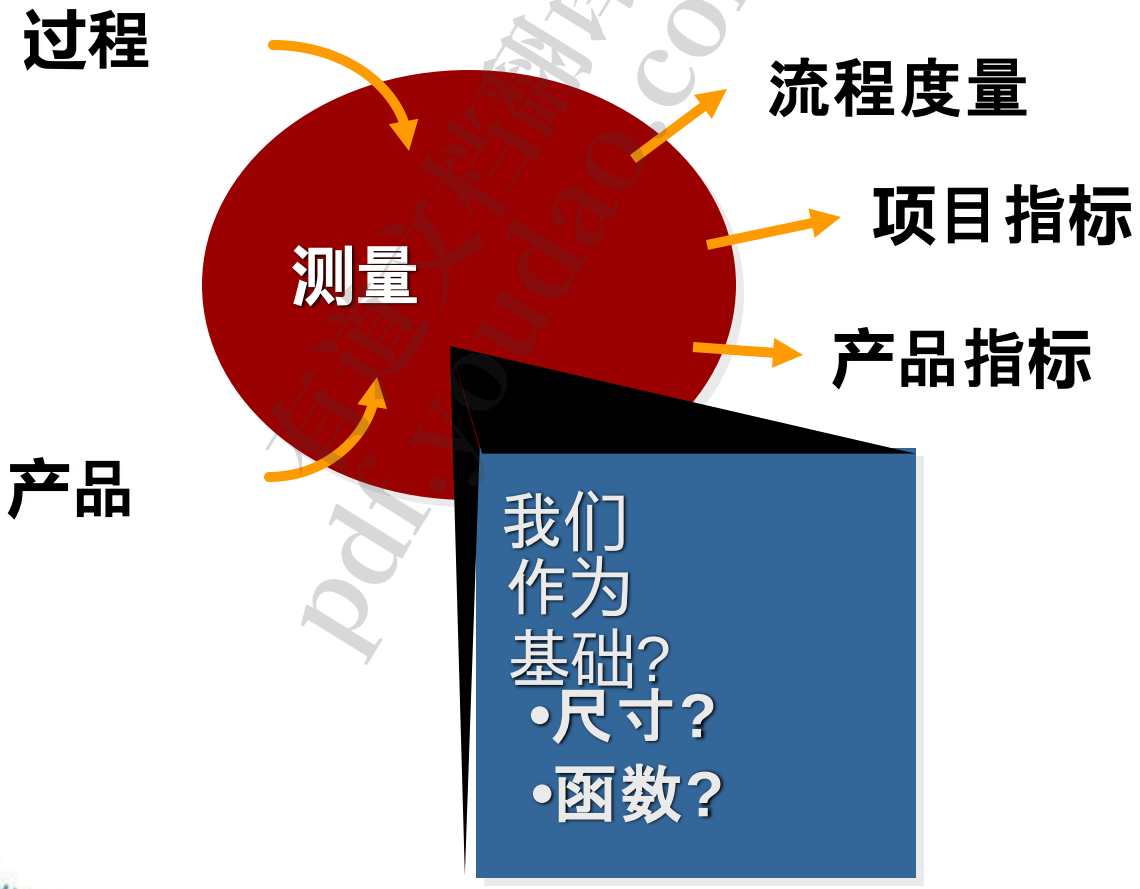
第32章过程和项目度量

- 一个好的管理者会衡量

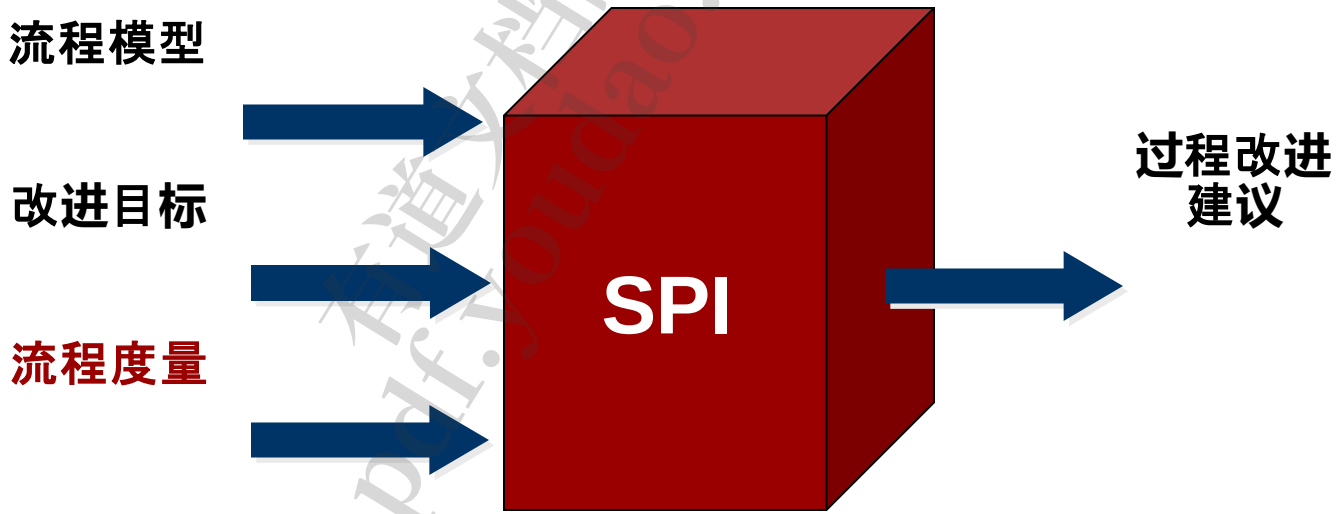


- **我们为什么要度量?**
  - 评估正在进行的项目的状态
  - 跟踪潜在的风险
  - 在问题变得“严重”之前发现问题区域，
  - 调整工作流程或任务，
  - 评估项目团队对软件工作产品质量的控制能力。

- **过程测量**
  - 我们间接地测量软件过程的功效。
    - 也就是说，我们根据可以从过程中得出的结果得出一套度量标准。
    - 结果包括
      - 软件发布前发现的错误度量
      - 交付给最终用户并由最终用户报告的缺陷
      - 交付的工作产品(生产力)
      - 人类的精力
      - 日历时间消耗
      - 计划的一致性
      - 其他措施。
  - 我们还通过测量特定软件工程任务的特征来得出过程度量。

- **流程度量准则**
  - 在解释度量数据时使用常识和组织敏感性。
  - 定期向收集度量和指标的个人和团队提供反馈。
  - *不要用指标来评估个人。*
  - 与从业者和团队合作，设定明确的目标和用于实现目标的指标。
