精准测试白皮书

V3.0

（2019 版）

目录

第一章　精准测试诞生的背景 1

第二章 精准测试的定义 4

第三章 精准测试的基础架构介绍 5

3.1 精准测试的技术架构 5

3.2 软件示波器 7

3.3 精准测试的双向追溯 10

双向追溯技术正向追溯 11

双向追溯技术反向追溯 12

数据追溯技术-追溯测试用例的全景调用 13

数据追溯技术-针对多系统多模块（微服务）的追溯 14

3.4 分布式结构下的数据穿透 14

第四章 精准测试的核心组件与功能 16

4.1 风险控制 17

4.1.1 七种测试覆盖率 17

4.1.2 新增代码覆盖率 19

4.1.3 测试覆盖率范围筛选与再统计 20

4.2 工作协同 21

4.2.1 打通开发与测试的隔阂 21

4.2.2 源码动静态数据的统一 22

4.2.3 缺陷最后执行时序分析 25

4.2.4 智能缺陷定位 26
4.3 敏捷迭代 29
4.3.1 敏捷迭代下多版本白盒测试数据的聚合 29
4.3.2 聚类分析 30
4.3.3 漏洞检出 32
4.3.4 精准测试与自动化测试对接 34
4.3.5 最小测试用例集 34
4.4 团队管理 35
4.4.1 精准测试的企业私有云可信化报表 35
4.4.2 精准测试的企业私有云-测试效率的直观展示 37
4.4.3 精准测试的企业私有云-测试用例排行图 39
4.5 知识库累积 41
4.5.1 精准测试数据的价值 41
4.5.2 精准测试智能回归测试用例智能选取 41
4.5.3 精准测试在回归测试中的性能评估 43
第五章 精准测试的管理报表分析 43
5.1 项目指标 44
5.1.1 程序代码信息汇总 45
5.1.2 程序覆盖率指标 45
5.2 测试用例-按日趋势图 47
5.2.1 测试用例汇总信息 47

5.2.2 测试用例按日趋势图 48
5.3 测试用例-测试用例列表 49
5.3.1 星云精准测试软件示波器（测试用例跟踪） 50
5.4 测试缺陷-Bug 信息汇总 52
5.4.1 Bug 按日趋势图和 Bug 类型分布组合 52
5.4.2 Bug 提交排行榜 53
5.5 测试缺陷-Bug 详细列表 54
5.6 覆盖率-按日增长趋势图 55
5.6.1 覆盖率信息汇总 55
5.6.2 覆盖率按日增长曲线图 56
5.6.3 雷达图 56
5.6.4 函数|类|文件覆盖率统计 57
5.7 覆盖率列表 58
5.7.1 覆盖率列表与单函数的覆盖率、复杂度雷达图 58
5.7.2 函数对应的调用关系图 59
5.8 复杂度-函数|类|包复杂度统计 60
5.8.1 复杂度统计信息 60
5.8.2 复杂度列表 61

第一章 精准测试诞生的背景

现代社会是建立在各种以计算机为基石的软件技术基础之上的。随着日新月异的需求变化，软件系统越来越复杂。很多人觉得软件开发才是重要环节，但实际上，无法对大型软件进行有效的质量把控，就无法真正构建与维护大型软件。——系统中任何一个错误都可能导致整个系统的崩溃，造成无法弥补的损失，系统的任何一个微小的修改都可能引入新的缺陷导致维护困难重重。

然而，如何从极端庞大复杂的系统中迅速及时地找到故障所在，却是行业的一大难点。目前国内软件测试基本处于两种状态：一是绝大多数企业采用功能（黑盒）测试，二是部分对软件产品有高可靠性要求的关键软件，企业会使用代码级的白盒测试工具，但这两种传统的测试办法在目前的软件智能化趋势下，更像是用竹竿打怪兽，完全没办法应付的。

功能（黑盒）测试，测试者看不到程序内部逻辑结构，这种办法对软件可靠性要求不高的应用来讲问题不是很大，但是对于大型金融保险、工业软件、航天军工等关键系统就意味着时刻携带隐形的巨大风险。为此，功能测试后期需要极高的人力投入才能完成复杂逻辑的用例分析和设计。然而对于黑盒测试来说，由于我们无法获知内部的逻辑构造，程序越大，杀虫剂效应越明显。而行业内当作银弹的自动化测试，当自动化程序本身规模扩大以后，它的维护本身就存在了很严重的问题。

代码级（白盒）测试工具一般重点应用在研发阶段的单元测试上，满足了客户的部分高可靠性需求，但由于其价格高昂、技术老化，仅适合于小规模迭代瀑布式开发的软件，

无法完成复杂的系统级别的测试以及分布式基于云的测试，更无法适应敏捷迭代的开发模式。而且值得一提的是，目前白盒测试工具基本都是国外产品，通常这些产品无法完成深度的定制化功能以及快速的用户响应，代码安全也是一个较大的问题。

随着国内军民各项大型核心软件系统的上马，研发一种面向高复杂度大型软件、自主可控的高性能智能精准测试平台，显得迫在眉睫。正是在这种时代背景下，2012 年初，星云测试团队开始心无旁骛的研发征程。精准测试是个交叉学科，里面涉及到编译器、测试分析、图形技术、高性能通信与存储，软件的研发等多项底层技术。

经历无数个不眠之夜对技术难点突破的煎熬与最佳解决方案的反复推敲，星云精准测试产品在诸多方面率先实现了重大技术创新，成功突破了白盒测试使用难度大、价格高昂的桎梏，有效消弭了国外高端测试产品垄断的壁垒。星云精准测试产品更偏向于软件测试业界的“灰盒测试”，即用简单的黑盒操作办法，可以同时得到单元级和系统级的精准测试数据。

“星云精准测试”在众多性能上大幅超越国外进口高端白盒测试工具产品，并在数据追溯、覆盖率可视化、智能回归、智能缺陷定位、分布式数据穿透与追踪等特性上有突出贡献。“星云精准测试 VIP 大企业离线版云平台”在整体测试功能上的优异特性，成功获得了一批重要大型企业的高度认可及产品采购。

星云精准测试的首发版本为：穿线测试 ThreadingTest，2014 年 6 月 6 日上线，侧重于系统级白盒测试技术，测试用例和代码逻辑的双向追溯技术，测试示波器技术，覆盖率可视化技术。

2015 年 8 月 6 日，“穿线测试”正式更名为“星云精准测试”。在继承穿线测试整体技术上，星云精准测试增强了回归测试用例的自动选取技术，缺陷最后执行时序

分析、智能缺陷定位、敏捷环境下多版本白盒测试数据的聚合、聚类分析、结合代码结构与动态数据的测试漏洞检出、代码安全特性，全面的测试管理特性等几十种优秀功能。

目前有“星云精准测试 VIP 大企业离线版云平台”、“星云精准测试 PASS 在线云平台 www.teststars.cc“、“全自动测试用例驱动生成系统 Wings”等多种工具产品。

星云精准测试旗下产品平台有 Horn、Paw、Shell、Wings 等系列产品。适用语言和平台暂为：Java、Object-C、C89、C99、C++0X11、C#等；适用平台：Android、J2EE（、Web）、Java Desktop、iOS、MacOS、Linux（X86、X64、mips、arm、powerpc、UNIX(AIX)、VXworks、Windows(visual studio.net)、Windows 操作系统、WinCE 嵌入式平台等。为响应广大用户的需求，目前正在进一步扩展适应的语言和平台覆盖面。

通过精准测试，即继承了传统功能测试前期的高效率运行区间，又能在后期通过系统的数据，让开发、测试充分协同，完成全程高效的测试。

（1）将测试团队的价值放大，能够将开发与测试更加紧密的连接起来，互为支撑。

（2）采用精准、可信测试技术，测试管理的难度大幅度降低。

（3）降低企业对人员的过度依赖，通过系统适应人员的变更。

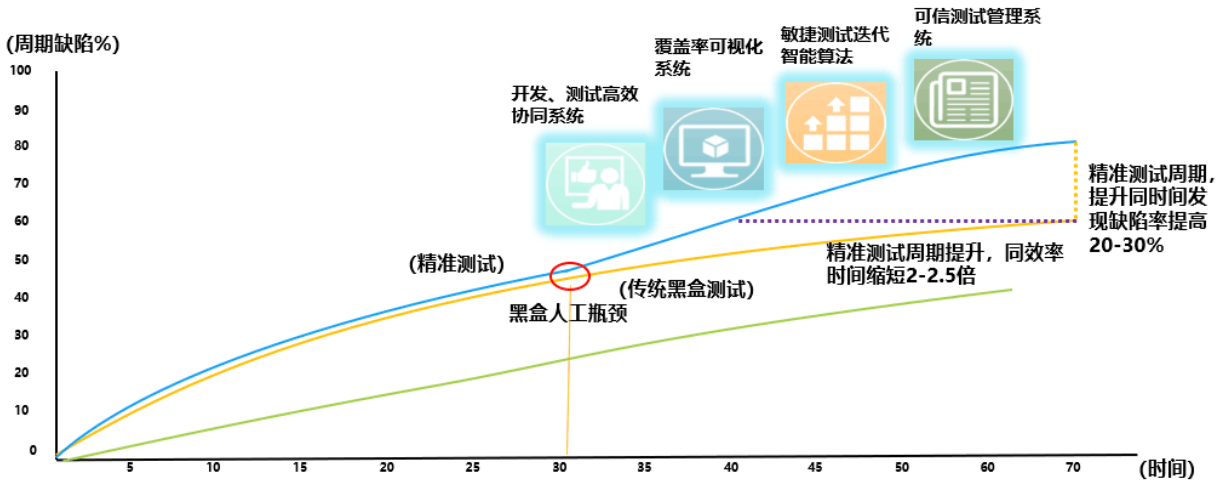


图 1-1 精准测试在大型系统的效率运行分析

星云精准测试，既保证了传统功能测试前期的高效率运行区间，又能在后期通过系统的数据，让开发、测试充分协同，完成全程高效的自动化精准测试。

第二章 精准测试的定义

精准测试：是一种国际首创的软件测试技术，旨在建立大型软件系统的测试数据与源代码之间高度的可视化追溯机制，实现精准缺陷预防及定位。它有力的打破了软件开发、测试、维护及管理人员等之间的数据交流屏障，支持超大型应用从开发、迭代、维护全流程的可视化精准测试跟踪和测试分析。即使是初级测试人员也能易于学习掌握，用黑盒测试的方法实现精准化测试。

精准测试使软件测试从完全依赖人工记录、验证，转换为机器智能的全过程精准、可视、可信的全新检测模式。精准测试数据和黑盒测试优雅对接，在不改变常规测试流程的情况下，就可以获得大量的精准分析数据，并直接引导用户进行高效的后续测试与质量风险评估。用户手动“点测“或者与自动化对接被测试应用的同时，可

以快速记录对应的代码执行逻辑并实施同步运算和分析，给出被测试应用的质量诊断报告。例如测试过程中的关键模块漏测分析、测试充分度度量、代码静态质量分析以及崩溃的代码级的捕获和分析等。

精准测试有着超强的数据追溯机制，通过建立用例和代码运行时数据的映射关系，能够很好的协同开发和测试工作；它适用于当前流行的敏捷开发、测试体系，在版本迭代中，能够准确的计算出由于版本迭代影响和波及的测试用例，快速给出测试复杂度报告并核确定测试范围优先级，极大减少上线风险。在团队管理上，精准测试亦产出数十张过程及管理的不同剖面报表，以满足各级管理需求。

第三章 精准测试的基础架构介绍

3.1 精准测试的技术架构

星云精准测试的技术架构：通过对源代码的插装分析出代码的静态结构信息，运行插装后的代码，测试工程师通过人工或自动化的执行用例，软件示波器通过采集到的这些数据，进行相关密集运算，得到测试数据。结合之前已有的代码静态结构信息，在星云客户端可实现用例与函数直接的互相追溯，再通过星云测试工具的企业项功能，缺陷定位、用例聚类分析、回归测试用例和最小测试用例集得到相应的测试数据，星云测试通过报表的形式展示测试数据，导出批量测试报告。

精准测试从某个层面来讲，是赋予了测试用例真正的生命力，传统的测试用例仅仅是一些只能够依赖人去理解和分析的文本文件而已，在计算机和算法层面则没有存在意义和价值。下图是精准测试的整体架构图：

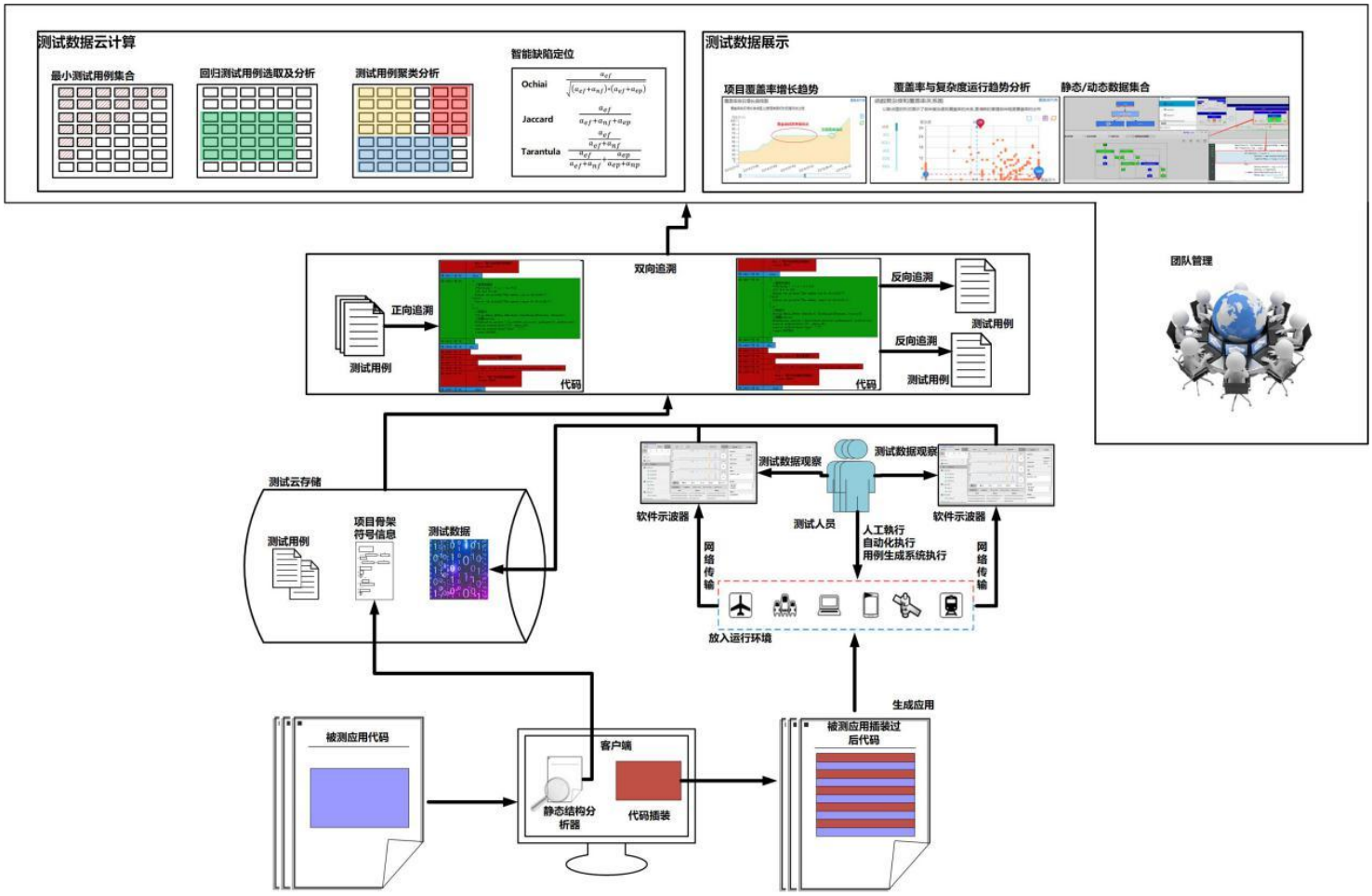


图 3-1-1 精准测试的总体架构图

大家首先可能会比较好奇，“用例魔方”的概念是怎么来的？测试用例魔方是在精准测试的设计、开发和商业实践中自然产生的功能集合的一个统称。当我们把精准测试的和用例分析相关的功能画成架构图形表示的时候，它自然而然地看起来就像魔方，所谓“魔”则是精准测试核心算法所赋予的超能力。

