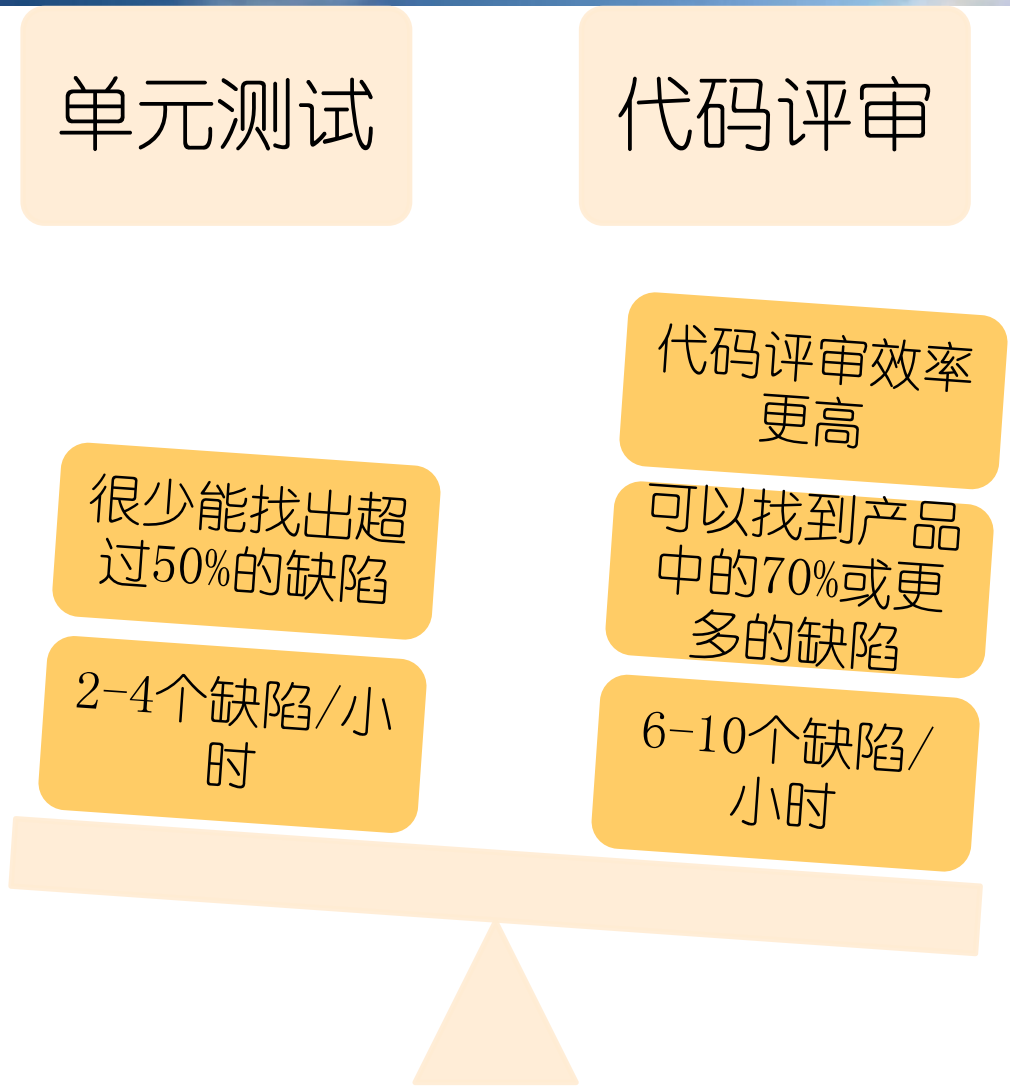
最重要的意义

- 有效发现缺陷
- 在早期发现缺陷

其他意义

- 对代码进行优化
- 保证代码规范
- 进行新人培训，帮助进行新人成长
- 建立良好的开发氛围

代码评审更有效


单元测试
代码评审
很少能找出超过50%的缺陷
2-4个缺陷/小时
代码评审效率更高
可以找到产品中的70%或更多的缺陷
6-10个缺陷/小时

评审成功的案例

- Yourdon和3个有经验的软件人员用45分钟时间评审了一个200行PL/1程序，发现了25个错误，其中有5个错误是不可能通过测试发现，他们认为，评审比测试更有效
- 在软件维护方面，Freedman和Weinberg报告，在引入评审前，变更维护出错率为55%，引入评审之后，这一出错率降至2%。另一报告说，引入评审后，产品故障率降低了77%
- AT&T的贝尔实验室在其开发中引入评审后的成功案例：
  - 生产率提高了14%，质量提高了10倍
  - 有一个大型电力交换系统，发现错误的成本降低了10倍
  - 在发现错误方面，评审的成效是测试的 20 倍
- TRW对一个大型软件进行了研究，发现2019个由用户发现的错误导致代码变更。分析结果表明，在这些错误中，通过代码评审可以发现62.7%，通过设计评审可以发现57.7%

案例：某CMMI3级企业的度量数据

Measures

项目名称	算法类项目A	MIS 类项目B
规模(行)	3200	10700
代码评审发现的缺陷	139	127
代码评审工作量(人时)	181.7	119.5
代码评审的效率(个/小时)	0.76	1.06
测试发现的缺陷	9	27
测试的工作量(人时)	185	115.5
测试效率(个/小时)	0.05	0.23
效率差异(评审/测试)	15.72	4.55

代码评审的常见方法

Measures

- 结对走查
- 会议走查
- 代码审查

代码评审的常见问题

Measures

- 评审的专家很难读懂被评审的代码，效率比较低
- 找到的缺陷大都是轻微缺陷
- 快速评审很多代码，没有发现很多问题
- 专家没有时间做代码评审
- 专家发现的问题作者不认可

