TiD 2017
质量竞争力大会

打造高质量、可复用的开源生态系统
——大规模集成开源代码的项目测试保障实践

丁国富 华为中央软件院
dingguofu@huawei.com

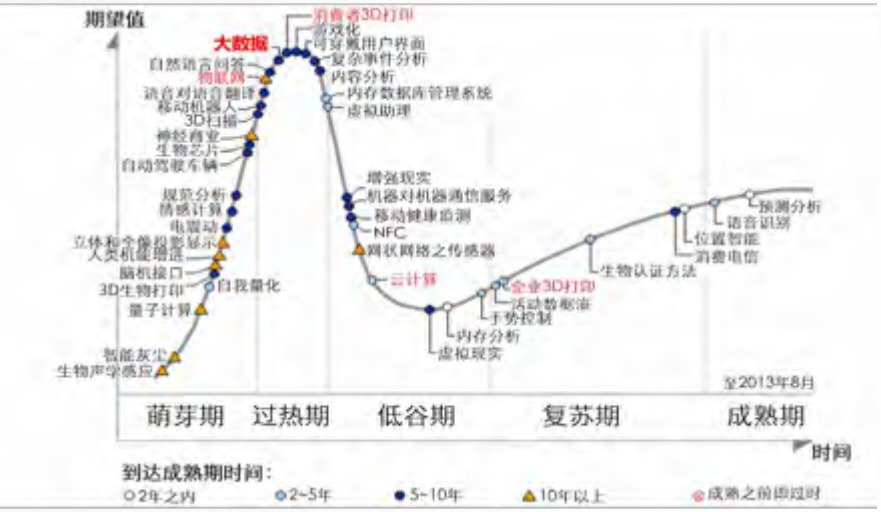
软件研发
SOFTWARE DEVELOPMENT

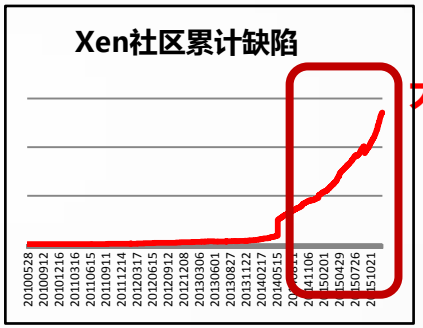
目录

背景
开源两大特征给测试带来的收益及挑战

大量集成社区代码的项目，爽了开发苦了测试！

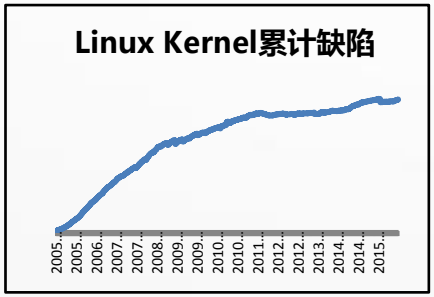
挑战1
集成大量新社区代码，意味着集成大量缺陷



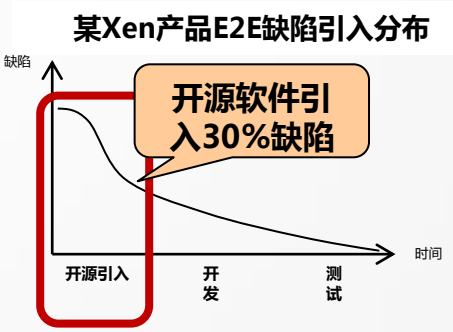


大量社区BUG将被引入！

新社区
发散



老社区
收敛



社区新技术从提出到能可靠商用，需要时间！

挑战2
测试设计过程被压缩，写/改一行需测百行