上图是星云精准测试系统的总体结构图，“测试魔方”即分布在左上角区域。大家知道精准测试的核心技术是测试用例与代码的追溯关系的建立，而在此之上就可以构建测试魔方的核心功能区。如下：

图 3-1-2 精准测试的测试魔方

所谓“方”实际上是代表测试用例的集合，每个测试用例用一个小方块标识，所有测试用例的集合用一个大方块。精准测试体系中，测试用例对应的代码逻辑都可以实现全自动的追溯和存储，因此测试用例就具备了进行深入分析的基础。在精准测试的用例魔方中，目前存在三个面（随着后续功能的增加，将增加分析的面），即回归测试用例选取、测试用例聚类分析、测试用最小化，同时辅之以智能缺陷定位技术。下面对精准测试的功能做详细的说明。

3.2 软件示波器

精准测试采集到的测试数据在软件示波器页面，通过可视化的窗口展示，实时展示采集到的块、条件和函数信息，在下方列表实时展示函数调用信息。软件示波器采集到的测试数据，完美实现了用例与代码的自动关联。通过测试数据的反向追溯分析，开发人员可进行一致性修改，避免修改引入新的缺陷，通过正向追溯结果，开发可对用例的执行进行全面掌握，可用于快速修复缺陷和详细实现确认。

用例与代码的在追溯是精准测试的基础功能，后面的高级算法都在这个基础上展开，用例和代码的追溯就像一个全景的调试器，只要功能由测试人员进过运行，所有的内部代码执行逻辑瞬间就可以展示出来。

软件示波器中的测试用例可以从现有的测试管理系统导入进来，当准备开始执行一个用例的时候，选中用例点击开始，然后驱动被测试系统运行，那么软件示波器就会采集到程序内部运行逻辑对应的波形信息，当用例执行结束，点击停止。这个用例运行阶段的数据，通过开始和结束的边界就记录下来了。

软件示波器主要起到有效的可视化测试过程的作用。在执行用例过程中，如果没有采集到测试数据或者程序出现崩溃的情况，软件示波器就像人的心脏并没有跳动一样，一根横线拉直。正常采集到数据，将有持续的波形展示出来，高效而精准地监控到程序细微的运行状况。它可以精密捕获每个软件单元任何微小的运行波动和行为改变，并支持多次运行数据的比对。

同时软件示波器也提供一个辅助的等价类划分的功能，它将一个用例从开始到结束所执行的路径信息终值，完整记录下来。如果两个用例终值不一样，就可以确定为不是等价类。对于很多从功能表面很难界定是否等价类的测试用例，软件示波器可以给出精确结果。

通过软件示波器高速采集程序数据：

（1）只要测试开始执行，即可以透明方式采集功能运行过程中对应的程序的运行逻辑。

（2）在系统高速运转下采集，可保证对原有应用无干扰，超过 1500w/s 的采集速率。

（3）可采集程序的条件，执行路径，执行参数，内存使用等动态运行数据。

软件示波器的采集速度极快，目前最高可以每秒钟采集 1500 万条测试数据，对被测试程序的性能影响非常小。

图 3-2-1 软件示波器

为了方便客户在对测试时的实时数据监测，数据实时动态刷新的时候能够方便看到数据，星云做出了实时数据监测的悬浮窗，这样就能在运行项目的时候就能更方便的看出数据的变化

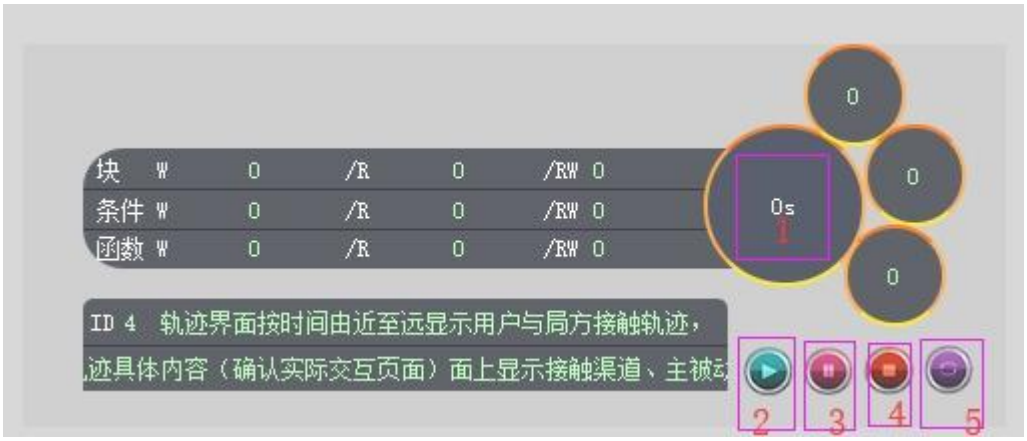


图 3-2-2 软件示波器悬浮窗

只要将鼠标移至悬浮窗就可以看到这条测试用例的 ID 和名称

悬浮窗的块，条件和函数就是动态实时监测界面的块块，条件和函数

悬浮窗的额绿色按钮表示开始，中间的红色按钮表示暂停，后面的按钮表示通知

后面的圆圈分别表示了块，条件和函数的消息数，中间的圆圈表示了测试用例运行时间

1 位置：鼠标点击可以收放左侧的数据块

2 位置：点击鼠标左键开始接收当前用例的运行的数据 快捷键：Space

3 位置：点击鼠标左键暂停当前接收 快捷键：Ctrl+Shift+Space

4 位置：点击鼠标左键停止当前用例的数据接收 快捷键 Space

5 位置：类/块数据类型切换[视图切换] 快捷键：Ctrl+Shift+Q

3.3 精准测试的双向追溯

精准测试提出了测试用例和代码的双向追溯，它也是精准测试核心技术之一。即运行一个测试用例以后，精准测试可以通过程序自动的记录和显示这个测试用例执行的代

码。如果测试人员关注某一些代码行，它可以追溯出哪些测试用例在运行过程中运行过这段代码。通过这个技术特性，测试工程师的每个测试用例都可以进行量化分析和统计，这些量化数据既可以用来对测试工程师进行工作的考量，也可以提供开发人员和测试人员之间进行信息化的交流。

双向追溯技术记录了每个测试用例对应的程序内部的执行细节，细致到每个条件、分支、语句块的执行情况。开发人员可以通过双向追溯的结果去理解程序逻辑，进行软件维护以及进行可一致性的修改。开发和测试可以顺利交流，增加测试和开发的交流效率。

双向追溯技术正向追溯

将测试用例和代码执行信息自动关联，可到函数级别及代码块级别；通过正向追溯可直接在代码级定位测试现场故障和缺陷逻辑，并提供最后运行的时序数据；通过正向追溯自动记录产生功能对应的详细设计实现，辅助软件解耦和架构分析。

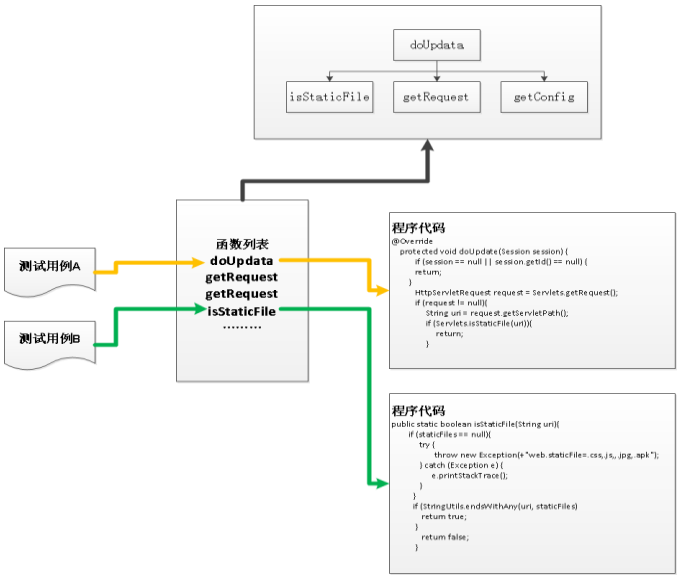


图 3.3-1　双向追溯(正向)-测试用例追溯到代码

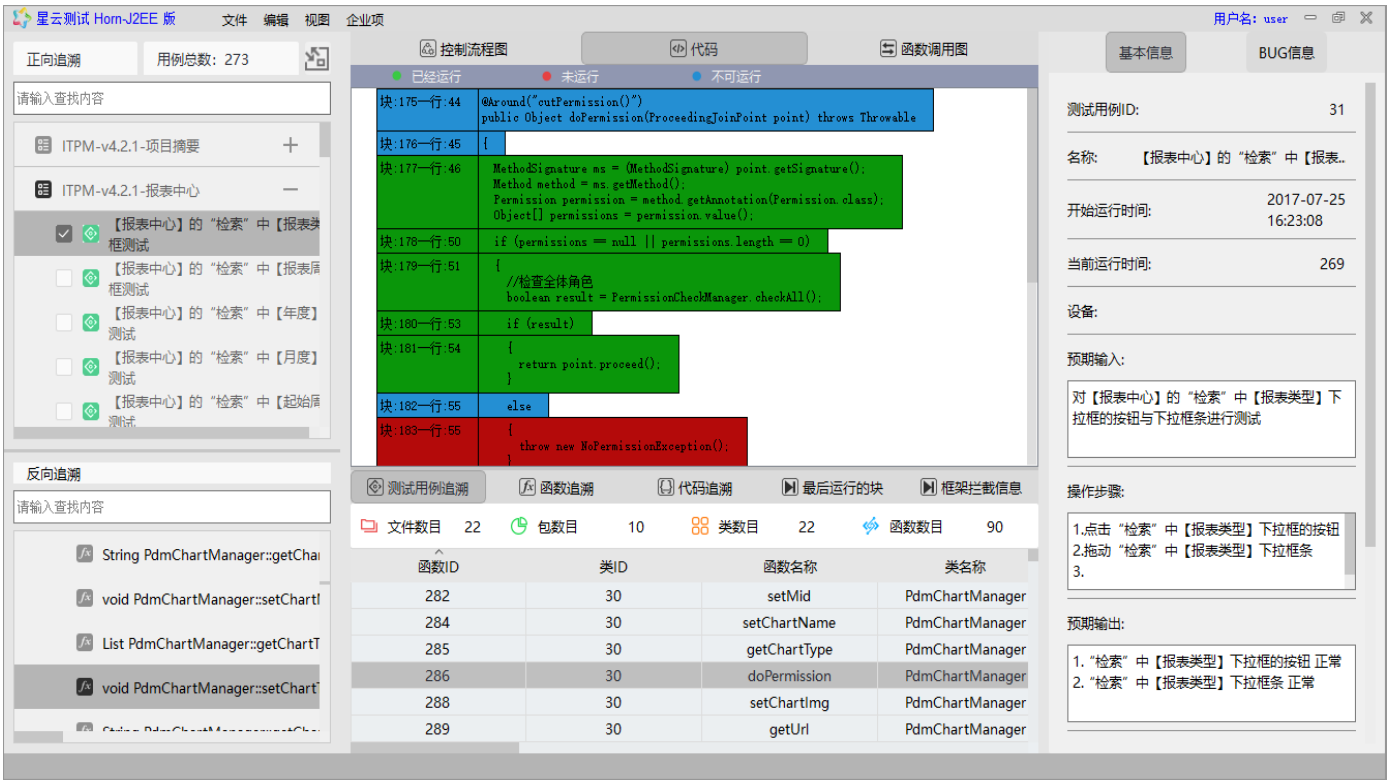


图 3.3-2　双向追溯(正向)-测试用例追溯到代码

双向追溯技术反向追溯

将代码执行、函数、代码块级别和测试用例执行信息自动关联，通过反向追溯可直接在观察代码变动所影响的测试范围，帮助开发人员代码修改影响功能范围评估与测试人员对代码修改部分所影响的测试用例进行评估。