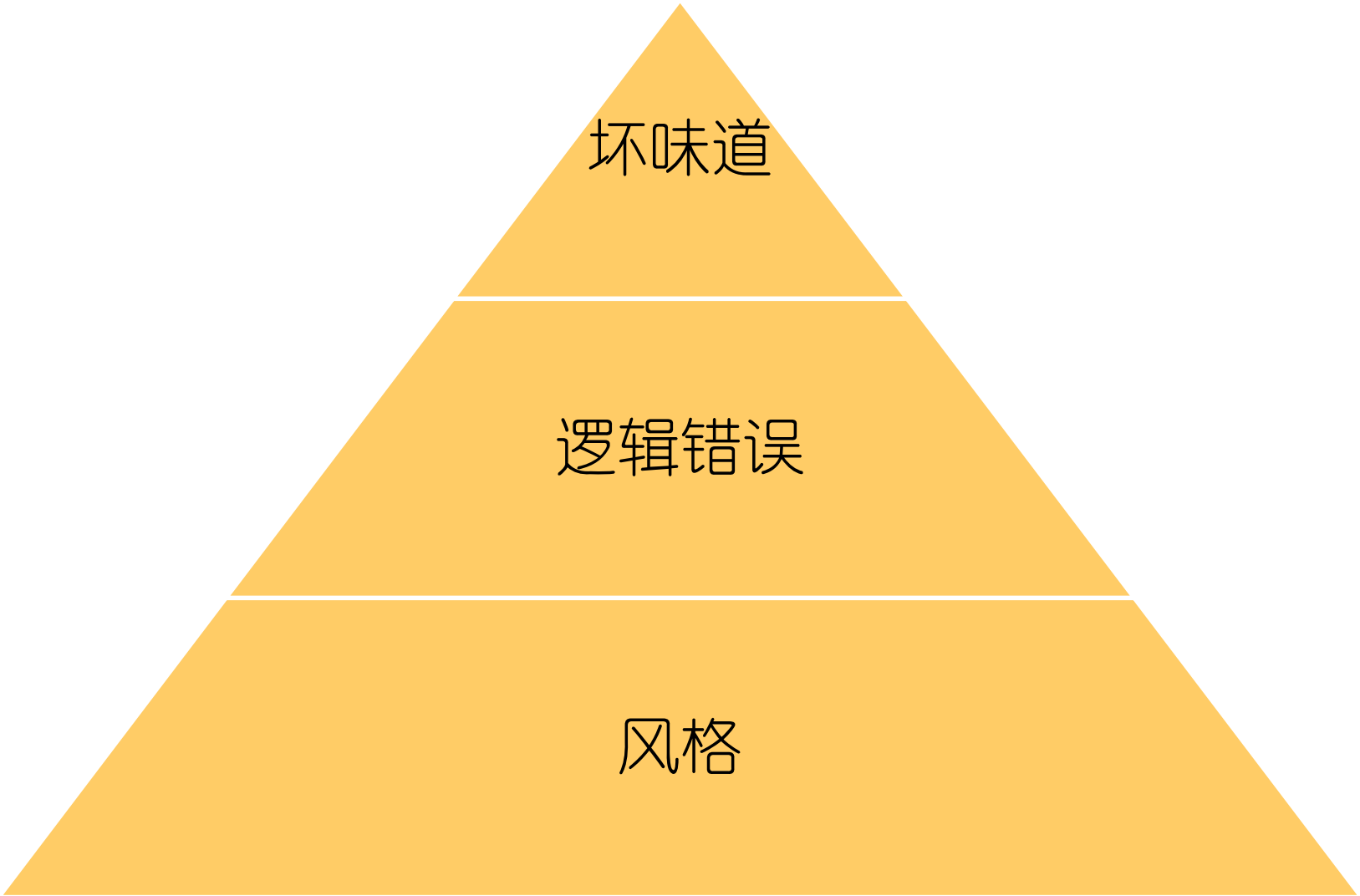
代码评审的最佳实践

Measures

- 先做个人评审再进行编译
- 坚持使用代码评审检查单
- 即使不能对所有代码做检查，也要对部分代码检查
- 轻量级代码走查更高效
- 每次检查少于200—400行代码，每次代码检查不超过90分钟
- 建立量化目标并获得相关指标数据从而不断改进流程
- 优先使用静态检查工具
- 建立代码走查的文化