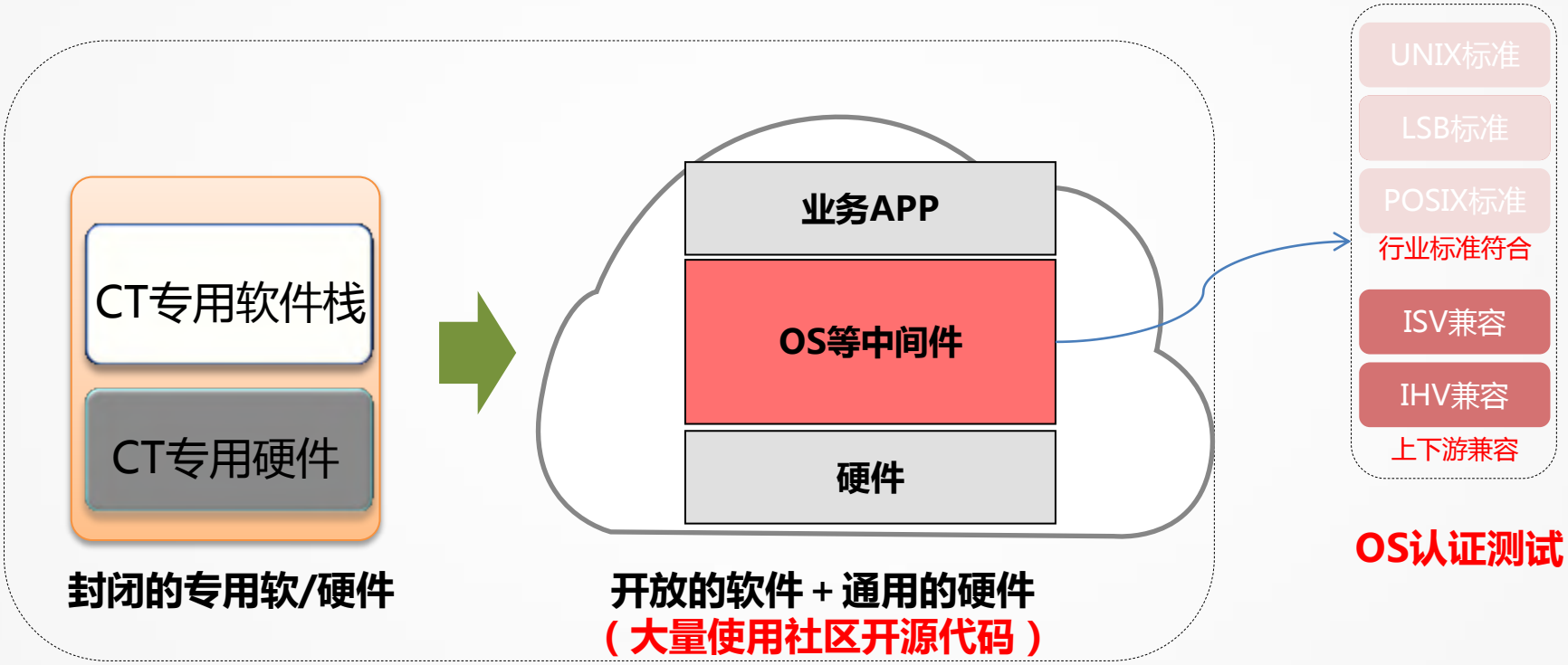
闭源和大规模集成开源代码的项目比较

	闭源项目		开源项目	开源测试挑战
	新项目	继承开发项目		
代码基线	无继承代码	**稳定**的继承代码	海量代码（**功能可用，可靠性待评估**）	**写一行测百行**，海量存量代码需测试覆盖； 社区缺少质量度量数据，测试评估难； 社区代码patch和**漏洞**，需要**快速精准验证**
	系统的度量及质量**评估，有正式的质量标准**		**缺乏**度量**分析**和正式的**风险评估**	
文档基线	**完整的关键文档**：需求说明书/架构设计说明书/特性分析说明书/设计方案/...		**少量简单**文档，如用户指南（构建、部署、功能说明）	**文档缺失**导致测试分析设计难
测试基线	**充裕的**测试基线构建**时间**	**系统的**用例基线/测试工具，**可继承**	**非系统性/少量的**测试基线，**重白盒轻黑盒**，无解决方案测试	补充商业级**可靠性及客户场景用例**