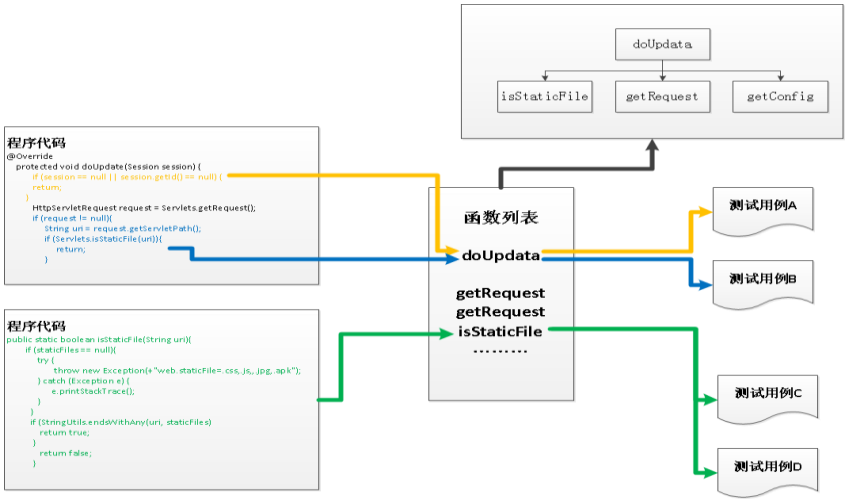


图 3.3-3　双向追溯(反向)-代码追溯到测试用例

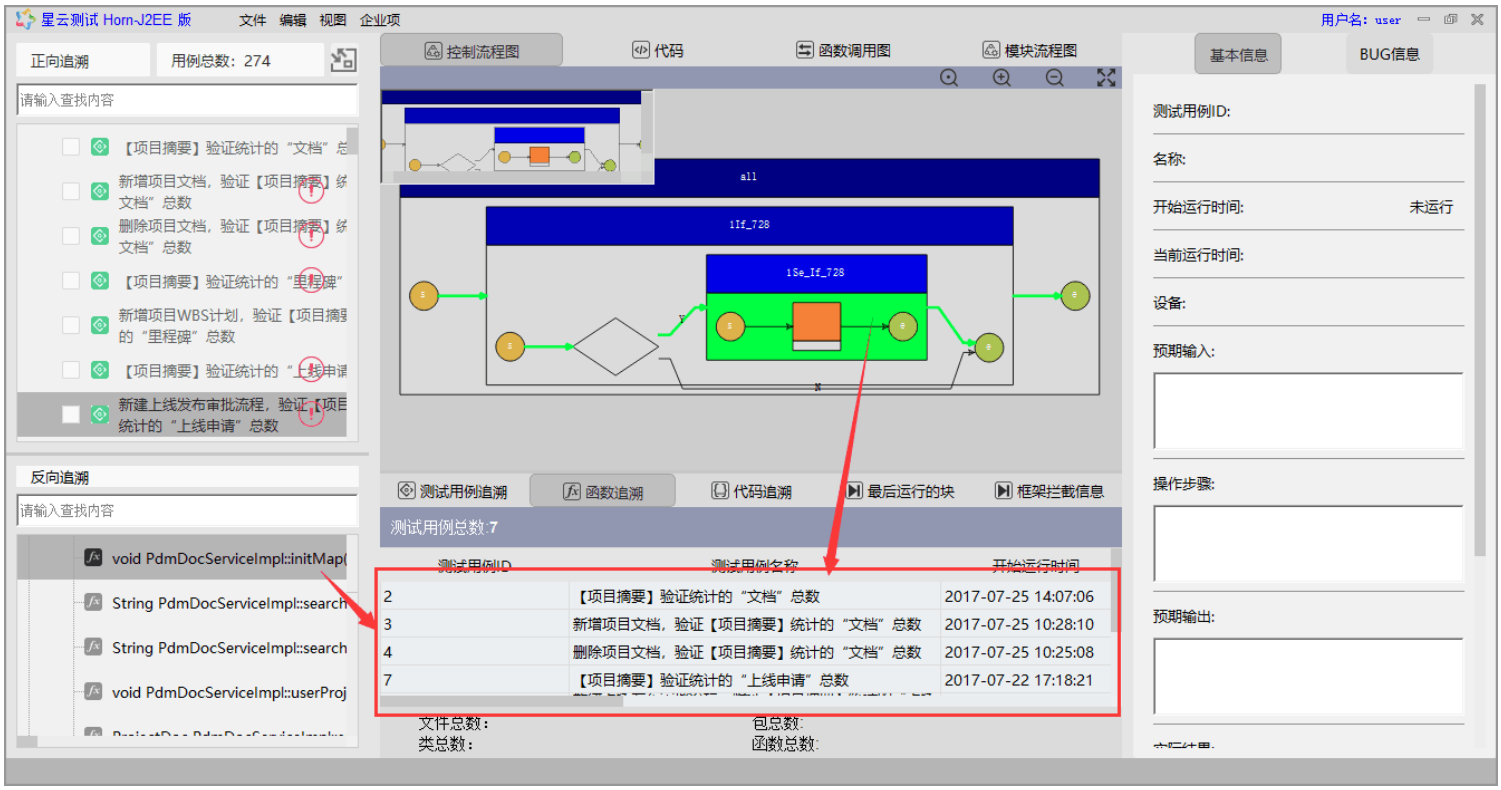


图 3.3-4 双向追溯(反向)-代码追溯到测试用例

数据追溯技术-追溯测试用例的全景调用

精准测试通过正向追溯把测试用例运行的代码执行进行了全景绘制，在全景图中，测试人员可以有效的观察到函数之间的整体的调用与走向，观察出被测模块与上层之间的调用关系

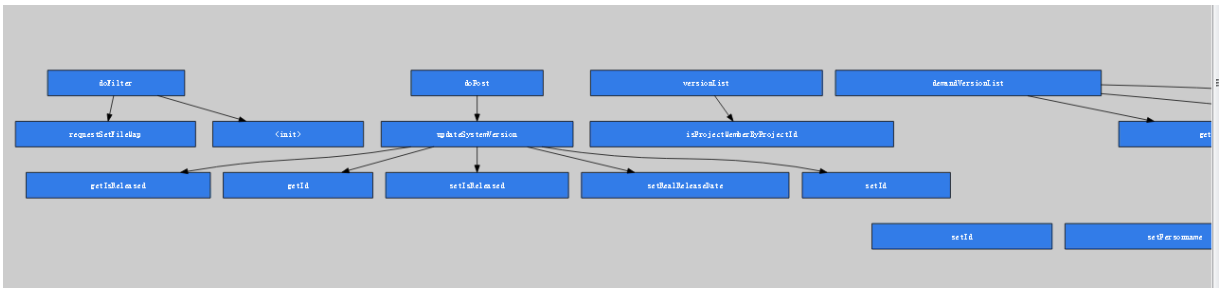


图 3.3-5 测试用例运行的代码整体调用

数据追溯技术-针对多系统多模块（微服务）的追溯

对于系统之间或模块之间往往通过 HTTP、HTTPS、等通信协议进行，而星云测试通过 agent 技术，把测试用例进行过的多个系统或多个模块之间的调用进行了记录并绘制成展示图，测试人员可以很直观的观察出测试用例从起始点到进行的各系统或各模块之间的调用关系图。

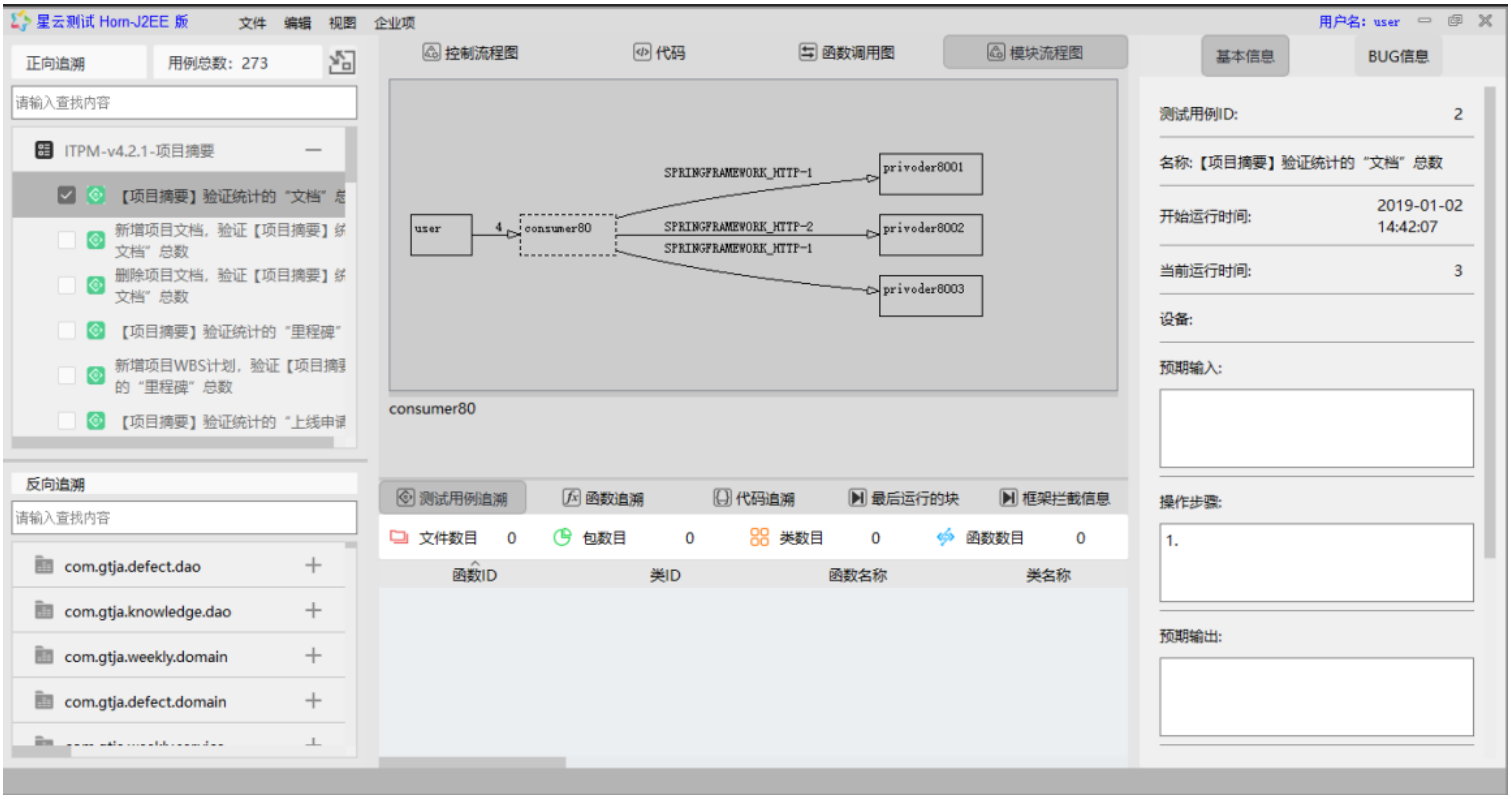


图 3.3-5 多业务模块数据穿透之间的调用

3.4 分布式结构下的数据穿透

微服务是一个新兴的软件架构，它把一个大型的单个应用程序和服务拆分为数十个的支持微服务，独立部署、互相隔离，通过扩展组件来处理功能瓶颈问题，比传统的应用程序更能有效利用计算资源。微服务之间无需关心对方的模型，它通过事先约定好的接口进行数据流转，使业务可以高效响应市场变化。但微服务一个明显的表象就是随着

服务的增多 ,传统的测试模式受到很大制约 ,无法有效进行下去 ,威胁到整体系统质量。

星云测试(www.teststars.cc)发布分布式微服务精准测试解决方案 ,是目前市场上唯一可达到在复杂分布式系统中跨多个服务器进行代码白盒级分析 ,并实现请求分布式追踪的测试平台。其中产品内的穿透模块，可以支持各种主流微服务通信架构，例如 httpclient，springcloud 以及消息队列，将并发访问场景下跨多个服务多组代码逻辑分离并重建追踪出来。实现了业务逻辑的代码在开发层面通过微服务离散后，在测试阶段则可以反向复原整个完整代码执行视图。精准测试里面的穿线概念(Threadingtest) 增加了第三层含义，即针对的分布式服务的穿透能力。

星云测试针对复杂的分布式系统中跨多个服务器（比如启动多个 spring boot）进行代码白盒级分析提供分析，实现请求分布式追踪，产品内的穿透模块，可以支持各种主流微服务通信架构，例如 httpclient，springcloud、dubbo 以及消息队列等。

星云测试将多个用户并发执行测试用例场景下跨多个服务多组代码逻辑分离并重建追踪出来。

默认情况用户标识采用浏览器的 cookie 值，测试前端浏览器设置。

微服务支持以下协议:

h :HTTP3 ,HTTP4 ,OKHTTP ,org.springframework.http.client ,cn.hutool.http，dubbo，feign 客户端

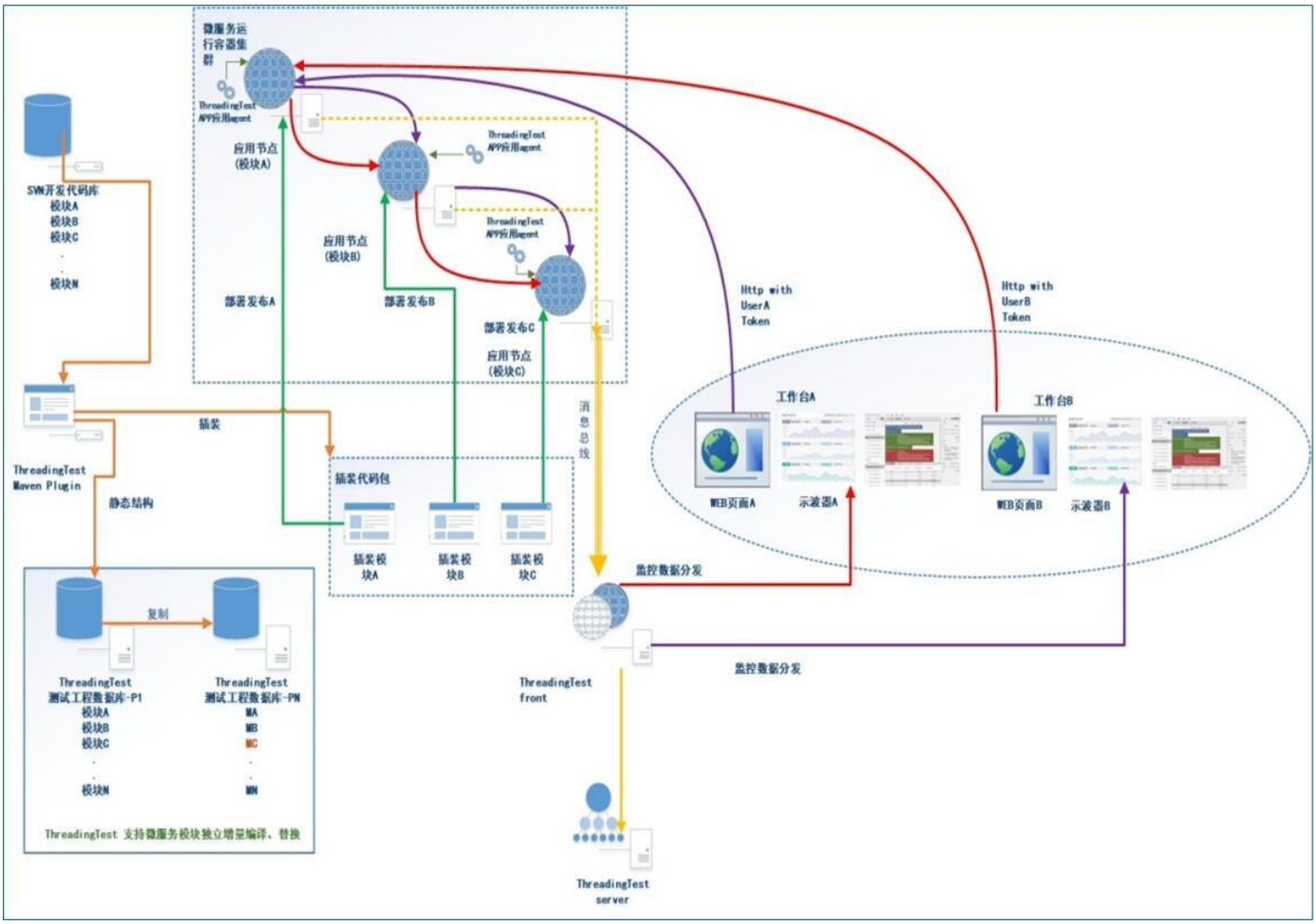


图 3.4 微服务

第四章 精准测试的核心组件与功能

精准测试的核心组件与功能包含：软件测试示波器、用例和代码的双向追溯、智能回归测试用例选取、覆盖率分析、缺陷定位、测试用例聚类分析、测试用例自动生成系统，这些功能完整的构成了精准测试技术体系。