代码评审的三个层次

Measures

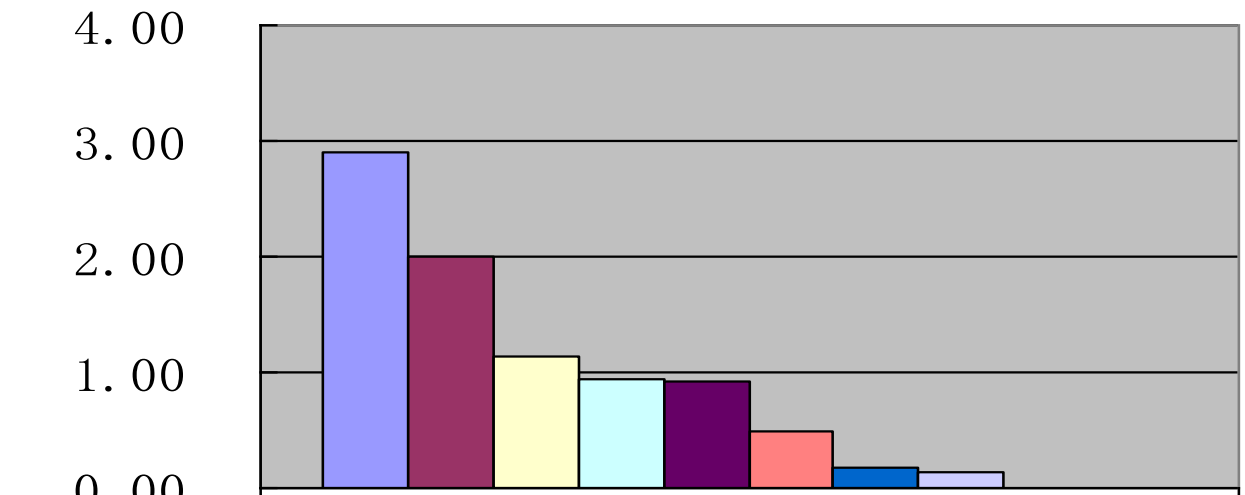


代码评审需要收集的数据

Measures

- 基本度量元
  - 评审的规模
    - 代码行数
  - 评审的时间
    - 个人评审的时间
    - 会议的时间
  - 评审的工作量
    - 个人评审的工作量
    - 会议的工作量
  - 发现的缺陷个数
    - 个人评审发现的缺陷个数
    - 会议发现的缺陷个数
- 派生度量元
  - 评审速率
    - 规模/时间
  - 评审效率
    - 缺陷个数/工作量
  - 缺陷密度
    - 缺陷个数/规模