- 某活跃社区**patch 1000+/周，迭代周期1~2周，文档很少**

需要构建基于代码的快速测试分析及设计能力！

挑战3
IT/CT云化趋势下，需对上下游厂商及行业标准进行认证测试



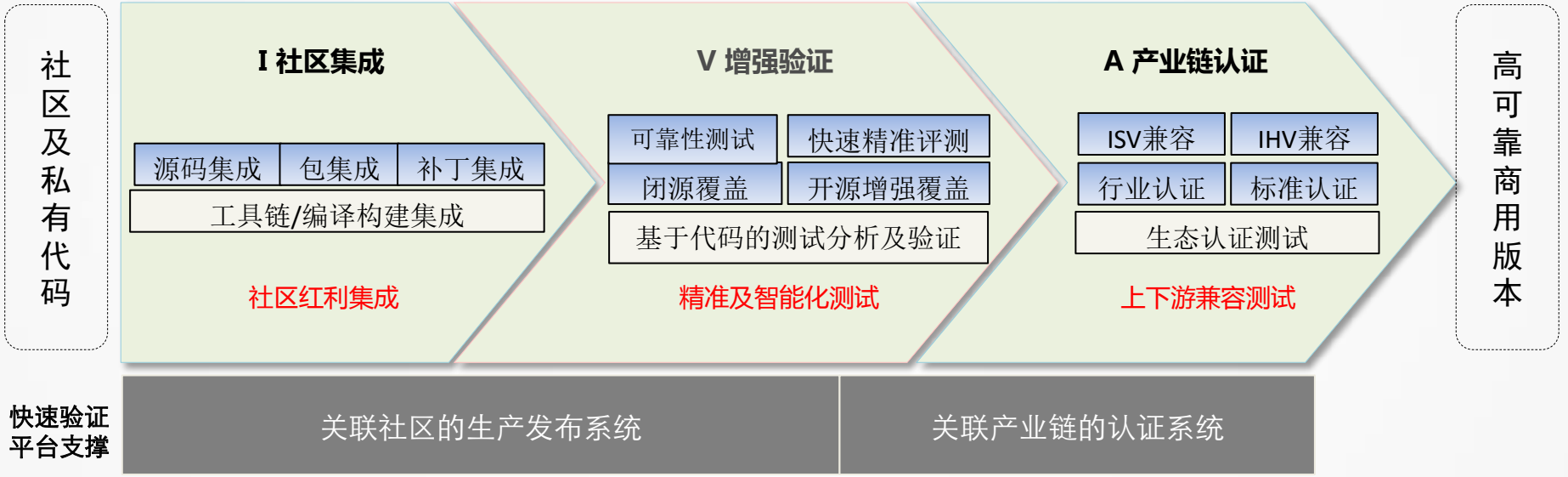
- 以OS为例，上下游ISV3000+，IHV5000+，还有各种行业标准；

生态伙伴的双向认证可减少兼容性测试的巨大投入！

目录

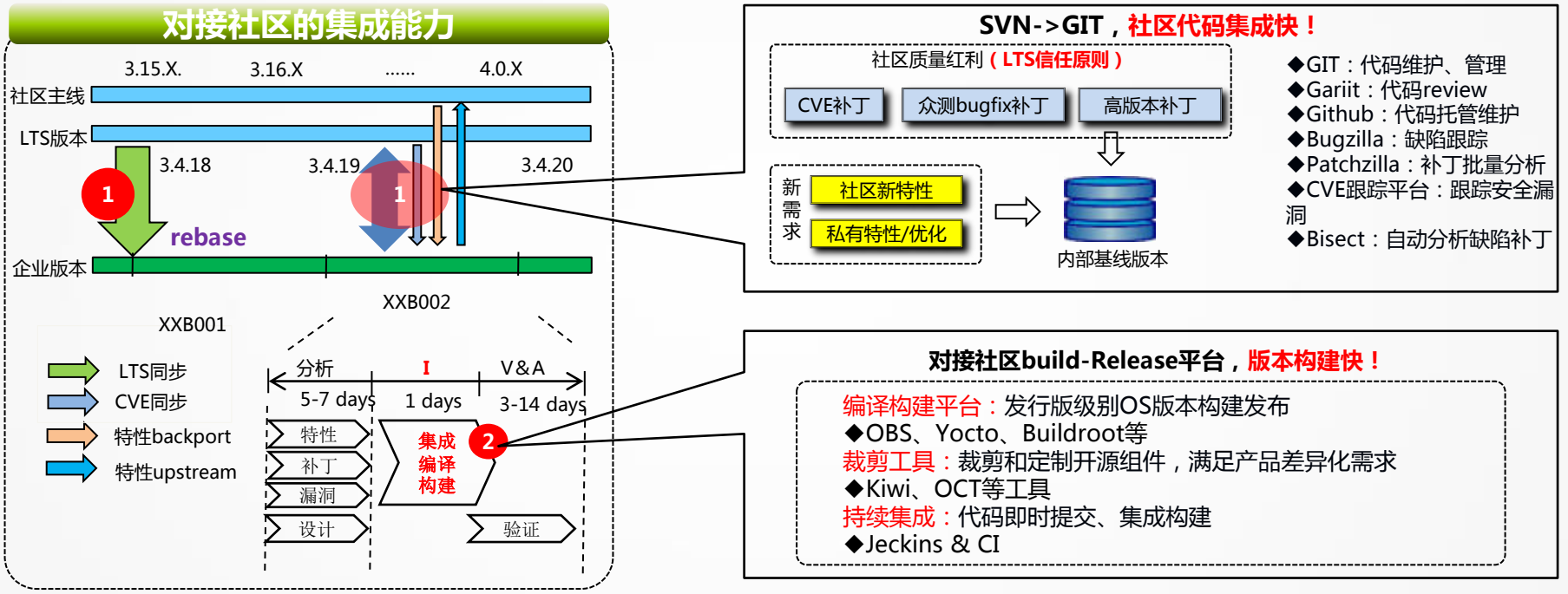
- 开源给测试带来的挑战
- 如何构建E2E开源测试能力
  - I 集成开源成果，享受社区质量红利
  - V 海量代码智能化覆盖增强测试
  - A 自动化认证测试

总体策略
建立对接社区的I&V&A能力（集成&验证&认证）



I(Integration)：集成社区质量保障红利，合入社区稳定版本补丁，快速构建基础质量版本；
V(Verification)：精准及智能化验证，开源及私有组件交互分析，基于代码覆盖的精准增强测试；
A（Authentication）：生态认证测试，产业链ISV/IHV兼容认证，行业标准认证测试；