精准测试系统的本质是一套强大的计算机开发与测试系统，实现数据可视化联动的辅助分析系统，它的关键技术是测试用例和代码的双向追溯技术。在这项技术的基础上，很多高级测试算法得以应用同时将测试和开发进行非常紧密的连接。精准测试

系统并没有取代人工设计用例、执行用例的过程，但是通过对该过程深入到代码层的分析，可以相当大的程度改进人工测试所产生的各种问题。

接下来将从风险控制、工作协同、敏捷迭代、团队管理、知识库累积五个方面详细解析精准测试的核心组件与其功能。

4.1 风险控制

4.1.1 七种测试覆盖率

星云精准测试提供 7 种测试覆盖率：分别为：SC0 语句块覆盖率、Ture 覆盖率、Both 覆盖率、CDC 覆盖率、Branch 覆盖率、MC/DC 覆盖率。

精准测试支持查看一个模块的范围内的覆盖率，以及把一些代码排除出计算范围重新进行计算等高级功能，也可以查看到新增代码部分的覆盖率情况。通过对这些覆盖率数据的分析，可以将风险控制到最低。

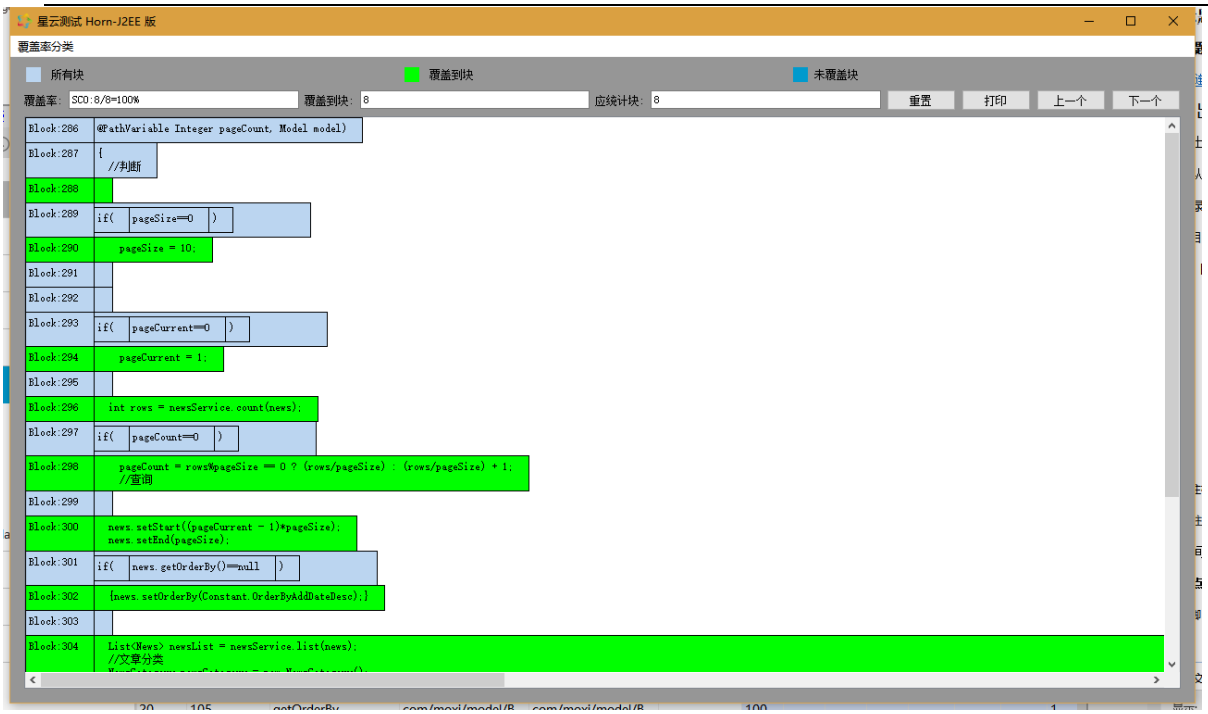


图 4-1.1-1 七种测试覆盖率

MC/DC 覆盖率可视化

星云精准测试覆盖可视化技术使每种覆盖率如何计算、分子分母分别对应程序的哪些单元，展示的非常清晰。

星云测试提供 MC/DC 覆盖率，即修正判定条件覆盖，该覆盖率数据 MC/DC 是 DO-178B Level A 认证标准中规定的，欧美民用航空器强制要求遵守该标准。对于金融系统的一些关键模块，也可以采用这个覆盖率标准，MC/DC 覆盖率可以基本保证被测试软件不存在缺陷。

MC/DC 覆盖是指所有符合条件中的子条件，在保持其他子条件不变的情况下，它自己的真假变化就会引起整个条件结果的变化，如果符合条件中的每个子条件都满足了，那么整个条件的 MC/DC 就满足了。星云精准测试提供相关的条件组合的结果展示，直接分析 MC/DC 覆盖率的满足情况。

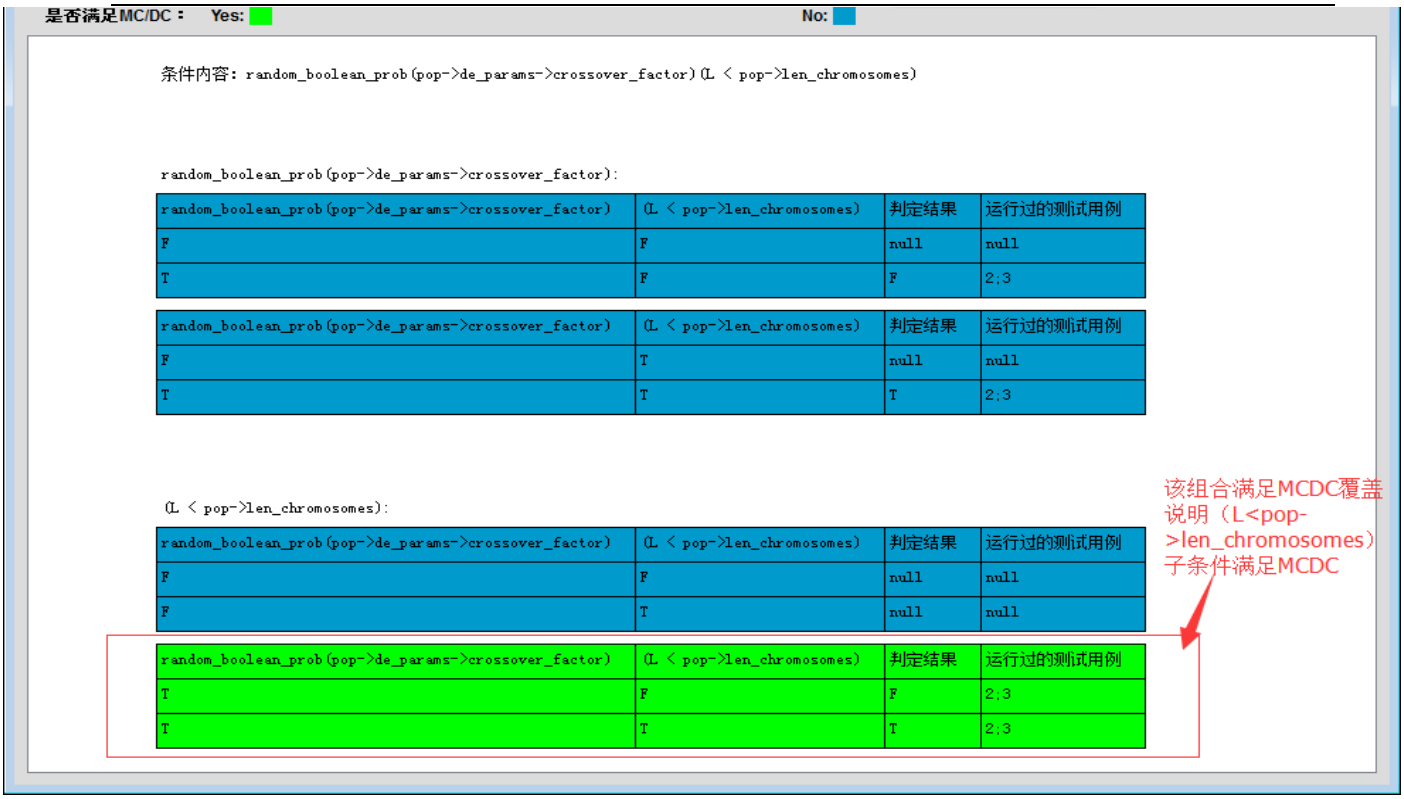


图 4-1.1-2　MC/DC 覆盖率可视化

4.1.2 新增代码覆盖率

敏捷模式下迭代频繁，测试人员往往被要求对本次变动或者新增的功能进行回归，但实际过程中新版本的代码经常由很多开发进行修改，容易出现彼此不知道或遇到有代码洁癖的，改了别人的代码，大家都不知道。通常情况是，要么测试范围定小了，遗漏了；要么测试范围过大，付出过多代价，而精准测试通过新版本与老版本之间的差异进行比对，给出变动和新增的代码的范围，帮助测试人员对本次要求的变动代码和新增代码进行针对性的覆盖率统计展示。

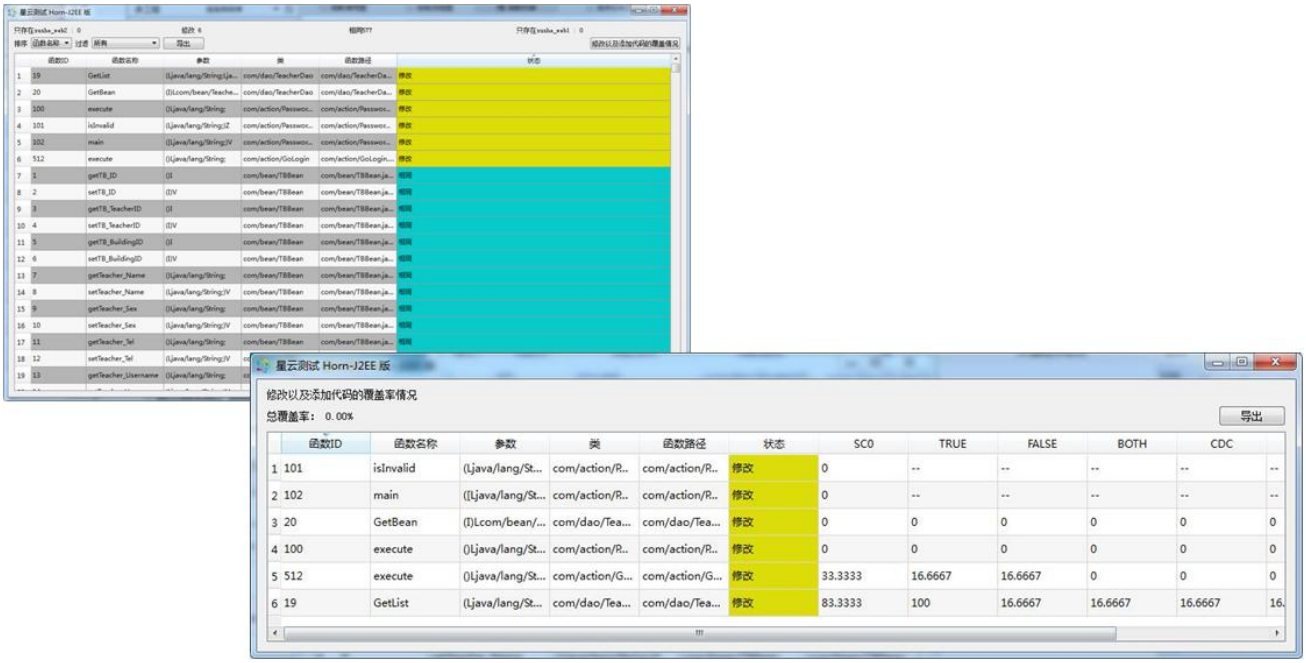


图 4.1.2　新增代码覆盖率

4.1.3 测试覆盖率范围筛选与再统计

在做精准测试或统计覆盖率时，往往测试管理者、开发人员、测试人员为了保证测试覆盖率的正确性，会对某个方法、类进行查看或在统计中把代码中一些废弃的函数或特殊测试不到代码进行移除，从而让测试代码覆盖统计率达到更加准确。星云精准测试在设计中，通过多种搜索、方法、类、模块过滤等功能把需要统计的范围进行缩小或不需要的统计的进行去除，并根据用户的选择进行覆盖率再统计展示。