案例：按模块的缺陷密度分析



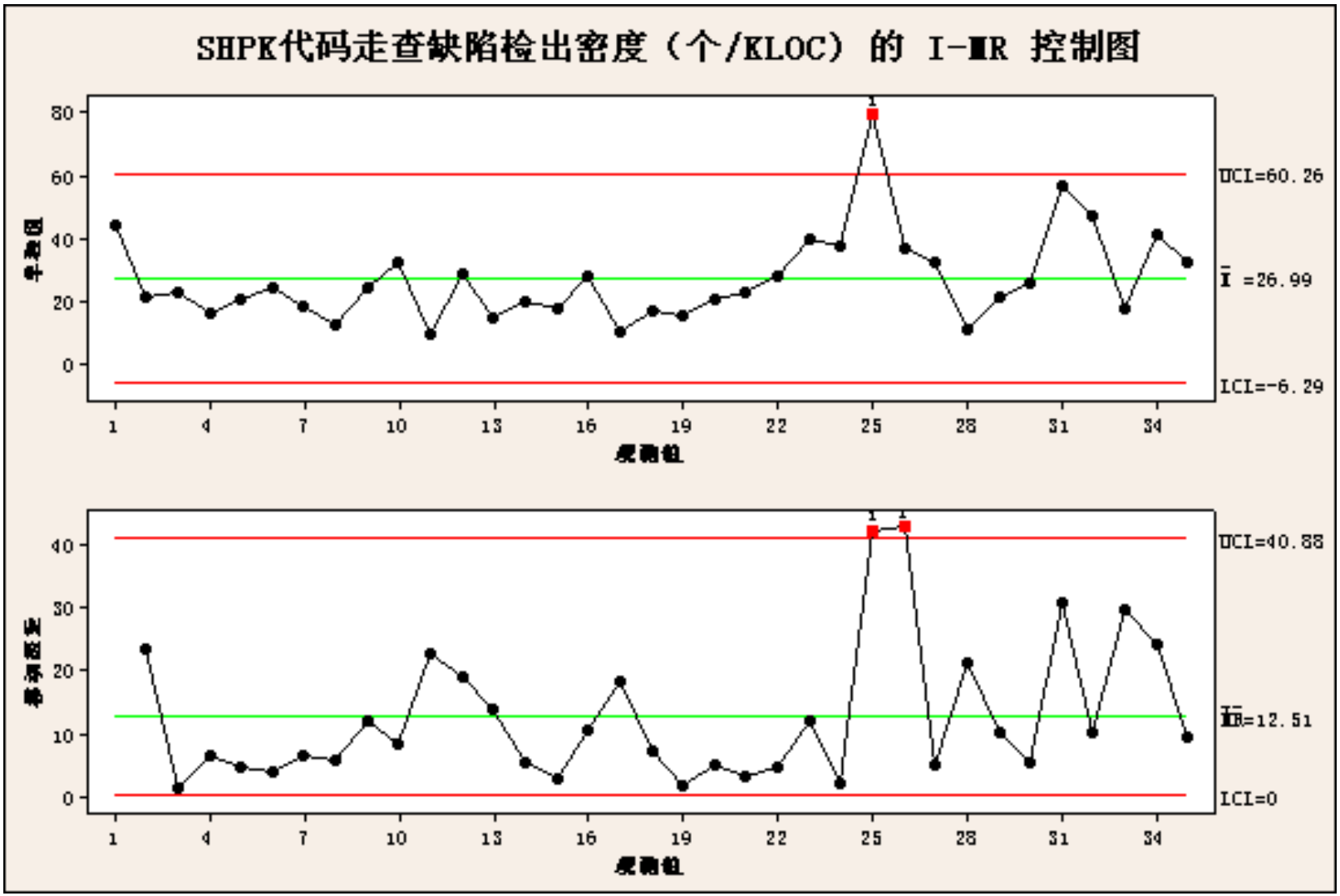
案例：按人员的缺陷检出密度分析

Measures

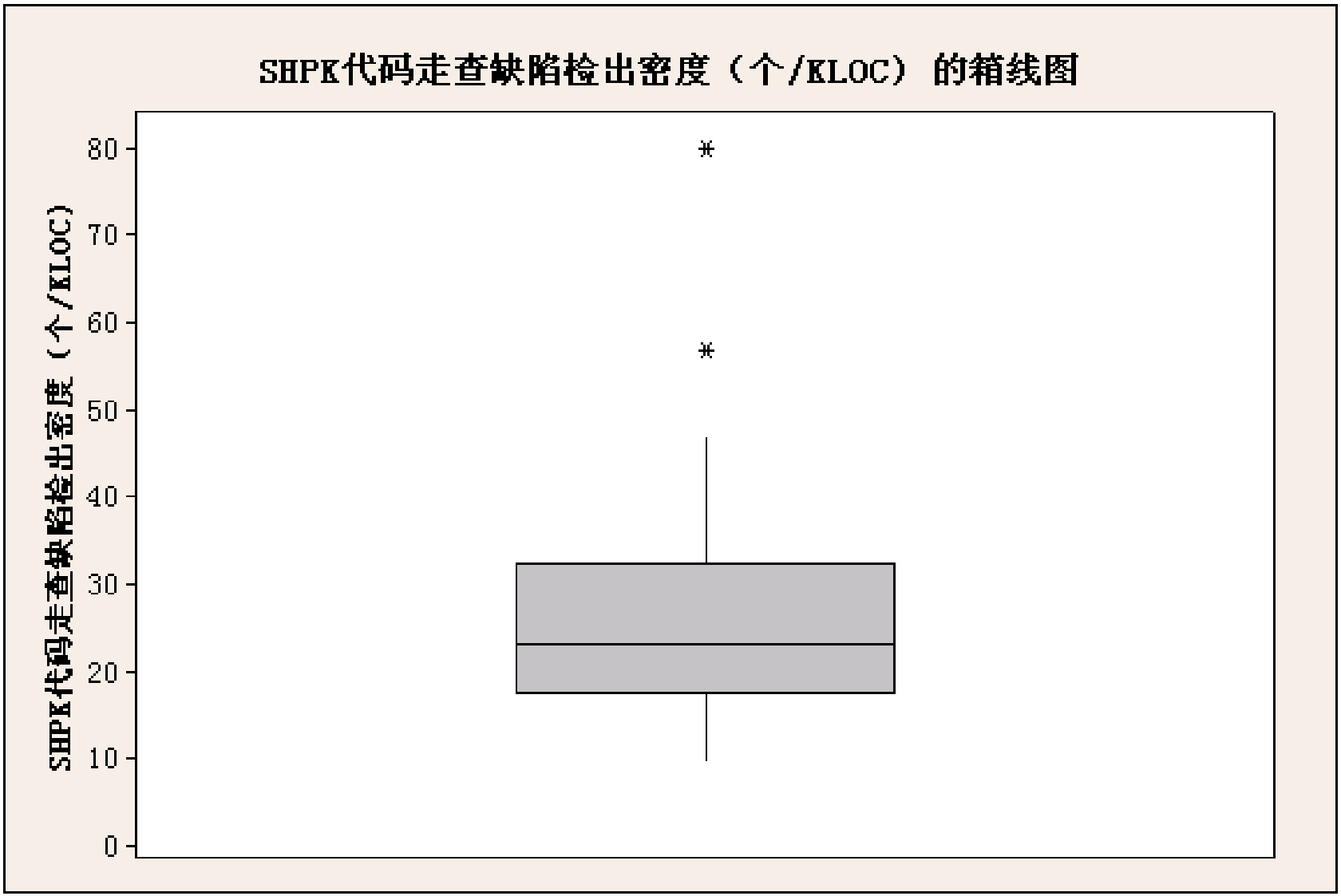
人员	代码行数	缺陷个数	缺陷检出密度
谢*	11734	236	20.11
顾**	8757	148	16.90
邱**	6710	78	11.62
徐*	7305	84	11.50
乔*	10884	119	10.93
强**	5222	57	10.92
牛**	8688	91	10.47
卢*	8370	80	9.56
张**	8511	81	9.52
熊*	10646	100	9.39
耿**	10417	76	7.30
孙**	4914	35	7.12
姚*	13974	93	6.66