I 集成社区质量红利
互动回馈，保持社区粘性，快速集成开源组件和补丁



某团队集成效果

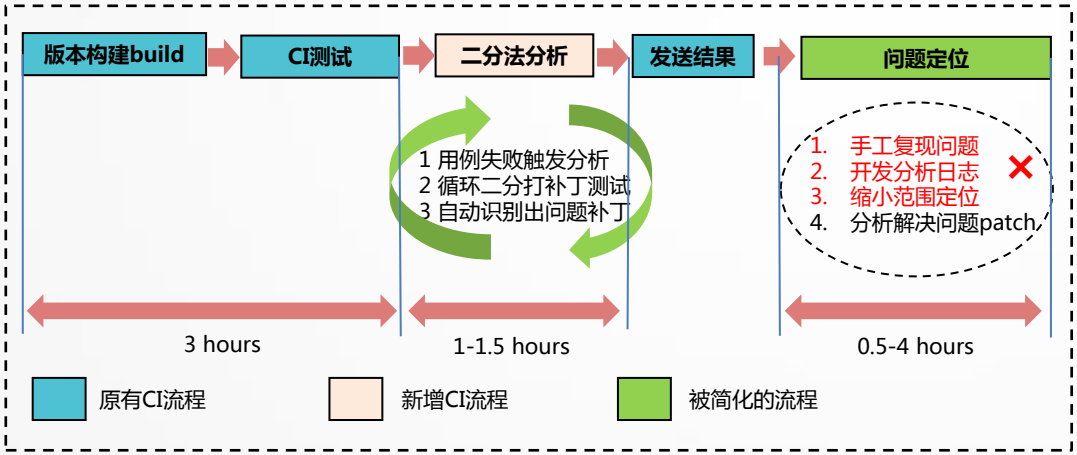
- **借力社区**：开发10人/测试4~5人参与社区互动，看护1K万行代码，继承社区数百工程师成果；
- **缺陷预防**：月同步bugfix补丁200+，CVE补丁50+，其他主线补丁100+，减少自投入；

基于信任频繁批量合入补丁，需构建问题补丁的快速甄别能力！

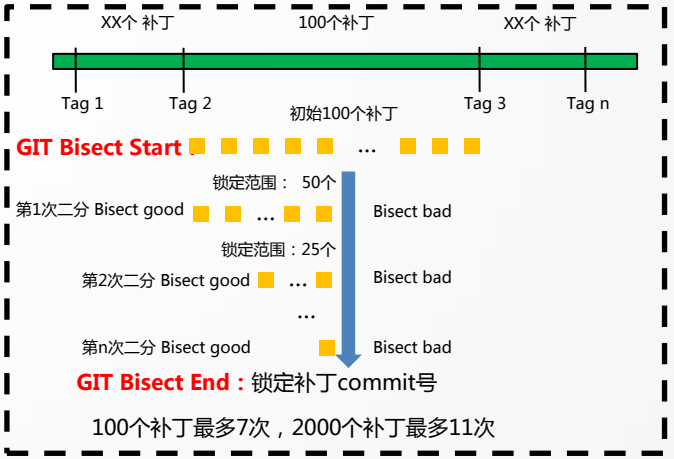
I 集成社区质量红利（续）
自动回滚批量补丁包，分析甄别问题补丁

基于信任合入的批量补丁，个别会与企业分支版本有配合问题，通过bisect二分批量操作，配合CI循环验证，可自动识别bad/good补丁，效率提升一倍以上！

CI集成二分工具自动识别引入问题补丁



二分法处理



要享受社区红利，就要采用社区研发模式及生产系统！

目录

- 开源给测试带来的挑战
- 如何构建E2E开源测试能力
  - I 集成开源成果，享受社区质量红利
  - V 海量代码智能化覆盖增强测试
  - A 自动化认证测试