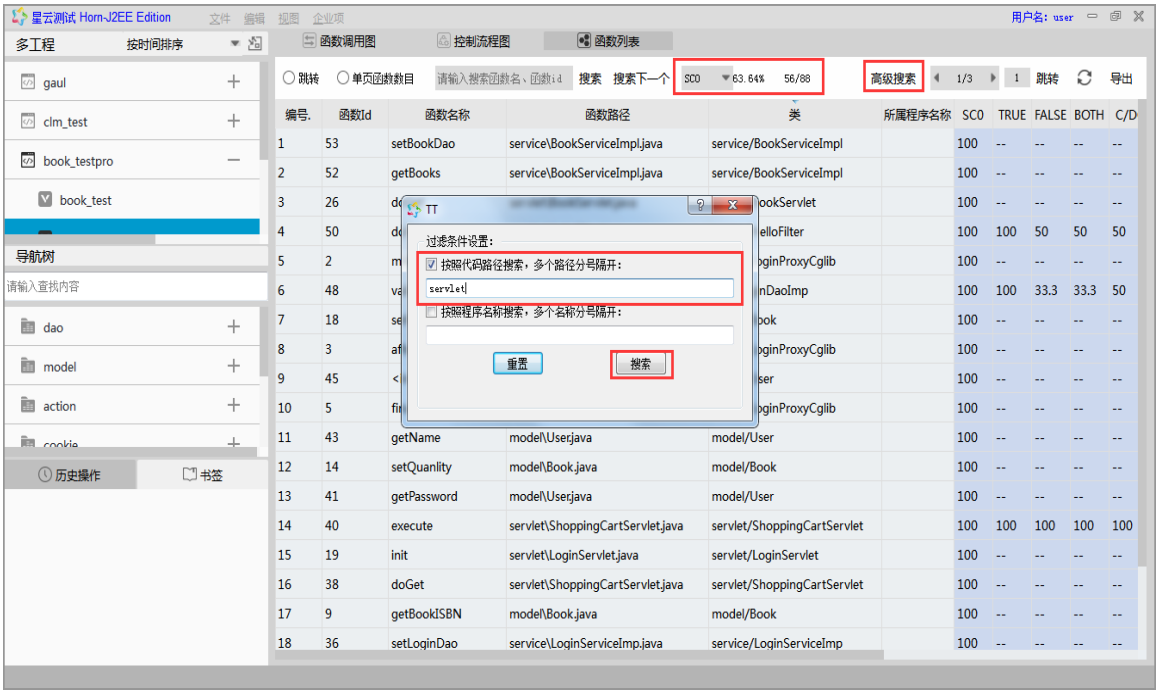


图 4.1.3　测试覆盖率范围筛选与再统计

4.2 工作协同

4.2.1 打通开发与测试的隔阂

精准测试打通开发与测试的协同工作通道，使得开发与测试能够更好的沟通，提高工作效率。传统模式下，开发人员关注的是代码，测试人员关注的是业务角度的测试用例，他们没有直接关联。开发和测试的沟通，基本就是采用自然语言、Excel 表格、内部系统沟通，存在大量的问题。例如测试工程师发现一个缺陷，提交到缺陷系统，开发需要花费大量时间理解、准备数据、复现、调试，直到最后的修正。因为业务上的功能执行和代码并没有明确的关系，通常测试工程师执行完功能测试用例后，让开发人员帮助评审也非常困难。

若测试工程师提供的测试结果都是比较模糊的功能逻辑描述，重现缺陷需要花费大量的时间。开发人员修改代码后，对于变更描述，以及变更引起的关联问题描述通常也都很模糊，导致测试又出现新问题。

企业采用精准测试技术后，通过执行用例可以直接追溯到对应执行的程序代码块，这样的数据化沟通，将使开发人员和测试人员之间的协同工作效率大大提高。

图 4.2.1 协同模式

4.2.2 源码动静态数据的统一

星云精准测试通过插装得到的项目静态结构信息，结合测试后采集到的测试数据，能够精准记录测试的过程，通过这些静态数据和动态数据视图，便于开发人员基于图形化结果进行快速分析。

对于不懂开发的测试工程师，通过程序控制流程图的图形以及通过颜色表示的覆盖信息，可以直接看到程序内部漏测的逻辑是什么，也可以通过这些结果直接与开发沟通，

进行辅助用例和逻辑的补充。

因为内部逻辑能够图形化的打开和看到，可以有力保证黑盒测试后期，开发快速理解和介入瓶颈问题，保持全程测试的高效执行。

图 4.2.2-1 源码静态结构与动态测试数据统一图（函数调用图）

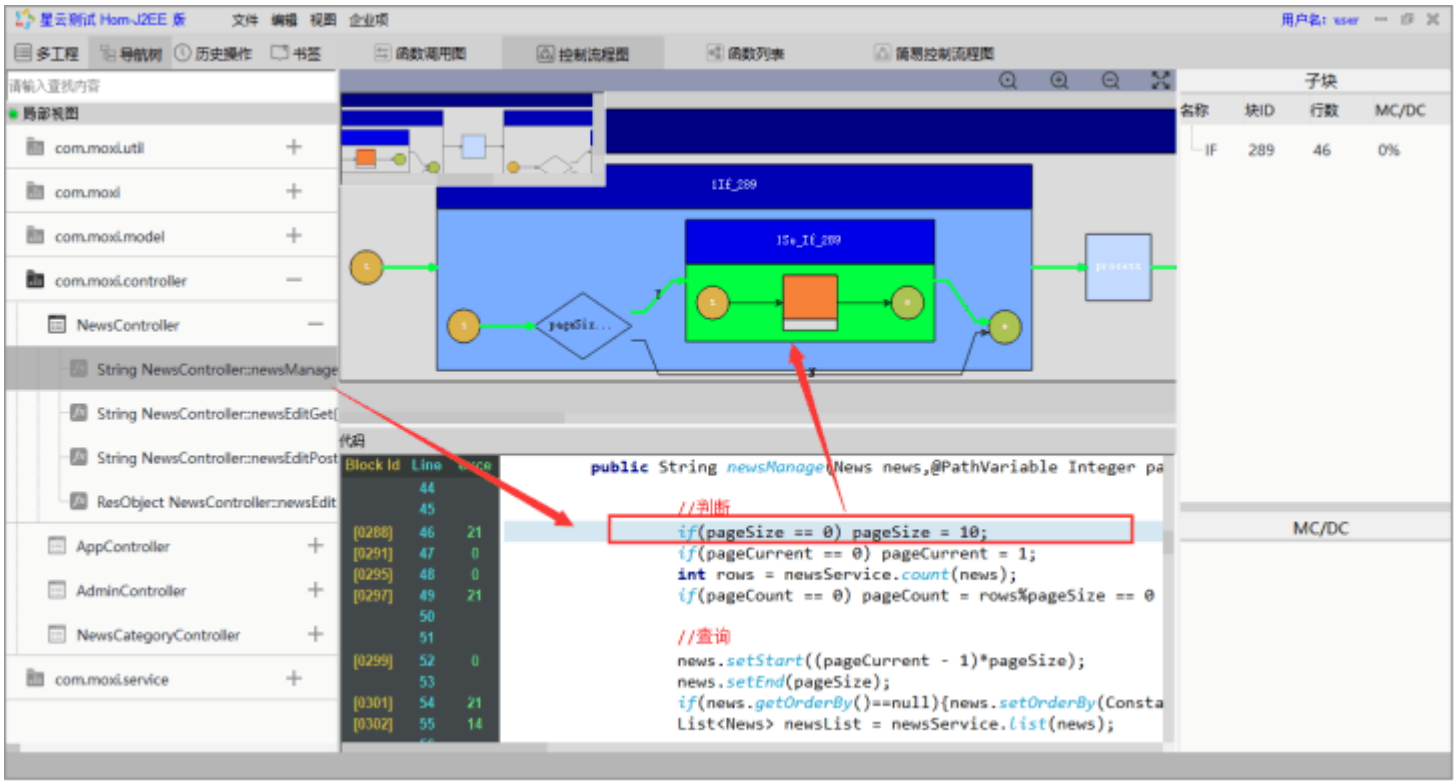


图 4.2.2-2 源码静态结构与动态测试数据统一图（控制流程图）

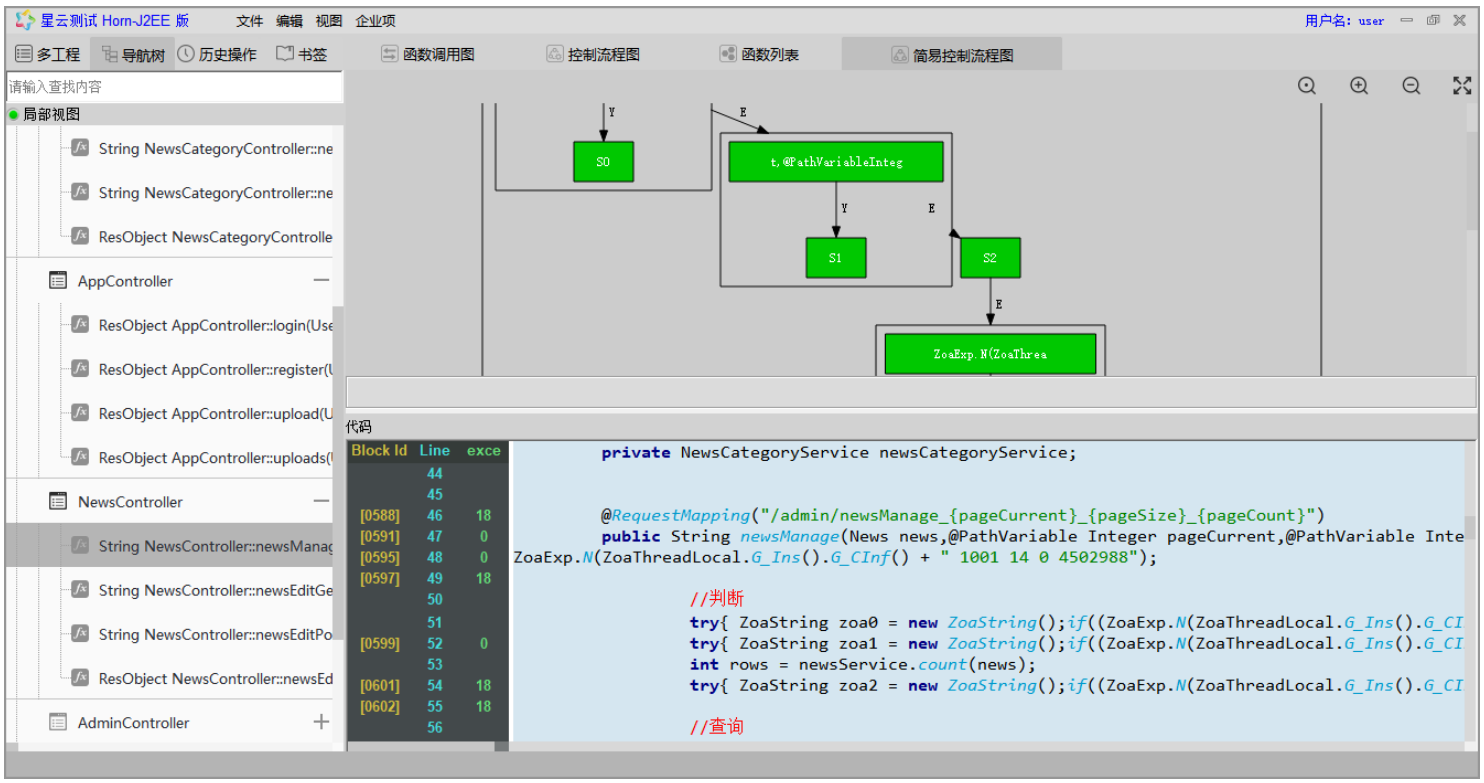


图 4.2.2-3 源码静态结构与动态测试数据统一图（简易流程图）

下图展示的是一个函数的静态结构，与动态测试数据结合的流程图，如图框架是静态结构，绿色显示部分是覆盖到的代码块。

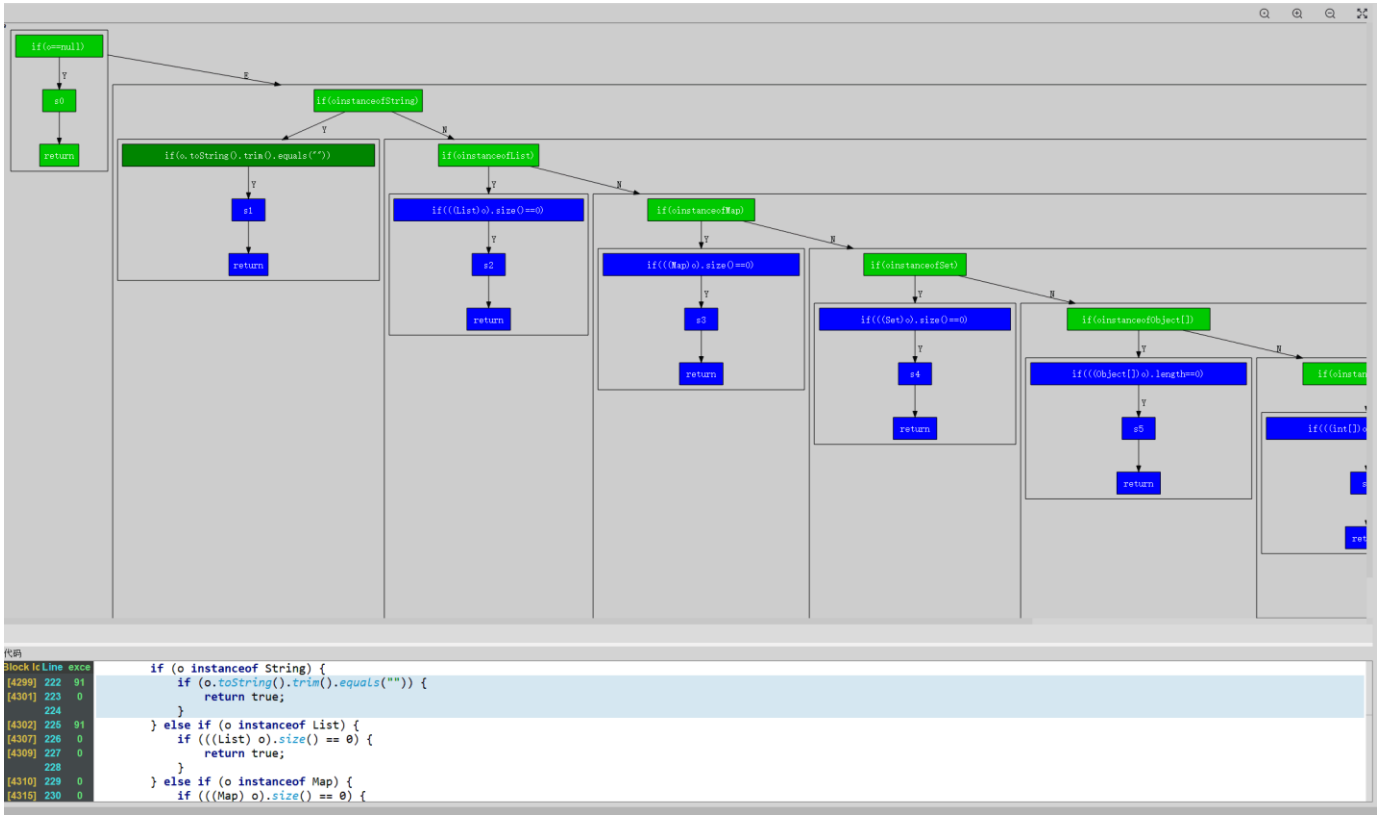


图 4.2.2-4 源码静态结构与动态测试数据统一视图

4.2.3 缺陷最后执行时序分析

星云测试采集数据时，可自动捕获缺陷或崩溃发生时之前程序执行的详细路径信息。当缺陷发生后，开发人员能够直接看到缺陷出现时，代码执行的时序和路径信息，直接定位缺陷和排查问题，节省大量的沟通以及复现和调试的时间成本。