案例：代码走查的缺陷密度控制图

Measures



案例：代码走查缺陷检出密度的箱线图分析



案例背景

Measures

- 某公司2009年11月初开始启动过程改进，第一阶段改进代码走读、系统测试、度量过程。
- 该公司是产品开发类项目，产品开发的周期比较短，销量比较大，质量要求比较高。
- 该公司每天下午5点到6点为固定的代码走读时间，对当天完成的代码进行走查。2人结对，一个作者，一个专家。

代码走查推广措施

Measures

类型	措施
制度	建立每日走读一小时的制度
	建立每日走读的流程
	定义每日走读的检查单
	代码走读检查单学习考试
	SQA对代码走读的质量进行抽检识别必须复检的走读
	定义代码走读发现的缺陷类型
度量	度量代码走读的效率、速率、工作量
宣传	制定代码走读的宣传标语
	代码走读的度量数据定期大幅度公布，代码走读的典型人物公开宣传
	根据每个人的度量数据识别出代码走读的榜样
人员培养	代码走读的典型案例采集
	典型代码的走读教育
	代码走读的内部培训讲师的识别
工具	分析pclint的报警数与系统测试的bugs、客户反馈的Bugs之间的相关性
	建立pclint的告警数在转系统测试时的阈值
	函数复杂度分析工具的使用SourceMonitor
	代码行数的统计工具
	程序流程自动分析工具的收集与研究
	定义Pclint查出的哪些告警必须修改
	代码走读的IT系统

12月份代码走查的原始数据

Measures

	软件1部	软件2部	软件3部	软件4部	软件5部	软件6部	合计
代码走查发现的Bugs	1035	908	465	153	782	344	**3687**
严重与致命bugs	113	125	76	4	98	48	**464**
代码走查投入工作量	753.2	791.6	476.4	96	711.4	257.5	**3086.1**
代码走查的行数	172912	132772	76388	18212	96802	30882	**527968**
代码走查缺陷检出效率(BUGS/人时)	1.37	1.15	0.98	1.59	1.1	1.34	**1.19**
严重与致命bugs缺陷检出效率	0.15	0.16	0.16	0.04	0.14	0.19	**0.15**
缺陷检出密度(个/KLOC)	5.99	6.84	6.09	8.4	8.08	11.14	**6.98**

