软件测试的几个研究方向

主要内容

1. 测试用例演化
2. 基于故障模型的软件测试
3. 软件测试中的数据挖掘
4. 软件缺陷预测技术
5. 自动驾驶汽车测试
6. 小结

主要内容

1. **测试用例演化**
2. 基于故障模型的软件测试
3. 软件测试中的数据挖掘
4. 软件缺陷预测技术
5. 自动驾驶汽车测试
6. 小结

1. 测试用例演化

- **测试用例**是为某个特殊目标而编制的一组测试输入、执行条件以及预期结果，以便测试某个程序路径或核实是否满足某个特定需求。

输入	期望结果
输入正确的用户名，错误的密码，点击“登录”按钮	提示“密码不正确，请重新输入”

- 由于功能增加、性能调优、软件重构、错误修复等原因，软件通常处于动态演化。
  - 如：登录时增加了验证码输入功能，测试用例需要随之改变

输入	期望结果
输入正确的用户名，错误的密码，**输入验证码：图片中的数字，**点击“登录”按钮	提示“密码不正确，请重新输入”

1. 测试用例演化

- 软件演化带来的挑战:

(1)新的测试用例不断产生，会积累大量冗余测试用例。

解决：测试用例选择

(2)软件演化导致部分测试用例不可用，直接丢弃这些测试用例将降低错误检测能力。

解决：测试用例修复

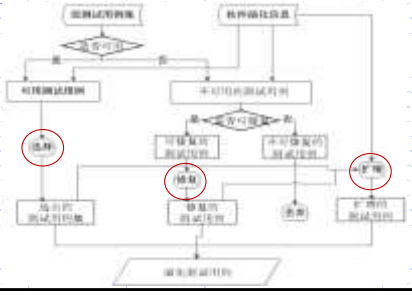
(3)软件模块的增加和修改，已有测试用例不能完全覆盖这些模块。

解决：测试用例扩增

1. 测试用例演化

- **测试用例选择**是从原有测试用例集中挑选部分测试用例并尽量满足新版本测试需求。
- **测试用例修复**是指对旧版本程序测试用例集中的不可用测试用例进行修复，使得修复后的测试用例能够在新版本程序上执行.
- **测试用例集扩增**是指根据新版本程序信息和已有测试用例信息来生成新的测试用例,以覆盖新版本程序的修改部分和新增部分.
- 测试用例选择、测试用例修复和测试用例扩增之间**相互依赖和补充**.
- 从技术成本上看,测试用例集扩增成本高,测试用例修复次之,测试用例选择成本低。

1. 测试用例演化



7

1. 测试用例演化

研究现状

- **测试用例选择**目前的技术较为成熟；
- **测试用例修复**对于 GUI 程序较为重要,可以解决 GUI 程序中因简单控件修改而导致大量测试用例不可用的问题；
- **测试用例扩增**的研究目前较少。

8

主要内容

1. 测试用例演化
2. **基于故障模型的软件测试**
3. 软件测试中的数据挖掘
4. 软件缺陷预测技术
5. 自动驾驶汽车测试
6. 小结

9

2. 基于故障模型的软件测试

- 属于静态测试技术
- 传统测试方法的局限性
  - **测试效果不确定**：每次测试时，能检测哪些故障是不确定的；
  - **难以检测小概率故障**：若存在故障的语句没有得到执行，则检测不到故障；
  - **某些故障用传统方法测试不出来**：如内存泄露等。