V 商用级测试能力构建三部曲
对接社区完成项目选型、快速引进、补充增强

1:开源测试项目选型

目的：社区测试能力跟进

例： Linux测试项目跟踪

✓LTP（Linux Test Project），包括功能、性能等测试套120+；

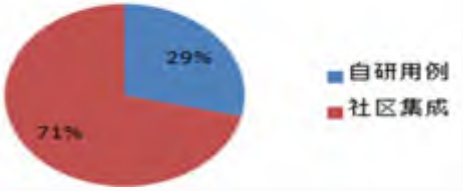
例：社区框架演进跟踪

✓AutoTest演进成Avocado框架 ，增强了对虚拟化验证插件的支撑；

2:开源用例引进并基线化

目的：快速构筑基础防线

开源用例复用>70%



开源测试套测试类型分布

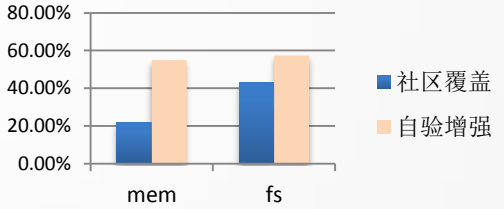
3:竞争力覆盖测试增强

目的：可靠性等测试覆盖增强

在LTP项目上补充覆盖

✓引入学术界测试能力；

✓提高关键特性覆盖，例如：



补齐社区测试能力缺口

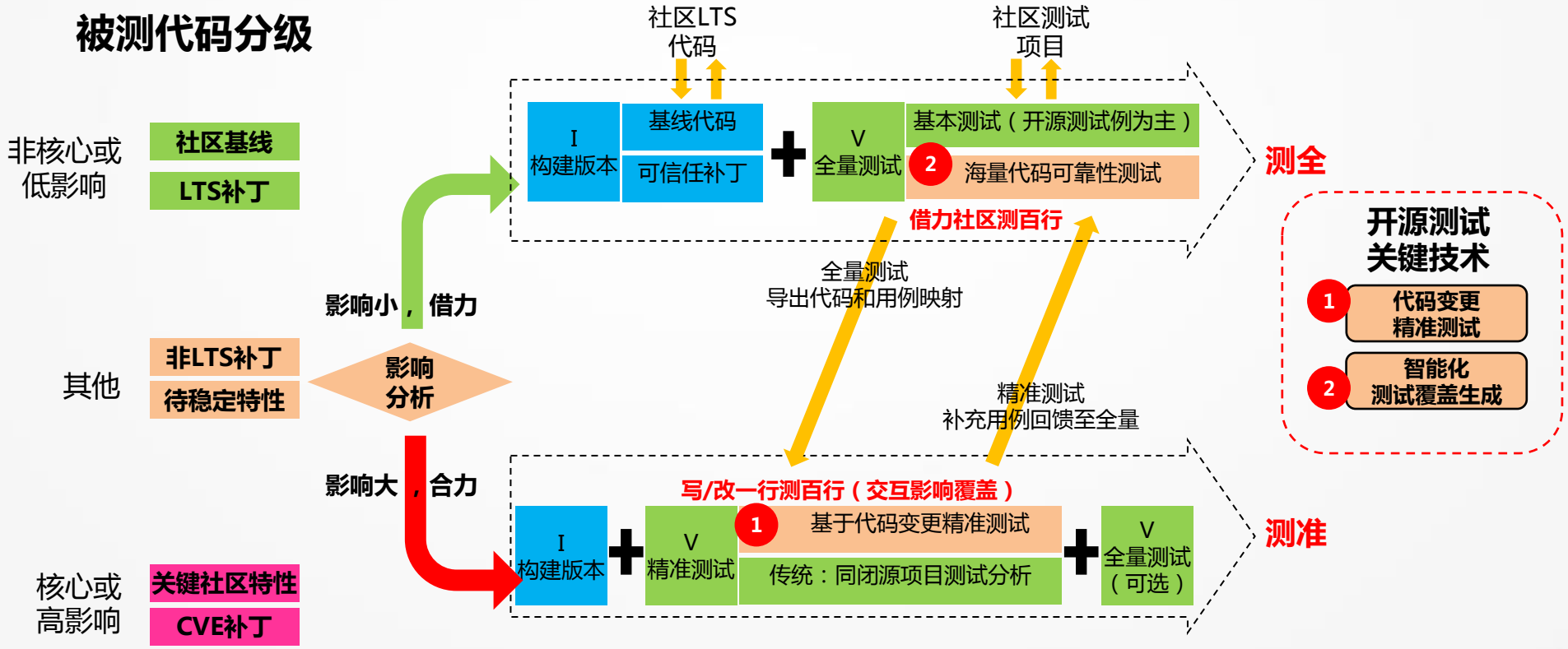
✓增强性能、可靠性、安全、公有云等专用场景用例xx个；

