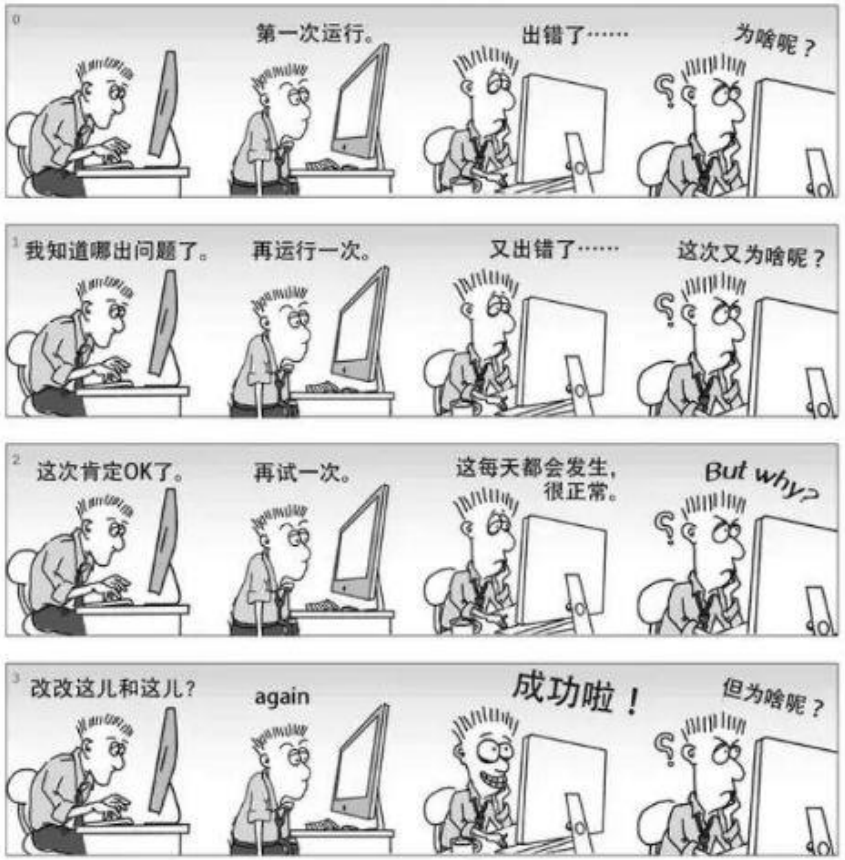

软件分析

# 缺陷修复技术

熊英飞
北京大学
2018

# 程序员的人生

- 就是写Bug和修Bug交织在一起的悲歌



# 到底花了多少时间修Bug?


- 软件维护35.6%的时间是在修Bug[1]
  - 软件维护成本通常认为占软件成本的90%
- 开发人员花在修复上的时间占全部开发时间一半左右[2]
- 开发团队可能没有足够资源修复所有缺陷[3]
- 软件在包含已知缺陷的情况下发布[4]

[1] B. P. Lientz, E. B. Swanson, and G. E. Tompkins, “Characteristics of application software maintenance,” Commun. ACM, vol. 21, no. 6, pp. 466–471, 1978
[2] Britton et al. Quantify the time and cost saved using reversible debuggers. Cambridge report, 2013
[3] J. Anvik, L. Hiew, and G. C. Murphy, “Coping with an open bug repository,” eXchange, 2005, pp. 35–39
[4] B. Liblit, A. Aiken, A. X. Zheng, and M. I. Jordan, “Bug isolation via remote program sampling,” in PLDI, 2003, pp. 141–154

# 缺陷自动修复的形式


- 输入：一个程序和其正确性约束，并且程序不满足正确性约束
- 输出：一个补丁，可以使程序满足约束

研究和实践中考虑最广泛的正确性约束——
软件项目中的测试

# 缺陷自动修复的作用


- Yida Tao, Jindae Kim, Sunghun Kim, Chang Xu: Automatically generated patches as debugging aids: a human study. SIGSOFT FSE 2014: 64-74
  - 当程序员有高质量的补丁做辅助的时候，修复正确率大幅提高，修复时间小幅减少
  - 修复正确率大幅提高=>未来修复时间大幅减少