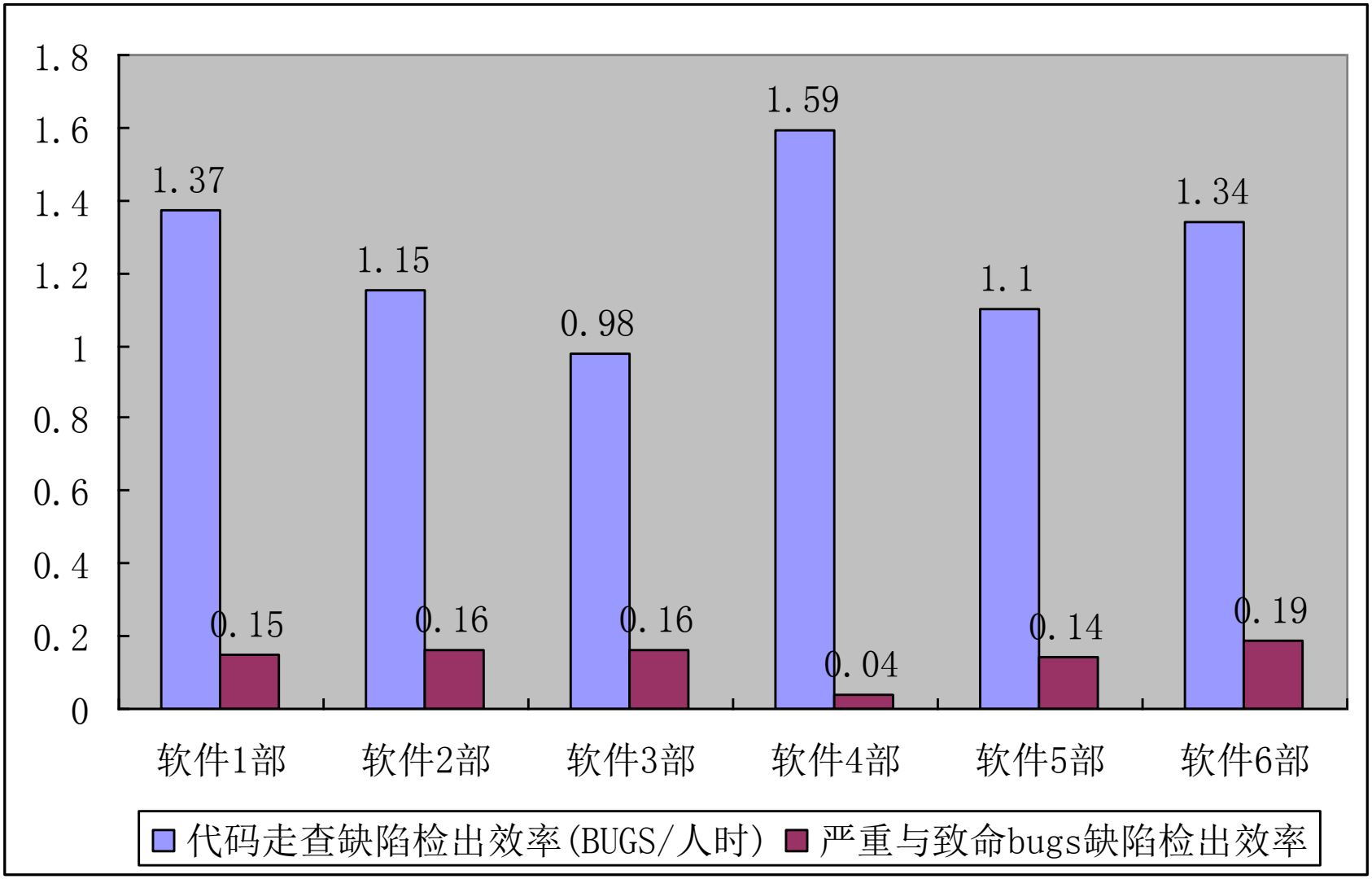
12月份代码走查的平均数据

Measures

代码走查缺陷检出效率(BUGS/人时)	1.19
严重与致命bugs缺陷检出效率（BUGS/人时）	0.15
缺陷检出密度(个/KLOC)	6.98
严重与致命bugs的检出密度(个/KLOC)	0.88