可靠性/安全等用例是核心竞争力，不一定被回馈社区，需各企业自行构建！

V 开源测试策略

基于代码核心度和变更影响，进行全量或精准的测试覆盖

测试能力及平台要求



继承开源的项目，需要基于代码的精准及智能化测试技术！

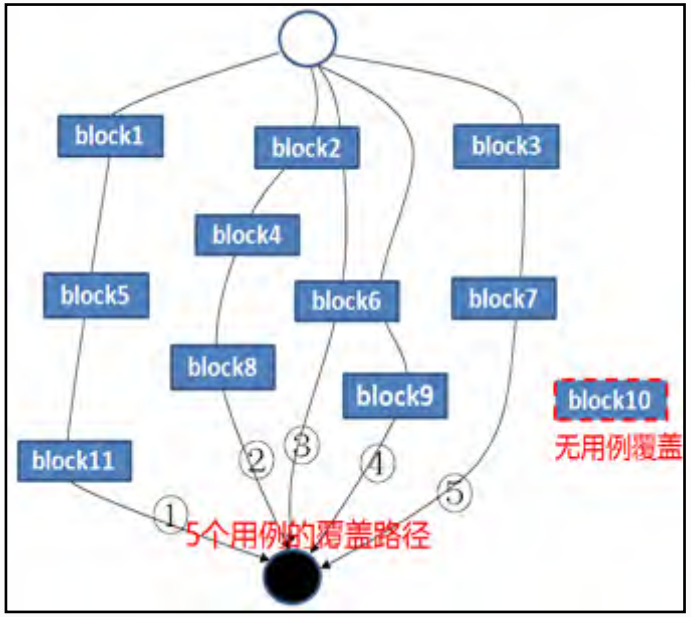
V 代码变更精准测试
自动筛选用例集，实现安全补丁或版本回归快速验证

分析难：改一测百，没文档

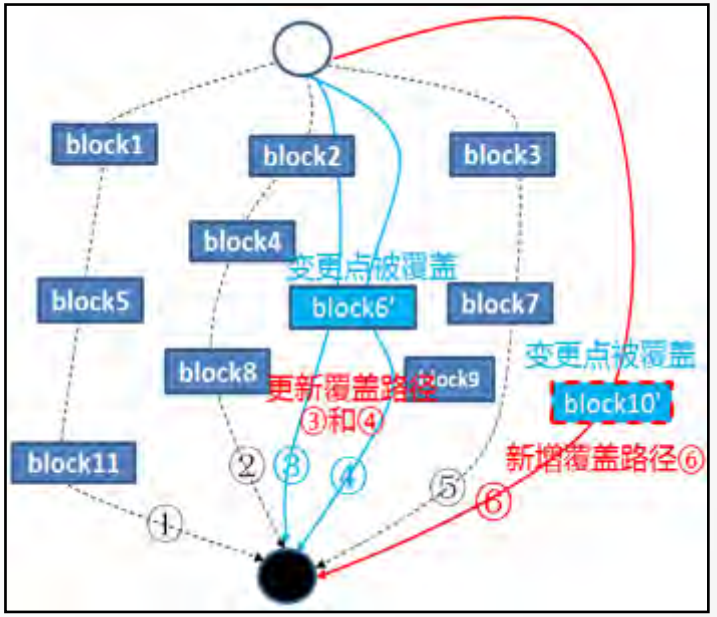
快：0day攻击补丁验证

迭代频：代码快速变更

测试面临挑战



步骤1：执行历史用例，使用GCOV收集覆盖率及用例代码映射关系



步骤2：基于代码变更，从映射库中精确选择关联用例集

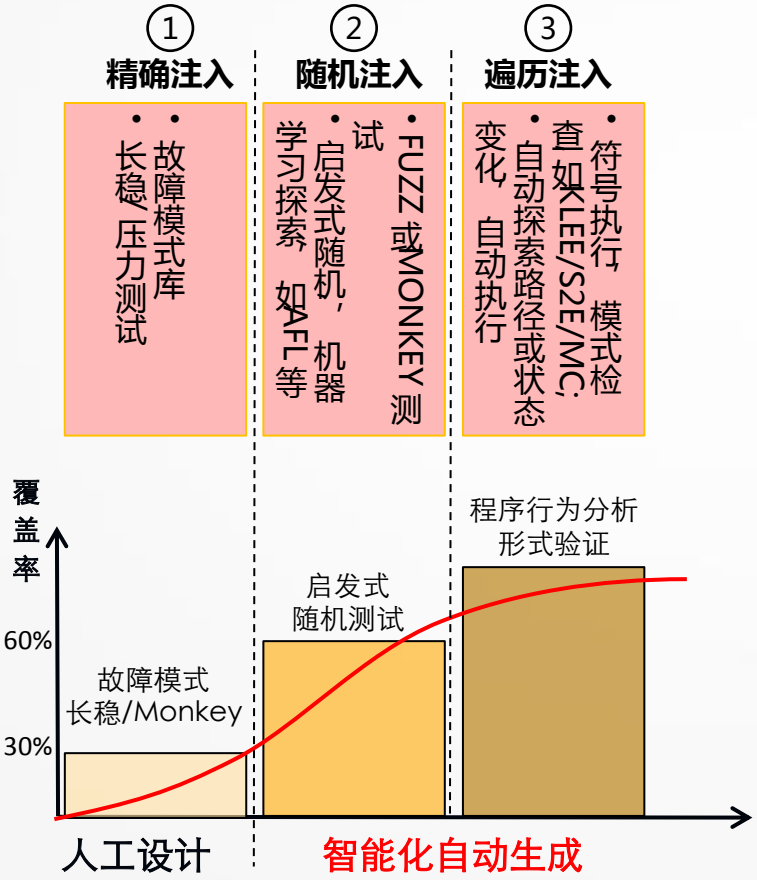
效果：回归用例数大减！

在相近覆盖率下用例规模缩小，减少验证时间和成本！

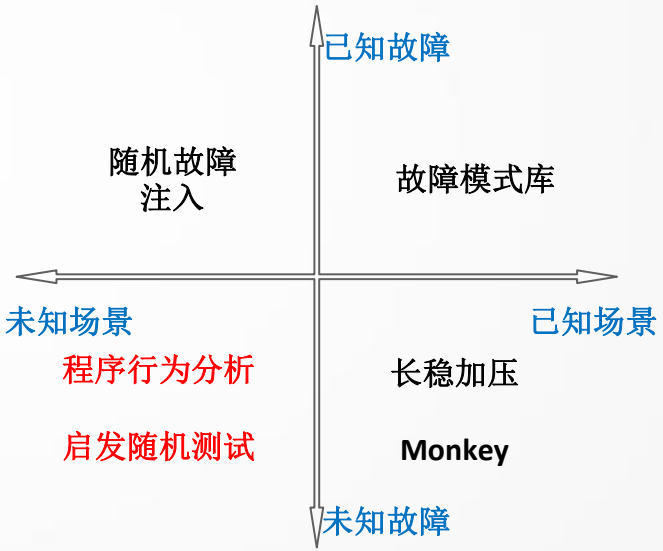
V 智能化测试覆盖实践1