IT 工程师可直接观察到程序出现缺陷后，最后执行的 50 个代码块、条件、判断的执行信息。配合示波器来观察，当功能执行发现缺陷或崩溃时，示波器可以设置成手动或者自动停止，清晰记录最后执行的 50 个代码序列的相关详细信息，以供查询和分析。

自动记录崩溃发生时刻之前程序执行的详细路径信息，捕获难以重现的缺陷并快速解决：

最后 50 个代码块。最后 50 个条件。最后 50 个判定执行。

除了在研发环境内，也可以在用户现场精准定位缺陷，而无需再用户现场部署任何代码。

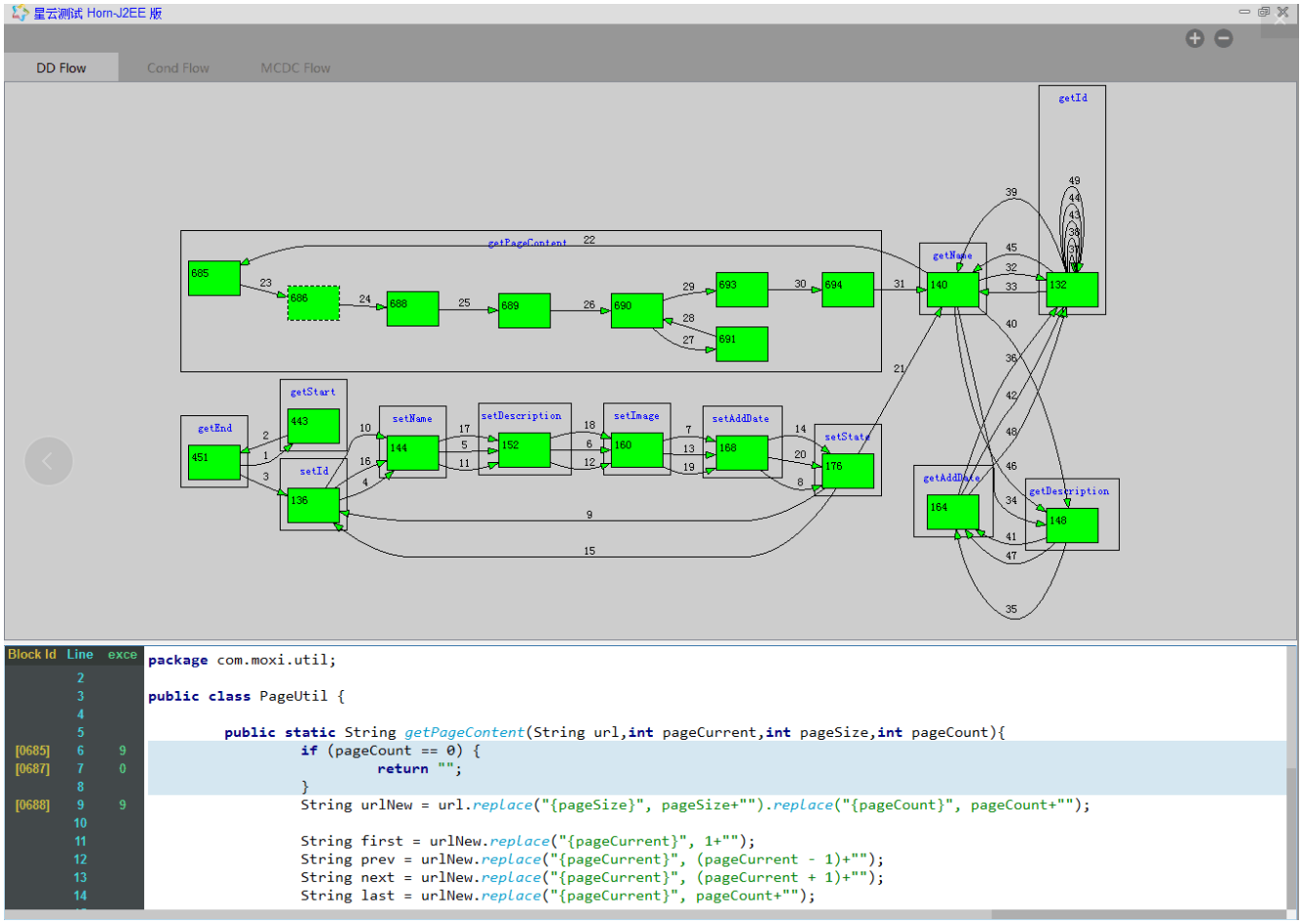


图 4.2.3 缺陷最后执行时序分析

4.2.4 智能缺陷定位

星云测试对企业级用户提供缺陷定位功能。通过测试人员标记用例执行状态和软件示波器自动记录的程序，可自动分析缺陷出现的可疑代码块。

传统测试仅仅能够发现缺陷，无法帮助开发提供有价值的代码层级的数据。通过智

能缺陷定位技术，测试工程师通过几组用例，例如有的正确，有的失败，尤其是输入差别不大，让被测程序表现为正确和失败的情况。精准测试通过功能上的状态以及用例对应的内部执行代码逻辑的差异分析，可以直接分析出现问题的代码，随后按照可疑度进行排序。如图：

通过测试人员在功能测试阶段标记的用例执行状态，以及软件示波器自动记录的程序 运行频谱，自动分析缺陷的出现的代码块。

（1）对于同类测试用例，经过多组测试可给出非常有效的结果。

（2）列出的可疑代码，可直接通过测试过程给出，提升测试的价值及产出。

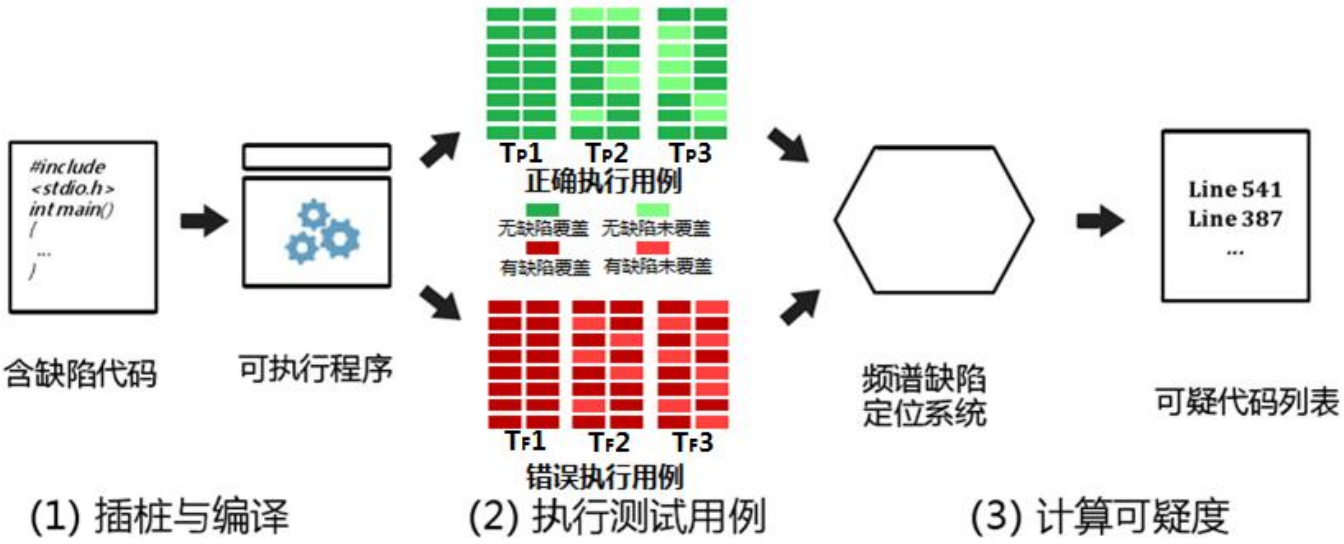


图 4.2.4-1 通过功能测试频谱法分析进行智能缺陷定位

选择可疑度算法、得到可疑度高的代码块，关联源码后，可根据代码可视化查看具体位置。可疑度计算有一个公式，并不复杂，通常每个代码块有 2 个变量，四种状态值。分别是：是否执行、是否通过，这样每代码块都有一个可疑度值。

星云精准测试提供 3 种常用计算公式，供大家参考。

Ochiai

$$\frac{a_{ef}}{\sqrt{(a_{ef}+a_{nf})*(a_{ef}+a_{ep})}}$$

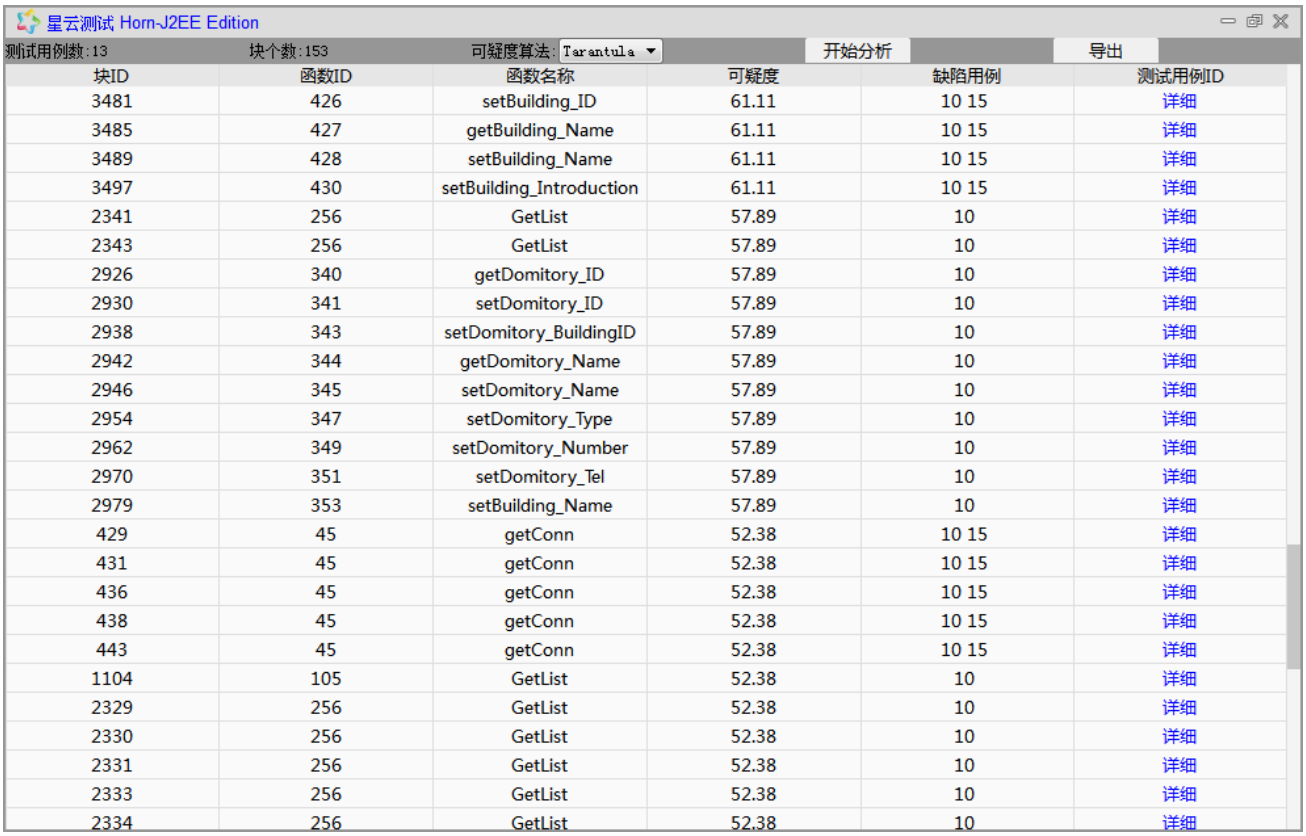
Jaccard

$$\frac{a_{ef}}{a_{ef}+a_{nf}+a_{ep}}$$

Tarantula

$$\frac{\frac{a_{ef}}{a_{ef}+a_{nf}}}{\frac{a_{ef}}{a_{ef}+a_{nf}}+\frac{a_{ep}}{a_{ep}+a_{np}}}$$

其中 aep 表示通过且覆盖到该块的测试用例的个数、anp 表示通过且未覆盖到该块的测试用例的个数、aef 表示未通过且覆盖到该块的测试用例的个数、anf 表示未通过且覆盖到该块的测试用例的个数。结果表示该块的可疑度。



块ID	函数ID	函数名称	可疑度	缺陷用例	测试用例ID
3481	426	setBuilding_ID	61.11	10 15	详细
3485	427	getBuilding_Name	61.11	10 15	详细
3489	428	setBuilding_Name	61.11	10 15	详细
3497	430	setBuilding_Introduction	61.11	10 15	详细
2341	256	GetList	57.89	10	详细
2343	256	GetList	57.89	10	详细
2926	340	getDomitory_ID	57.89	10	详细
2930	341	setDomitory_ID	57.89	10	详细
2938	343	setDomitory_BuildingID	57.89	10	详细
2942	344	getDomitory_Name	57.89	10	详细
2946	345	setDomitory_Name	57.89	10	详细
2954	347	setDomitory_Type	57.89	10	详细
2962	349	setDomitory_Number	57.89	10	详细
2970	351	setDomitory_Tel	57.89	10	详细
2979	353	setBuilding_Name	57.89	10	详细
429	45	getConn	52.38	10 15	详细
431	45	getConn	52.38	10 15	详细
436	45	getConn	52.38	10 15	详细
438	45	getConn	52.38	10 15	详细
443	45	getConn	52.38	10 15	详细
1104	105	GetList	52.38	10	详细
2329	256	GetList	52.38	10	详细
2330	256	GetList	52.38	10	详细
2331	256	GetList	52.38	10	详细
2333	256	GetList	52.38	10	详细
2334	256	GetList	52.38	10	详细