  - *永远不要用指标来威胁个人或团队。*
  - 指示问题区域的指标数据不应该被认为是“负面的”。这些数据仅仅是流程改进的一个指标。
  - 不要执着于单一指标而排斥其他重要指标。

- 软件过程改进



- **流程度量**
  - 质量相关
    - 关注工作产品和可交付成果的质量
  - Productivity-related
    - 与所花费的精力相关的工作产品的生产
  - 统计SQA数据
    - 错误分类和分析
  - 缺陷的去除效率
    - 错误从过程活动到活动的传播
  - 重用的数据
    - 生产的组件数量及其可重用性程度

- **流程度量**
  - 通过进行必要的调整来最小化开发进度，以避免延迟并减轻潜在的问题和风险
  - 用于持续评估产品质量，必要时，修改技术方法以提高质量。
  - 每个项目都应该衡量:
    - 投入——衡量完成工作所需的资源(例如，人员、工具)。
    - 输出——对软件工程过程中产生的可交付成果或工作产品的度量。
    - 结果——表明可交付成果有效性的度量。

- **典型项目指标**

  - 每个软件工程任务的工作量/时间
  - 每个评审小时发现的错误
  - 计划的和实际的里程碑日期
  - 变化(数量)及其特征
  - 软件工程任务上的工作量分配