10

2. 基于故障模型的软件测试–步骤

(1)**计算检查点**（IP - Inspective Points）：通过分析源代码，计算出与所给的故障模型相匹配的程序代码。

检查点可以是一个语句或语句的一部分，也可以是多个语句。

计算检查点的过程

11

2. 基于故障模型的软件测试–步骤

(2) 判断检查点的性质：

- **故障检查点**：肯定存在一个故障；
- **正确的检查点**：肯定不是一个故障；
- **不能确定的检查点**：无法确定是故障或者是正确的。

- 大约只有30%的检查点可以自动判断，一般仍需要人工判断。
- 一个有经验的软件测试工程师8小时可以处理100~200个检查点。

12

主要内容

1. 测试用例演化
2. 基于故障模型的软件测试
3. 软件测试中的数据挖掘
4. 软件缺陷预测技术
5. 自动驾驶汽车测试
6. 小结

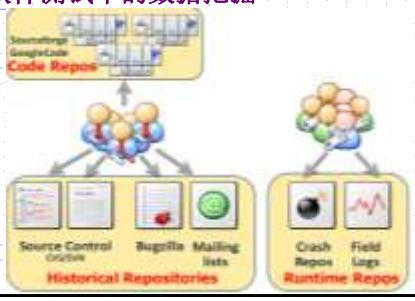
13

3. 软件测试中的数据挖掘

- 数据挖掘的通俗解释：数据挖掘旨在从大量数据中挖掘出未知且有用的知识。
- 只有通过挖掘，大数据的价值才得以体现， 挖掘对大数据的利用有着举足轻重的意义。
- 希望达到的目标：理解过去，预测未来，推荐最佳实践

14

3. 软件测试中的数据挖掘



15

3. 软件测试中的数据挖掘

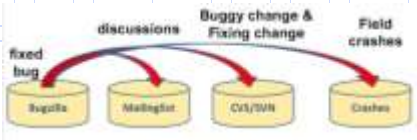
需要解决的问题

- 数据挖掘有两个基本问题：挖什么(what to mine)、怎么挖(how to mine)。
- “挖什么”决定从数据中抽取什么样的信息、统计什么样的规律，是在数据的收集、处理和挖掘的全过程中要考虑的问题
- “怎么挖”决定怎样具体进行抽取与统计，仅限于挖掘本身；是数据挖掘研究的核心
- “挖什么”在数据挖掘的应用中往往更为重要，因为它决定了挖掘结果的价值

16

3. 软件测试中的数据挖掘

挖什么？



New Bug Report
Estimate fix effort
Mark duplicates
Suggest experts and fix!

17

3. 软件测试中的数据挖掘—挖什么？

- 比如，通过挖掘项目的历史代码库、缺陷跟踪系统等，可以回答以下问题：

① 哪些程序员应该修复哪类缺陷?

② 如何减少冗余的测试用例？

③ 哪个时间提交的代码最容易出现错误?

发现：半夜12:00至凌晨4点之间提交的代码最容易产生错误；7AM至中午提交的代码产生的错误最少

④ 某类缺陷是属于哪个构件/模块内的？

⑤ 不同用户会报告重复的缺陷，如何找出重复报告的缺陷？

18

3. 软件测试中的数据挖掘—挖什么？

⑥ 缺陷报告中应该把哪些人加为抄送人？

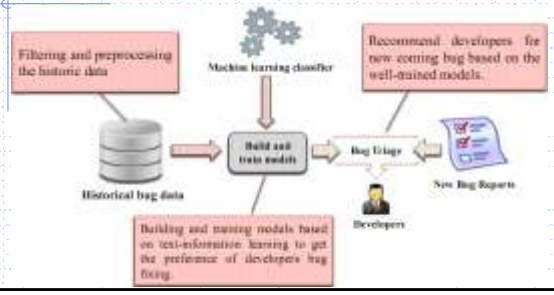
⑦ 对于新手，编码时为了完成某功能应该使用哪个API？这个API应该如何使用？这一点对于那些没有完善文档的API非常有用。

⑧ 已经使用了某些API的程序，其使用方法是否有潜在的错误？

⑨ 缺陷预测：哪些代码模块（或代码行）可能会有缺陷？可能会有多少缺陷？

⑩ .....

例子：缺陷应该交给谁负责修复？



主要内容

1. 测试用例演化
2. 基于故障模型的软件测试
3. 软件测试中的数据挖掘
4. 软件缺陷预测技术
5. 自动驾驶汽车测试
6. 小结

4. 软件缺陷预测技术

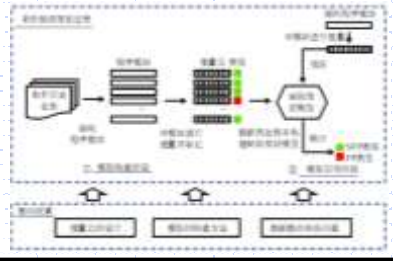
- **测试**技术在于**发现**缺陷，而**预测**技术则在于**预测还未发现的**缺陷.
- **软件缺陷预测技术**的目的在于预测软件系统的缺陷（没有发现但还可能存在的缺陷），以帮助确定在哪些地方应该重点测试，或决定系统是否可以交付使用。
- **静态预测技术**：基于缺陷相关的**度量数据**，对缺陷的数量或分布或倾向性进行预测的技术;
- **动态预测技术**：基于缺陷或者失效产生的**时间**，对系统缺陷随时间的分布进行预测的技术。