图 4.2.4-2　智能缺陷定位展示

4.3 敏捷迭代

4.3.1 敏捷迭代下多版本白盒测试数据的聚合

白盒覆盖数据通常与代码先关，而敏捷环境下代码每天都会发布几个版本，代码变更后白盒数据就无效了。

星云测试结合目前快速迭代的开发模式，通过分析代码增量，结合不同版本覆盖率，支持累计的合并计算，将多个敏捷局部测试的数据汇总到最新代码视图上统一展示。

星云精准测试的“敏捷环境下多版本白盒测试数据的聚合”功能，可以通过内部累加一个测试周期内的总体覆盖，并在最新代码视图上投影。用户可以看到在一个敏捷周期内的总体覆盖情况，虽然每个敏捷版本可能只是关注某一部分功能。例如一个函数如果一直从某个版本后一直未发生代码变化，那么从变化点以后的覆盖率就可以累加而之前就丢弃掉。

星云精准测试-敏捷环境下多版本白盒测试数据的聚合如图所示。

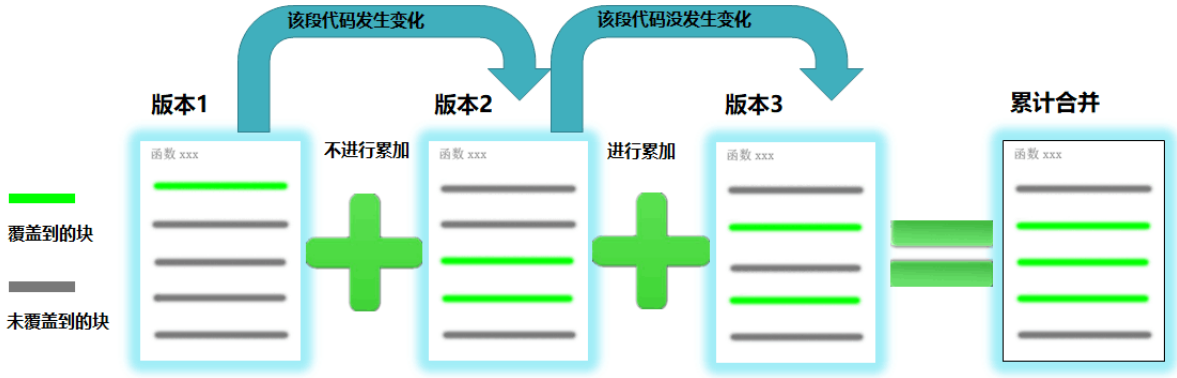


图 4.3.1-1 敏捷环境下多版本白盒测试数据的聚合

精准测试所有数据分析结果都特别考虑了快速迭代的开发模式。

（1）所有版本的测试数据，支持累计的合并计算，将多个敏捷局部测试的数据汇总到最新代码视图上统一展示。

（2）可以任意选择版本合并，观察任何时间节点的累计数据。

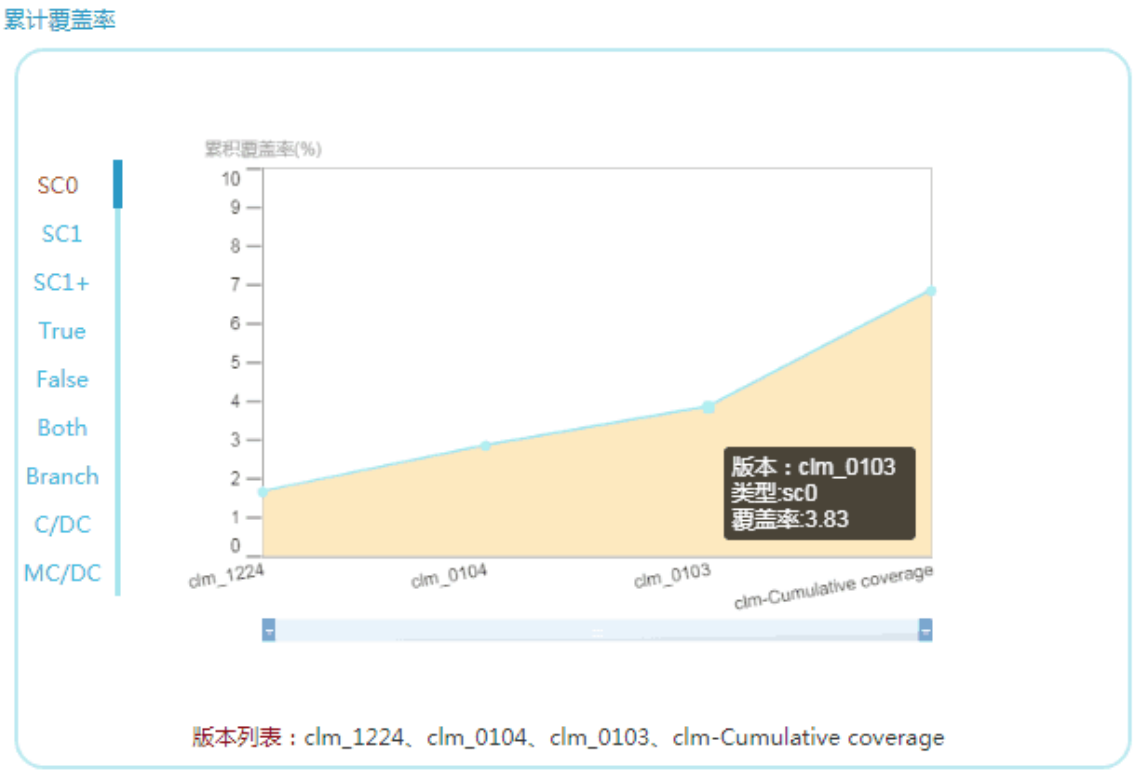


图 4.3.1-2 敏捷环境下多版本白盒测试数据的报表

4.3.2 聚类分析

星云精准测试提供的聚类分析功能，根据测试用例的函数执行剖面的向量化信息，对测试用例进行精确的空间距离计算后执行聚类分析。聚类结果可以分析被错误执行的用例，例如不相关的功能点聚类到一起，则说明其测试执行可能存在错误。

另外也可以辅助找到缺陷分布的密集区域。大部分情况下，缺陷分布会呈现 2/8 的聚集特性。在时间紧张的情况下，我们可以通过聚类结果，每个类选取中心点以及周边几个用例。如果没有问题，就可以去测试其他聚类，如果发现一个类缺陷概率高，那么这个类就需要进行重点测试。通过聚类结果也可以分析测试用例的分布密度等信息，辅助进行测试决策。

星云精准测试-通过用例聚类分析识别缺陷密集分布区域

（1）通过聚类结果给出测试密度，聚类圈中的密度越高，则说明该功能模块测试越密集

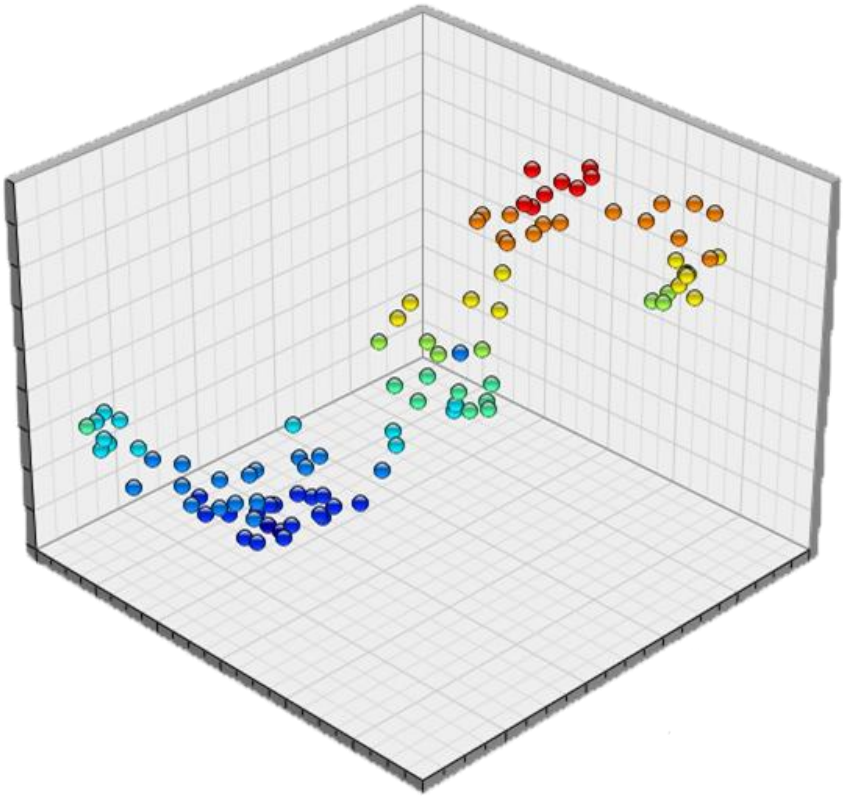


图 4.3.2-1　测试密度

（2）聚类结果可以分析被错误执行的用例，例如不相关的功能点别聚类到一起，则说明其测试执行可能存在错误。

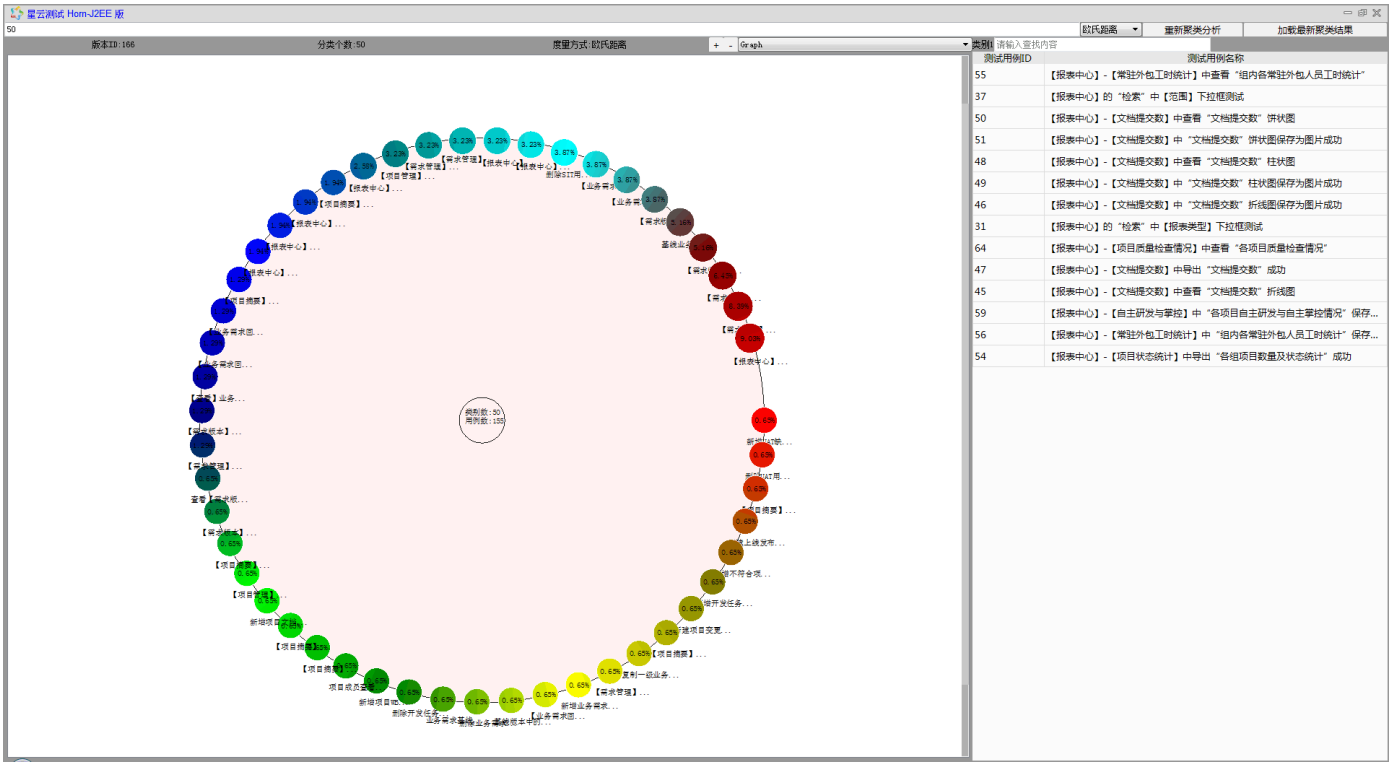


图 4.3.2-2 聚类分析

（3）从类中检出中心点测试用例以及随机的其他用例，可以快速确定类中是否存在较多缺陷，快速定位缺陷的分布，并进行重点测试。

4.3.3 漏洞检出

星云精准测试，结合代码结构和动态数据综合分析，通过计算直接筛选出潜在的高危测试漏洞，可以在短期内确定高危漏测模块。针对性的解决方案，帮助用户快速找到严重缺陷。

当测试时间不充分的时候，首先可以先看测试漏洞列表。列表里将显示通过静态信息和动态信息计算，得到的最高风险的漏测点模块。我们通过复杂度进行计算，复杂度高的模块。一般来讲是重要模块并且逻辑复杂，如果动态覆盖比较低，将被筛选出来。

第二，处于调用和被调用中间的模块，因为属于中间关键模块，我们也会计算它的扇入扇出和动态覆盖率信息。如果比率很高，也会被认为是高风险模块筛选、高亮标示

出来。

从测试效率角度考量，我们建议在基本的功能黑盒测试完成后，先看这些高危模块，补充他们的覆盖率。在时间不充分的时候，优先测试这些高危模块。星云精准测试-结合代码结构与动态数据的测试漏洞检出。