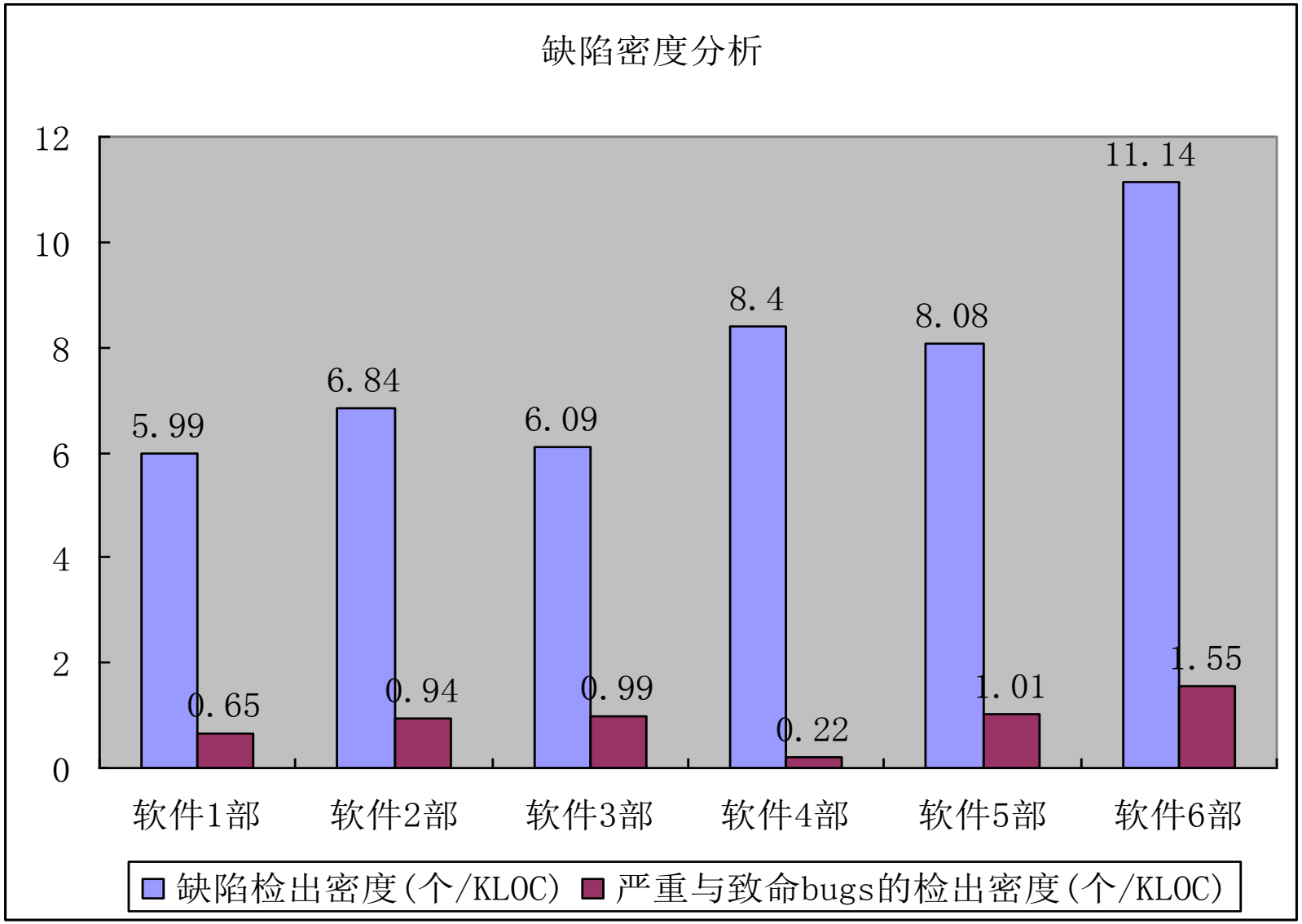
12月份各部门代码走查效率的分析

Measures



12月份缺陷检出密度的分析

Measures



12月份代码走查与系统测试效率的比较

Measures

	代码走查	系统测试	比值
发现的BUGS	3687	4556	**0.81**
发现的严重与致命BUGS	464	1511	**0.31**
工作量	3086.1	16062.4	**0.19**
检出效率(BUGS/人时)	1.19	0.28	**4.25**
严重与致命BUGS的检出效率(BUGS/人时)	0.15	0.09	**1.67**

Q & A