# 缺陷自动修复的学术价值


- “自动程序生成是计算机领域最核心的问题。”
  - Amir Pnueli, 图灵奖获得者
  - On the synthesis of a reactive module, POPL 1989
- “程序缺陷修复和自动程序生成是等价问题。”
  - 林惠民院士，雁栖湖会议2018
- 给定规约，给定空白程序，如果能修复空白程序相对规约的缺陷，我们就针对规约自动生成了程序

# 缺陷修复重要质量指标


- 正确率：产生的补丁中有多少是正确的
  - 决定技术是否可用
- 召回率：在所有的缺陷中有多少是能够修复的
  - 决定技术的应用效果
- 修复效率：每个缺陷要花多少时间修复
  - 决定技术的应用场景

# 发展历史-史前阶段

- 时间：-2009
- 修复一些特定类型的缺陷
  - 演化缺陷
- 修复一些特定类型的程序或软件制品
  - 布尔程序
  - 软件模型

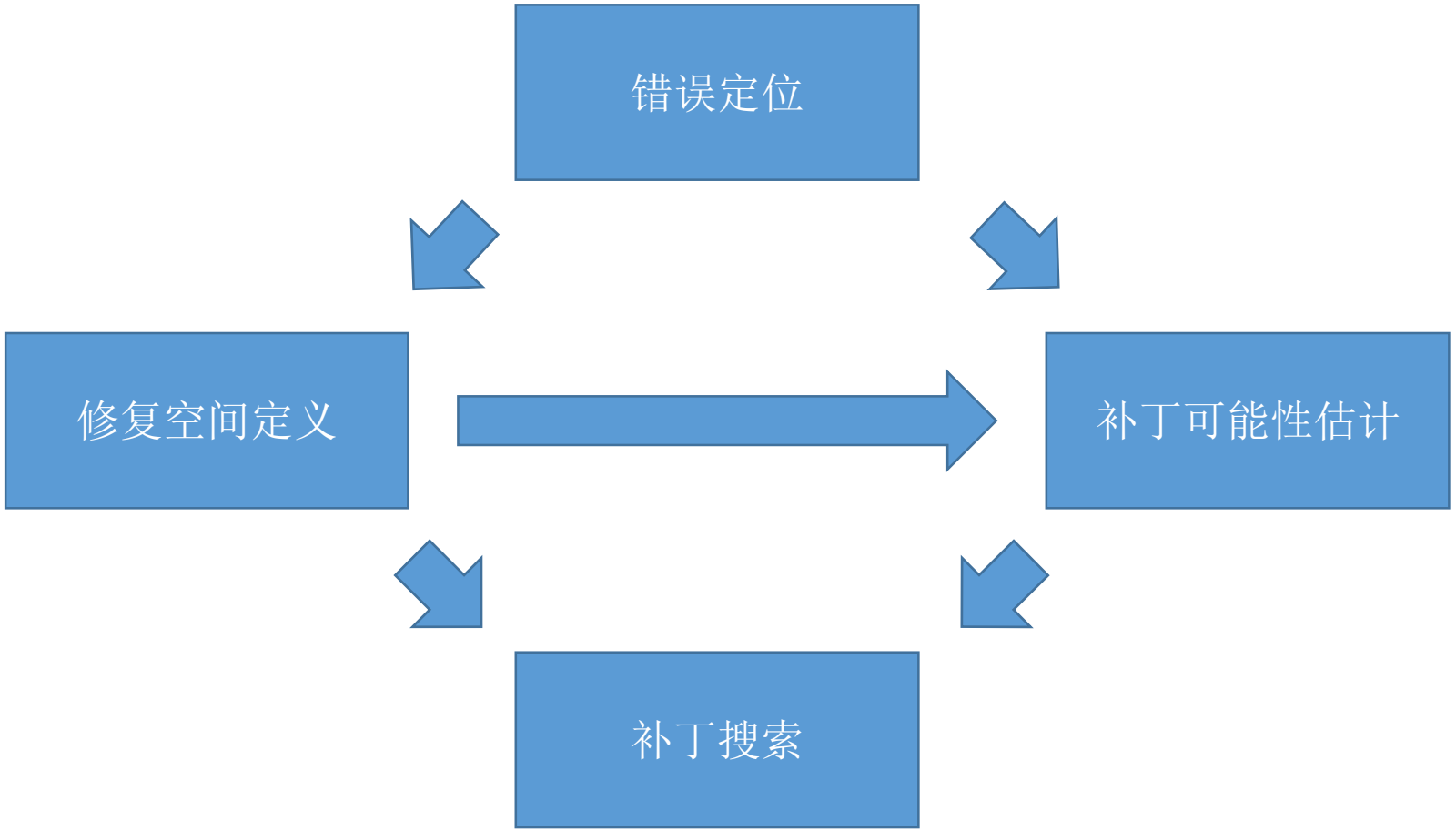
# 发展历史-GenProg时代


- Automatically finding patches using genetic programming.
  - Westley Weimer, ThanhVu Nguyen, Claire Le Goues, Stephanie Forrest. ICSE 2009: 364-374
  - 基本思路：天下程序一大抄
  - 随机从别的地方复制语句替换/插入到当前位置，或删除当前语句
  - 用遗传算法组合这些基本操作
- A systematic study of automated program repair: Fixing 55 out of 105 bugs for $8 each.
  - Claire Le Goues, Michael Dewey-Vogt, Stephanie Forrest, Westley Weimer. ICSE 2012: 3-13
  - 105个大型程序上的缺陷
  - 修复了一半
- 大量后续工作： AutoFix, RSRepair, NOPOL, relifix, SemFix, DirectFix, PAR…

# 发展历史-后GenProg时代


- An analysis of patch plausibility and correctness for generate-and-validate patch generation systems.
