Cooper
定位和价值

编制日期：2015-2
作者：Abner

更改履历

版本号	更改时间	更改的 图表和章节号	状态	更改简 要描述	更改申请 编号	更改人	批准人

注：状态可以为 N-新建、A-增加、M-更改、D-删除。

目　　录

定位与价值

1. 定位

Cooper 是一款软件研发工具，主要用于帮助软件架构师在构建软件系统时跟踪、评价和改善软件结构的质量。

2. 价值

软件结构即软件架构深刻影响着软件质量，好还是坏的软件结构在业界已经沉淀了很多资产，这些资产主要是以设计原则的形式存在。设计原则本身比较抽象，多数架构师在具体设计和实施一个软件系统时对设计原则的理解多有不同，在多种因素的影响下很难去重视这些原则，造成软件结构质量低下，维护困难，难于适应业务的变化，生命周期缩短。Cooper 通过将这些设计原则的要质模型化，并通过对程序代码的分析，给出软件结构质量的分数来对软件结构进行跟踪和评价。在此过程中，Cooper 可以发现程序结构中的问题，并给出改进建议，用户可以利用工具虚拟执行这些对结构的调整，通过对比前后的分数来促进架构师对软件结构进行重构，改善其质量。具体来说 Cooper 的价值表现在六个方面：

- 定量的评价模型

  随着软件系统承担的职责越来越多，其本身的规模越来越大，在设计软件结构时最大的办法就是模块化（组件化），基于模块化的重要设计原则就是高内聚、低耦合。另外，为了能够使模块变化时影响的范围可控，模块要有好的封装性，模块间的关系是单向（无环依赖原则）、并且由不稳定的组件指向稳定的组件（稳定依赖原则）。在构建领域组件内部业务逻辑的时候，可以通过抽象手段来为未来变化的业务预留扩展（稳定抽象等价原则）。

  Cooper 将以上原则转化为对软件结构四个考核方面，分别是：组件的关系合理性、内聚性、封装性，以及抽象程度合理性。每个方面得分 25 分，总分为 100 分。当分数在 80 分以上时为优，70-80 分为良，60-70 分为中，60 分以下为劣。

  这四个考核方面主要是由组件身上的多种指标计算得到（组件的划分可由用户指定，也可以由系统根据多种算法自动识别），而组件的指标由其包含的类身上的指标计算得到，而类身上的指标又由其类本身的特征以及它们之间的关系计算得到，总结一句就是系统的结构质量是由程序代码编写的具体特征计算得到。

- 结构问题可视化/早发现

  当 Cooper 根据评价模型评价软件结构的质量时，必然会发现在结构上做得不合理的地方，系统会将这些问题记录下来，并以图形的形式显示出来。

  在类列表中可以看到导致组件内聚性、封装性低的是那些类，那些类关系；

  在组件关系页面中可以通过颜色的差异看到存在问题的组件关系；

  等等；

用户可以通过创建可以被持久化的命令组和命令，以及命令关联的组件划分方式，对正在研发中的软件系统进行分析，从而更早发现系统设计以及研发中的问题，尽早进行调整和改进。

- 辅助识别可复用的领域业务组件

  在组件的指标中有一个是计算组件的稳定性的，通过该指标以及配合对抽象性指标的观察，可以在已经开发完成的系统中发现可复用的资产，并将其从原系统中独立出来，改进其封装性，成为构建其他相似系统的部件。

- 自动生成结构调整建议

  Cooper 以结构问题为线索，从问题比较严重的结构入手，自动计算出改进建议，给出优先级。用户可以通过待做事项页面查看这些调整建议，以及给出建议的依据。

  目前 Cooper 生成结构建议的依据主要有：

  ✓ 两个组件存在彼此依赖，应该避免强度比较弱的依赖线

  ✓ 违反稳定依赖原则

  ✓ 该关系在循环依赖链上表现为最弱的关系

  ✓ 违反高内聚低耦合原则

  ✓ 违反稳定抽象等价原则

  ✓ 蝶形对象不应该在依赖其他组件

  ✓ 作为易分对象又被其他组件依赖，需要拆分

- 结构调整虚拟执行

  针对结构调整的建议，以及用户在观察分析结果中发现的对结构进行调整的想法，可以通过 Cooper 提供的结构调整虚拟执行功能进行虚拟调整，Cooper 可以计算调整后的结构分数，并可以与调整前的结构进行对比，观察系统级、组件级、类级别的结构数值变化，判断这种调整是否是好的。用户也可以在调整后的结构上继续调整，Cooper 会保存一个调整的历史，随时调整到某一个状态，对其进行对比。

  通过这种虚拟执行功能可以提高用户进行架构重构的效率，避免不当的架构重构。

- 关注程序细节内容的分析器

  除了以上提供的价值外，Cooper 还提供了大量的分析器（目前 30 个），它们可以从不同的角度对程序结构进行有针对性的分析，加深你对程序结构的认识。目前的分析器列表：

  ✓ 蝶形对象检测

  ✓ 易分对象检测

- ✓ 组件接口检查
- ✓ DIP 检查
- ✓ LSP 检查
- ✓ 包含结构分析
- ✓ 继承结构分析
- ✓ 识别有移动倾向的类
- ✓ 识别有移动倾向的方法
- ✓ 浏览类关系
- ✓ 浏览类内外关系比例
- ✓ 关键字搜索
- ✓ 覆盖检查
- ✓ 识别重复类
- ✓ 识别共同调用的方法集合
- ✓ 搜索不合理的事务注解
- ✓ 搜索存在状态的 Action
- ✓ 搜索存在状态的 Service
- ✓ Callback 识别
- ✓ 设计模式识别