海量社区代码可靠性覆盖，需要依赖智能化自动生成

可靠性测试三层覆盖



开放/云化场景应用策略



三种方法特点：

① 指定路径探索，人工设计，覆盖率低，基础防护；

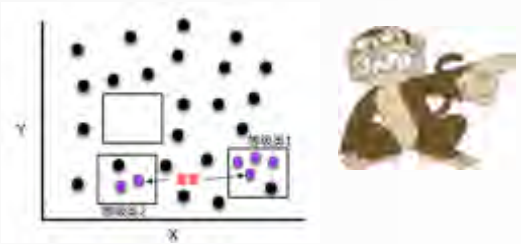
② **未知路径探索，速度快，覆盖率有上限，防人工用例老化；**

③ **全路径探索，覆盖率高，但使用门槛高，用作查漏补缺**；

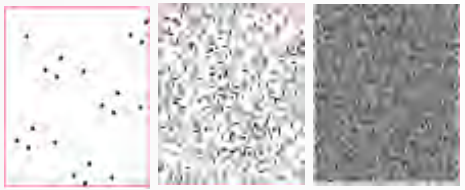
智能化测试生成，弥补开放场景下人工设计的局限！

V 智能化测试覆盖例实践1（续）
传统随机测试的不足及程序行为分析技术的优势

随机测试的不足



重复多，命中率低，亚健康无法检测

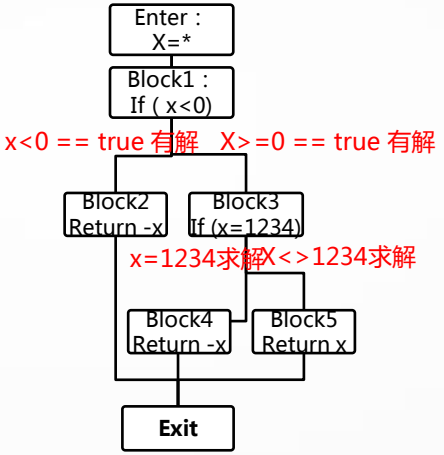


无结束机制，非第一现场，定位困难

例：if （input == 0x12345678）
{
//此处代码存在内存越界
}
随机测试遇到魔术字很难进入，靠运气。
覆盖有广度，无深度...