  - Zichao Qi, Fan Long, Sara Achour, Martin C. Rinard. ISSTA 2015
  - GenProg修复的55个缺陷中只有2个是正确的
  - 通过测试≠完整修复
- 后期修复技术将正确率作为重要指标
  - 通过和程序员的修复对比，等价的算作正确的
  - 大量技术致力于提供高正确率的修复：Prophet, Angelix, HDRepair, ACS, Anti-Pattern, Elixir, JAID, CapGen, Genesis…
- Precise condition synthesis for program repair
  - Yingfei Xiong, Jie Wang, Runfa Yan, Jiachen Zhang, Shi Han, Gang Huang, Lu Zhang. ICSE 2017: 416-426
  - 采用数据驱动的方式修复缺陷
  - 正确率提高到70%以上

# 现代缺陷修复基本模块



# 典型缺陷修复技术: GenProg


**错误定位**

基于频谱的错误定位

**修复空间定义**

以下三种操作的组合：

- 在错误语句前插入一条同项目任意语句
- 将错误语句替换成同项目任意语句
- 删除任意语句

**补丁可能性估计**

通过测试越多越有可能

**补丁搜索**

遗传算法

# 典型缺陷修复技术: ACS


**错误定位**

基于频谱的错误定位+
Predicate Switching

**修复空间定义**

生成形如x ? v的条件：

- x是变量
- ?是二元谓词，如>, <=, instanceof
- v是一个常量

**补丁可能性估计**

- 用启发式规则对变量x的可能性排序
- 用GitHub上的条件数据对?和v的可能性排序

**补丁搜索**

按可能性依次验证

# 错误定位


- 大多数基于已有错误定位技术

- 通常采用基于频谱的错误定位
  - 被失败的测试用例执行的程序元素，更有可能有错误
  - 被成功的测试用例执行的程序元素，更有可能没错误
- 针对特定缺陷类别也采用StackTrace定位、Predicate Switching等技术
- 部分方法采用了自己的定位技术
  - AutoFix和JAID通过分析不变式来定位