- **度量准则**
  - 在解释指标数据时使用常识和组织敏感性。
  - 定期向收集度量和指标的个人和团队提供反馈。
  - 不要用指标来评估个人。
  - 与从业者和团队合作，设定明确的目标和用于实现目标的指标。
  - 永远不要用指标来威胁个人或团队。
  - 指示问题区域的指标数据不应该被认为是“负面的”。这些数据仅仅是流程改进的一个指标。
  - 不要执着于单一指标而排斥其他重要指标。

- **典型Size-Oriented指标**
  - 每KLOC(千行代码)的错误
  - 每个代码缺陷
  - 每美元LOC
  - 每KLOC的文档页数
  - 每person-month错误
  - 每小时评审错误数
  - LOC / person-month
  - 每页文档$

- **典型的面向功能的指标**
  - FP(上千行代码)的错误数
  - 缺陷/ FP
  - 美元/ FP
  - 每FP的文档页
  - FP / person-month

- 比较LOC和FP

Programming Language	LOC per Function point avg.	median	low	high
Ada	154	-	104	205
Assembler	337	315	91	694
C	162	109	33	704
C++	66	53	29	178
COBOL	77	77	14	400
Java	63	53	77	-
JavaScript	58	63	42	75
Perl	60	-	-	-
PL/1	78	67	22	263
Powerbuilder	32	31	11	105
SAS	40	41	33	49
Smalltalk	26	19	10	55
SQL	40	37	7	110
Visual Basic	47	42	16	158

QSM开发的代表值

- **为什么选择FP?**
  - 编程语言独立
  - 使用在软件开发早期就确定的容易计数的特性
  - 难道不“惩罚”那些使用更少LOC的创造性(短)实现而不是其他更笨拙的版本吗
  - 使可重用组件的影响更容易衡量

- **面向对象度量**
  - 场景脚本(用例)数量
  - 支持类的数量(需要实现系统，但与问题域没有直接关系)
  - 每个关键类(分析类)的平均支持类数量
  - 子系统数量(支持系统最终用户可见的功能的类的集合)

- **WebApp项目指标**
  - 静态网页的数量(最终用户无法控制页面上显示的内容)
  - 动态网页数量(最终用户的行为导致页面上显示的自定义内容)
  - 页面内部链接的数量(页面内部链接是指在WebApp中提供一个指向其他网页的超链接的指针)
  - 持久化数据对象的数量
  - 对外接口系统的数量
  - 静态内容对象的数量
  - 动态内容对象数量
  - 可执行函数数量

- **测量质量**
  - *正确性——程序按照规范运行的程度*
  - *可维护性——程序可更改的程度*
  - *完整性——程序不受外部攻击的程度*
  - *可用性——程序易于使用的程度*

- 缺陷的去除效率

$$Dre = e /(e + d)$$

地点:

E为软件交付给最终用户前发现的错误数

D是交付后发现的缺陷数量。

- **小型组织的度量标准**
  - 从发出请求到完成评估所经过的时间(小时或天)，t队列。
  - 执行评估的工作量(人小时)，W eval。
  - 从完成评估到将变更顺序分配给人员所需的时间(小时或天)。
  - 进行变更所需的工作量(工时)，W变更。
  - 做出改变所需的时间(小时或天)，t改变。
  - 变更工作中发现的错误，变更。
  - 变更发布给客户后未发现的缺陷，D变更。

- **建立指标计划**
  - 确定你的商业目标。
  - 明确你想知道或学习什么。
  - 确定您的子目标。
  - 确定与你的子目标相关的实体和属性。
  - 将你的度量目标形式化。
  - 确定可量化的问题和相关的指标，您将使用它们来帮助您实现您的度量目标。
  - 确定您将收集的数据元素，以构建有助于回答您的问题的指标。
  - 定义要使用的指标，并使这些定义具有可操作性。
  - 明确你将采取什么行动来实施这些措施。
  - 为实施这些措施准备一个计划。