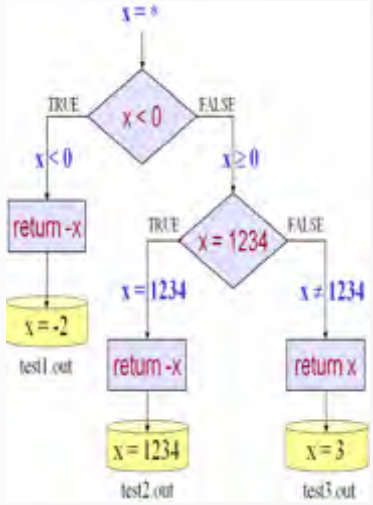
自动测试设计输入，精确遍历

程序分析控制流图（CFG图）



符号执行

路径探索结果，产生输入集（原始用例）



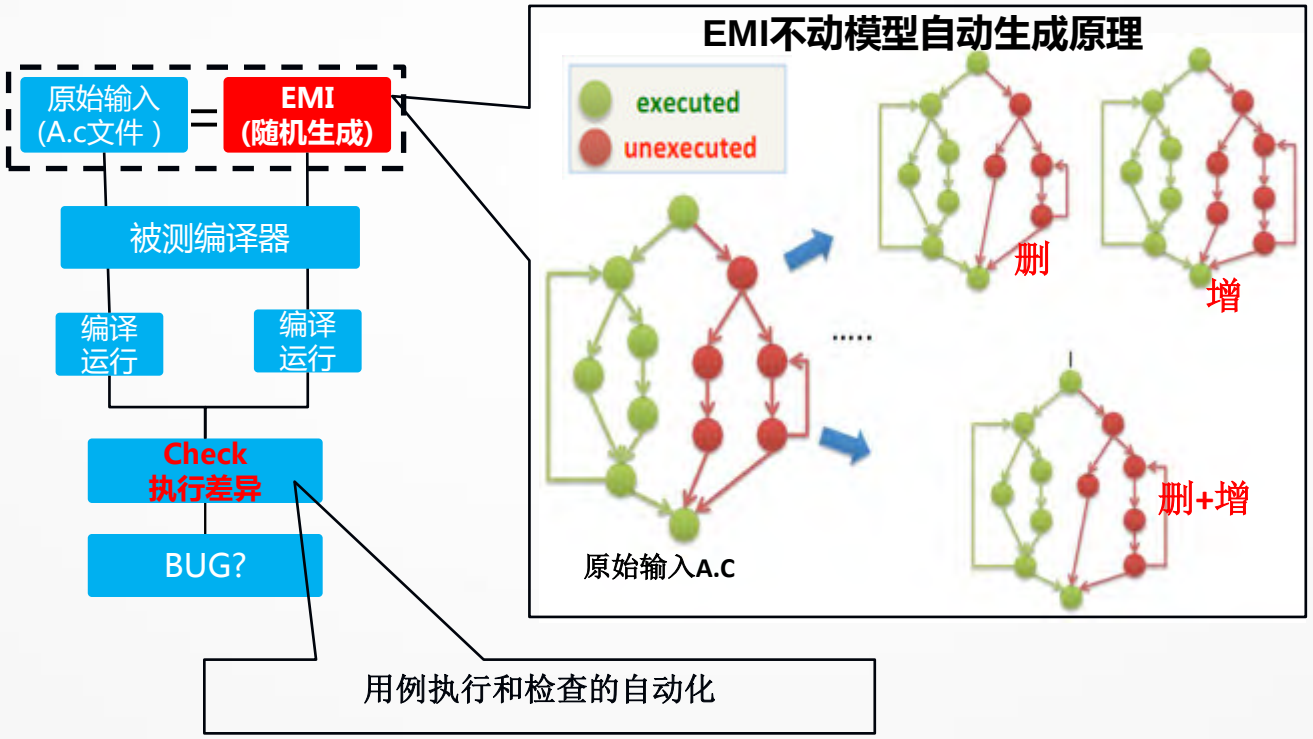
符号执行表示，包含value、path、path condition
test2.out为例，值x=1234，路径block1-3-4，条件(!(x<0))&&(x==1234)

优势：
高覆盖高命中率：对每条路径的精确分析和输入求解;
测试设计自动化：从关注执行到关注测试设计
效率高：测试自动生成，端到端效率提升；

V 智能化测试覆盖实践2
利用等价类/不动理论的编译器社区随机生成测试技术

EMI(Equivalence Modulo Inputs)：随机构造和原输入完全相同的**等价输入**，对比两次输入的运行/测试结果，如不同则可能出错；

应用效果：框架建好后，每天零成本自动产生并运行用例达到几十万个，未知场景代码路径的测试覆盖率大幅增加；



EMI执行过程：

1 原始输入A.C编译运行，找出unexecuted代码块（死代码block）；

2 构建等效模型B.C（增/删/改unexecuted的死代码，保证能编译通过即可）；

3 B.C编译运行，比较运行结果，不同则说明编译器有BUG；

目录

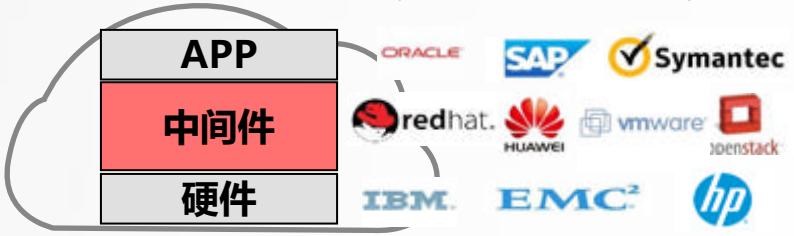
- 开源给测试带来的挑战
- 如何构建E2E开源测试能力
  - I 集成开源成果，享受社区质量红利
  - V 海量代码智能化覆盖增强测试
  - A 自动化认证测试

A 自动化认证测试实践
支撑产业伙伴间双向快速兼容测试，减少自投入

1）合作伙伴梳理：

●IHV：服务器/存储设备/交换机/芯片……

●ISV：ERP云套件/安全应用/操作系统/虚拟化/数据库……



2）兼容认证平台实现：

●测试套覆盖伙伴软硬件接口；

例：OS ISV认证需验证接口符合，IHV认证需验证CPU/网卡等硬件外设，保证生态伙伴间的互联互通及互操作。

●框架支持远程认证、测试自动化执行；

●报告支持加密及防篡改等安全要求；

3）自动化认证工厂：

●ISV/IHV环境及资源云化，远程使用；

●认证测试策略制定、环境部署、自动化执行；

●测试结果自动分析，输出兼容性列表；

认证5步走

自验测试

认证申请

认证测试

报告生成

结果确认

生态兼容，开放的产业链测试！

Thank you !