4.1 静态预测技术

- **基于软件规模的缺陷预测**：认为缺陷的多少取决于软件规模的大小
- **基于软件复杂度的缺陷预测**：随着软件规模的增加, 软件的复杂度也在逐步攀升, 人们开始意识到软件的缺陷不仅与规模有关, 还与软件的复杂度有关
- **基于软件的多种特征（或度量元），进行缺陷预测：**如：代码行数、操作符和操作数的数量、McCabe 圈复杂度、代码修改特征、开发人员情况、等等，这些都会与软件缺陷的出现有关

4.1 静态预测技术

基于数据挖掘技术，预测软件的缺陷倾向性或数量：



4. 1 静态预测技术

基于数据挖掘技术，预测软件的缺陷倾向性或数量：

- 挖掘软件历史仓库、设置与软件缺陷存在强相关性的度量元，是构建高质量缺陷预测模型的关键。
- 常用度量元：代码行数、操作符和操作数的数量、McCabe 圈复杂度、OO中的CK度量元、代码修改特征、开发人员情况、程序模块间依赖性、项目团队组织构架、等等。

25

4.1 静态预测技术

基于数据挖掘技术，预测软件的缺陷倾向性或数量：

模型构建方法：

- ✓ 基于机器学习的方法：
  - 缺陷倾向性：用分类方法，如： Logistic 回归、朴素贝叶斯、决策树等
  - 缺陷数或缺陷密度：回归分析法
- ✓ 基于缓存的方法

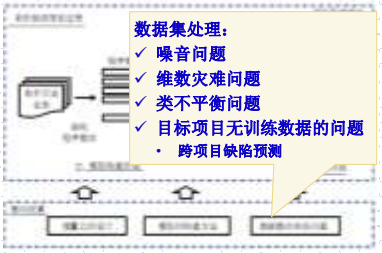
26

4.1 静态预测技术

基于数据挖掘技术，预测软件的缺陷倾向性或数量：

数据集处理：

- ✓ 噪音问题
- ✓ 维数灾难问题
- ✓ 类不平衡问题
- ✓ 目标项目无训练数据的问题
  - 跨项目缺陷预测



27

4.1 静态预测技术

项目内缺陷预测与跨项目缺陷预测：

28

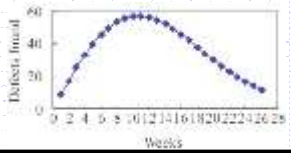
4.2 动态预测技术

- 动态缺陷预测技术是一种基于时间关系的软件缺陷预测技术；
- **基本思想**：基于经验研究和统计技术，发现软件缺陷随其生命周期或其中某些阶段的时间关系的分布规律；
- 最著名的有 Rayleigh 分布模型、指数分布模型和 S 曲线分布模型。
- Rayleigh模型是一个可以对软件开发全生命周期的缺陷分布进行预测的模型，是一种最常用的可靠性模型；
- 1984年，IBM联邦系统部报告，其缺陷的时间分布符合Rayleigh曲线；

29

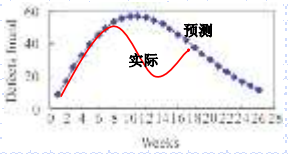
举例： Rayleigh分布模型

- 已知历史项目的缺陷密度为10.53/KLOC，新项目的总代码大约为100KLOC，要求新项目约减少 5%的缺陷；
- 根据Rayleigh模型，计算得到新项目期望的总缺陷数为1000；若新项目的周期为 26 周,再用Rayleigh模型,计算得到每周应发现缺陷数的时间分布,如下图：



30

举例： Rayleigh分布模型



- 当项目开发过程中实际发现的缺陷和分布曲线预计的缺陷出现**重大偏差**时，说明该过程出现异常,需要采取相应的纠正措施.