# 修复空间定义


- 基于程序变换模板的修复空间定义(PAR)
  - 交换函数变量
  - 更改函数名字
- 基于语法的修复空间定义(SemFix, Angelix, ACS)
  - 替换错误条件表达式为x ? v形式的表达式
- 基于代码比较的修复空间定义(GenProg, ssFix)
- 基于统计出的常见操作定义修复空间(CapGen, SimFix)
- 基于不变式导出(ClearView, JAID)

# 补丁可能性估计


- 基于固定规则排序
  - 改动越小越好[DirectFix, Angelix]
  - 对运行时行为影响越小越好[Qlose]
  - 对通过测试影响小，对失败测试影响大[Xiong-ICSE18]
  - 自定义排序规则[S3]
- 基于统计/机器学习排序
  - 对补丁空间的补丁整体排序[Prophet, Elixir]
  - 对补丁的各部分分别排序，然后整合[ACS, CapGen]
- 整合错误定位的排序
  - 用错误定位的怀疑度修正补丁的可能性[Prophet, Angelix]
  - 根据不变式的可能性排序[AutoFix, JAID]

# 补丁搜索


- 按排序顺序验证[Prophet, Elixir]
- 按补丁各部分的排序贪心查找[ACS]
- 采用启发式搜索
  - 遗传算法[GenProg]
- 加快验证的速度
  - 消除编译的冗余[Mao-ICSME13]
  - 消除运行时的容易[Xiong-ISSTA17]

# 最新技术能达到的效果


- CapGen
  - 在Defects4J的224个缺陷上做验证
  - 正确率：84%
  - 召回率：9.3%
- ACS+ICSE18
  - 在Defects4J的224个缺陷上做验证
  - 正确率：85%
  - 召回率：7.6%
- SimFix
  - 在Defects4J的357个缺陷上做验证
  - 正确率：60.7%
  - 召回率：9.52%
- 修复时间：通常设置为0.5-3小时不等

# 特定类型的缺陷修复


- 内存泄露修复
- 死锁修复
- 竞争修复
- 构建失败修复
- 安卓崩溃修复
- 作业修复

- 通过设计特定的修复空间和专门的搜索算法，通常能达到较好的修复效果

# 工业界应用现状

- Google
  - TriCoder[ICSE15]——支持修复的缺陷检查工具
- Facebook
  - SapFix, Getafix——基于历史学习修复空间
- Fujitsu
  - Elixir[ASE15]——基于机器学习的补丁排序

- 国内应用现状留给各位嘉宾

# 面向未来：缺陷自动修复的可能性


- Jiajun Jiang, Yingfei Xiong: Can defects be fixed with weak test suites? An analysis of 50 defects from Defects4J. CoRR abs/1705.04149 (2017)
  - 程序员在不了解软件需求的情况下手动修复Defects4J上的50个缺陷
  - 正确率和召回率均超过了90%

- 能否深刻了解程序员是如何修复缺陷或许是未来关键

# 缺陷修复社区资源


- 社区网站：http://program-repair.org/

- 缺陷修复综述：
  - Automatic software repair: a survey. Luca Gazzola, Daniela Micucci, Leonardo Mariani. ICSE 2018: 1219
  - Automatic Software Repair: A Bibliography. Martin Monperrus. ACM Comput. Surv. 51(1): 17:1-17:24 (2018)

- 微信缺陷修复群

# 缺陷修复小结


- 学术价值高，工业需求强
- 领域刚刚起步，未来充满希望
- 学术界和工业界共同努力，将程序员从修Bug的繁重劳动中解放出来