31

主要内容

1. 测试用例演化
2. 基于故障模型的软件测试
3. 软件测试中的数据挖掘
4. 软件缺陷预测技术
5. 自动驾驶汽车测试
6. 小结

32

5.1 多起自动驾驶汽车事故

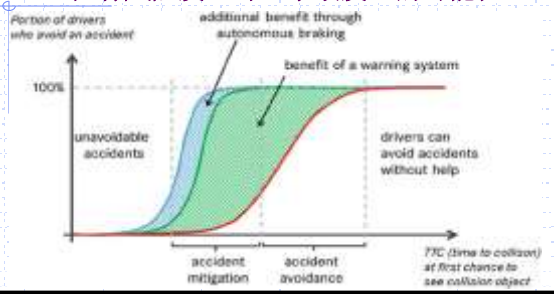
谷歌

特斯拉

33

5.2 自动驾驶汽车测试的挑战

5.3 在有风险发生时，事故发生的可能性



5.4 自动驾驶汽车运行时面临的安全风险

Category	Example: situation	Exposure	Severity
Continuous control	Keep vehicle in the lane, for any curve radius	3	3
Predictable end of automated driving	Planned exit of highway	3	1
Obstacles on the road	Sudden evasive manoeuver of vehicle ahead	2	3
Unexpected Infrastructure deficits	Lane marking not obvious	3	3
Traffic partner behaves "against the rules"	Vehicle ahead drives too fast	3	1
Weather-related challenges	Sudden glare from sun	3	3
Driver related misbehaviour	Driver not ready for take-over	2	2
Hardware problems	Complete sensor failure	1	3

5.5 测试——确保汽车在路上

用计算机产生各种道路类型和网络，看车能否保持在路上。

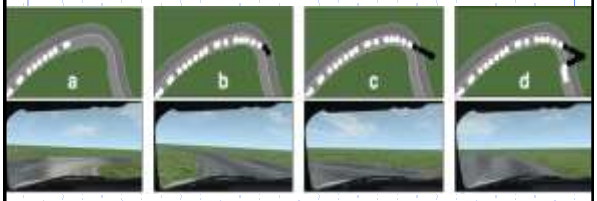
1.产生初始道路网络　　路段交叉　　路段合并

2.变换得到更复杂的道路网络

37

5.5 测试——确保汽车在路上

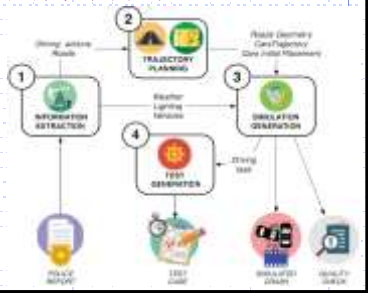
3. 汽车在产生的道路网络上模拟行使



38

5.6 在真实的事故场景中测试

- 在测试中，不可能模拟各种事故场景
- 根据实际的事故报告，重建事故场景，对自动驾驶汽车进行测试



主要内容

1. 测试用例演化
2. 基于故障模型的软件测试
3. 软件测试中的数据挖掘
4. 软件缺陷预测技术
5. 自动驾驶汽车测试
6. 小结

41

6. 小结

- "...testing can be a very effective way to show the presence of bugs, but is hopelessly inadequate for showing their absence. The only effective way to raise the confidence level of a program significantly is to give a convincing proof of its correctness."

– Edsger Dijkstra

1972年获得图灵奖, 1989年计算机科学教育杰出贡献奖, 2002年ACM PODC最具影响力论文奖。

6. 小结

软件形式化验证技术

- 是一类重要的静态分析方法，包括模型检测、定理证明、抽象解释等等。
- 以模型检测为例，其采用有限状态机描述系统的行为，通过遍历模型中所有可达的状态，检验其是否符合系统所期望的性质。
- 期望的性质可能是源代码级别的性质，如“程序不出现数组溢出错误”，也可能是高层级的性质，如“进程间不会有数据竞争”等。

43

THE END

Q&A

44