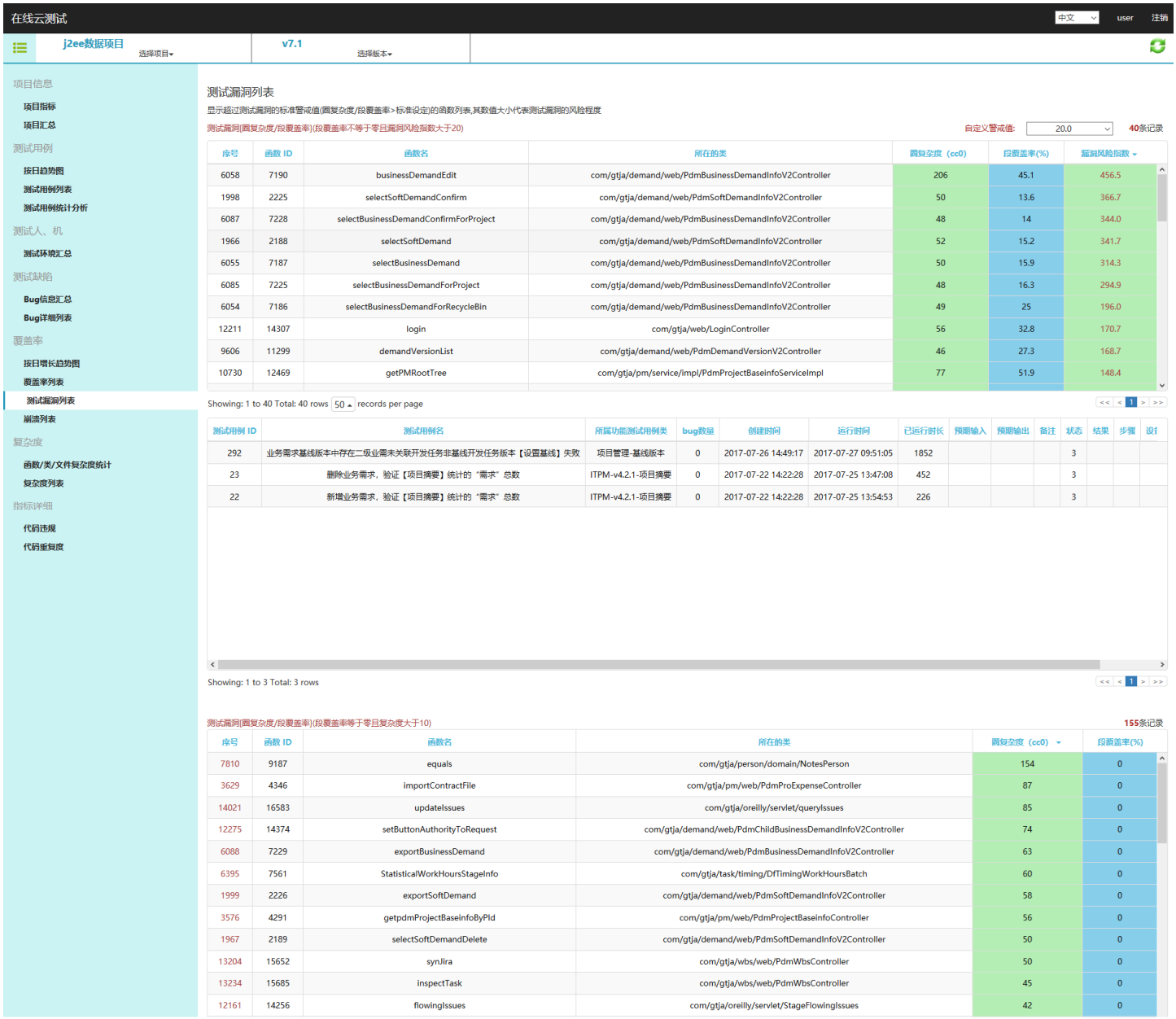


图 4.3.3 漏洞检测列表

4.3.4 精准测试与自动化测试对接

星云精准测试提供了一个通用的自动化接口调用包，为众多的自动化工具进行调用，可以无缝的对接各种主流的自动化工具，测试人员无需改变原有的测试脚本流程，即可执行原有的自动化并在星云精准测试软件示波器中自动建立测试用例并且与执行的代码进行相关联。

星云精准测试与自动化对接还能通过其精准测试的回归、新增代码覆盖率进行相结合使用达到新版本发布全自动的回归测试。

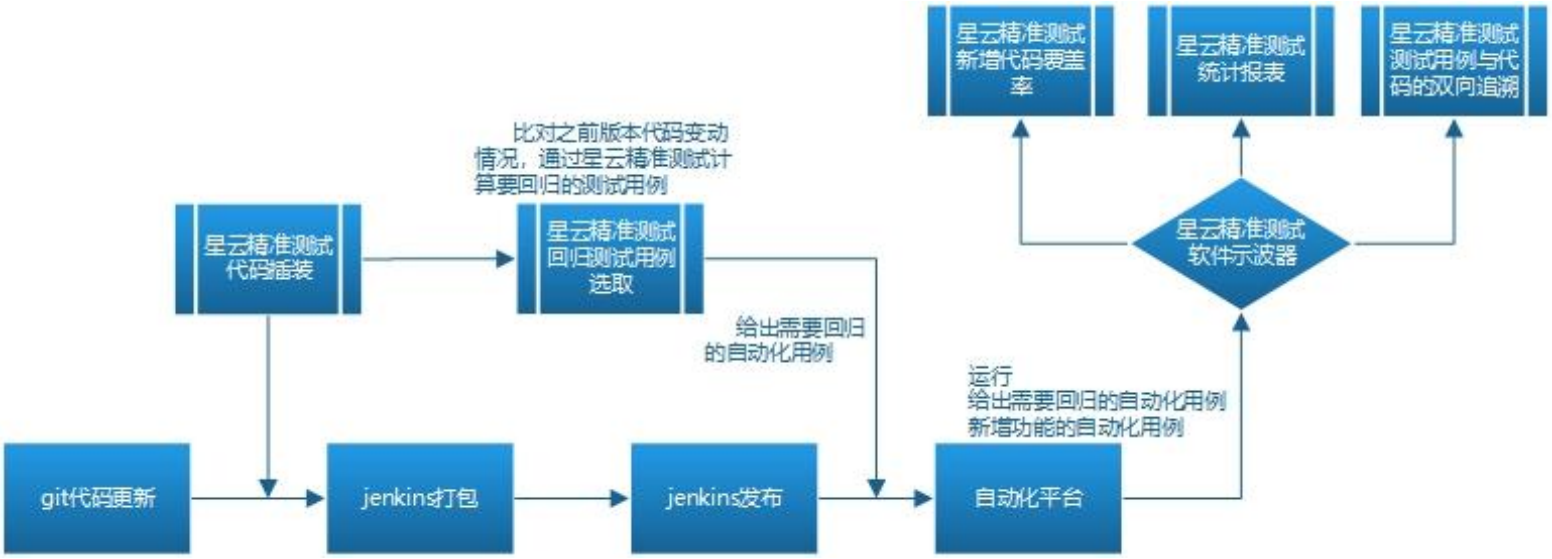


图 4.3.4 星云测试自动化流程

4.3.5 最小测试用例集

星云精准测试通过每个测试用例对应的代码进行统计计算，给出测试用例之间的冗余部分，即满足当前代码覆盖率所需要的运行的最小测试用例集合，主要用于对项目后期大量增长的自动化测试用例进行评审操作，从而降低对自动化用例的维护成本。

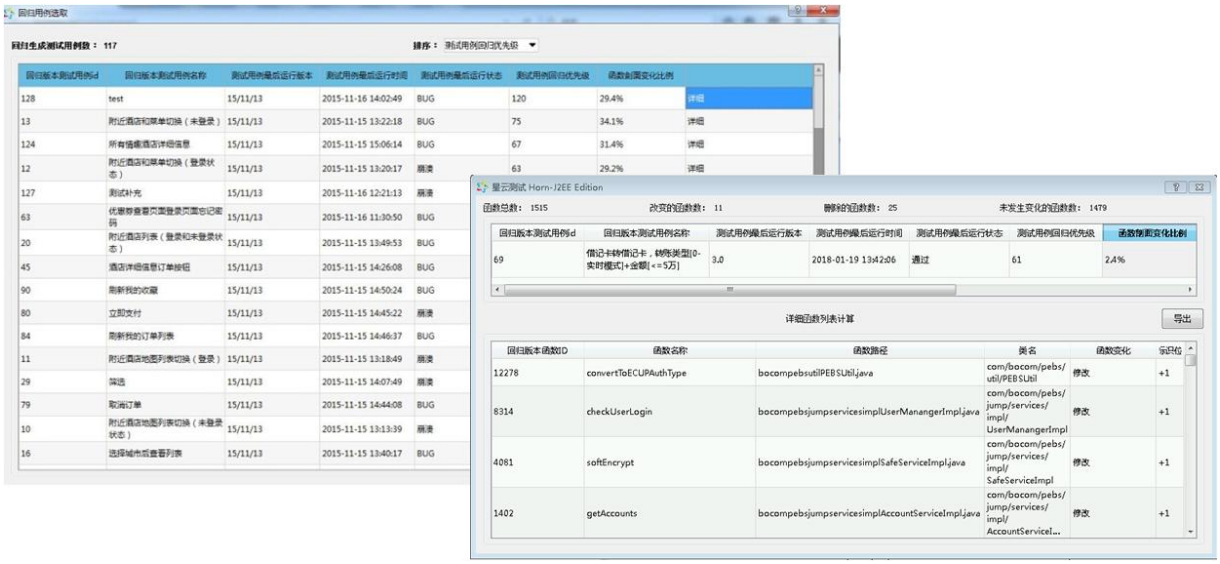


图 4.3.5 星云测试最小测试用例集

4.4 团队管理

4.4.1 精准测试的企业私有云可信化报表

星云测试提供云报表，来实时精准的追踪测试进度。星云测试 web 端的报表系统，当客户端录入测试用例并采集数据后，将在 web 端产生实时的、具备高可信度的测试情况报表。

该报表与普通的测试管理系统不同：普通的测试管理系统有人为录入数据的情况，数据本身的真实性就没办法保证。精准测试提供的报表，底层数据来自于执行测试用例时候代码数据的采集，通过专用接口上传，完全无法篡改和伪造。星云精准测试-企业私有云端实时、精准、可信质量跟踪。

（1）通过浏览器登录测试系统，选择需要跟踪的项目，就可以实时的对整个测试的质量、进度、人员进行精准的分析和管理。

(2) 企业私有云端管理系统展示的数据基于精准测试数据的分析，所有数据原生精确，支持移动测试+本地测试。

(3) 测试团队、开发团队、甲方负责人等多种角色都可以登录系统，从各个层面对测试、软件质量进行分析。

图 4.4.1-1 项目汇总展示

星云精准测试项目汇总中，包含了项目信息、版本信息、测试汇总信息、测试过程监控趋势图、测试设备组成和分布图、各版本覆盖率汇总图、复杂度汇总图等。

从另外一个角度来看，精准测试大企业版本也可以让不同区域、不同时间的测试人员实现协同测试与协同管理，最终达到多人同地测试、多人异地测试、数据实时汇总共

享与追踪、测试过程与完成度一目了然的管理目标。

项目信息

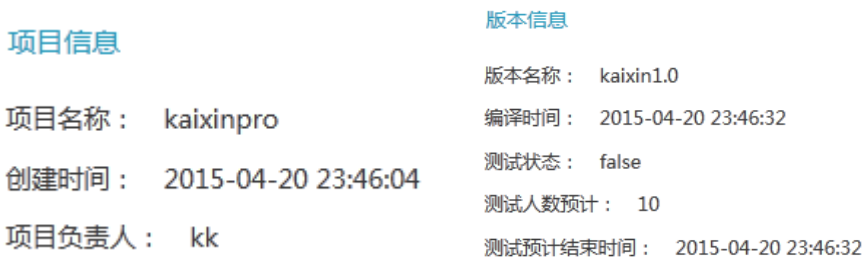


图 4.4.1-2 项目信息与版本

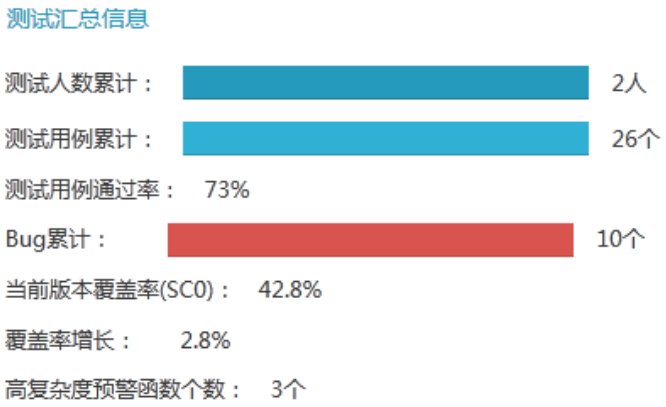


图 4.4.1-3 测试信息汇总

测试用例通过率：无 BUG 的测试用例

BUG 累计：测试用例运行完毕后提交的 BUG 数

当前版本覆盖率(SC0)：(执行过可见段数/可见段数)*100%的比例

覆盖率增长：相比前一天的 SC0 增长差值

高复杂度预警函数个数：高复杂度的函数个数

4.4.2 精准测试的企业私有云-测试效率的直观展示

星云测试报告可直观分析每天的测试效率，通过代码模块和复杂度关系图，看到函数群落测试情况分布及趋势，可直观精准识别系统测试所处阶段。

例如通过每日增长覆盖，我们可以看到整个团队的工作效率。一般系统都是在刚开始测试的时候覆盖率增长快，到了黑盒测试的瓶颈点，就上升很慢了。管理者可以通过报表，清晰的看到整个团队的效率趋势，并采取相关策略。

覆盖率和复杂度的关系图，可以很直观的看到测试的质量。如果测试不充分的时候，复杂度高的模块通常覆盖率都比较低，分布自一个左上角的区域，当测试深入进行，这些点就会向右侧移动。管理者可以非常直观的看到系统测试的充分程度和上线的质量把握。

（黑盒效率换挡点+测试深度运动趋势）

（1）累计覆盖率增长视图直观分析每天的测试实效以及确定从系统黑盒测试转换到精准测试的最佳时点。

（2）代码模块覆盖率和复杂度关系图，看到函数群落测试情况分布以及运动趋势，可以直观精确识别系统测试所处阶段。

图 4.4.2-1 覆盖率每日增长趋势图与黑盒测试瓶颈

图 4.4.2-2　测试效率换档点与测试深度趋势观察表

函数|类|文件复杂度和覆盖率关系图,以散点图的形式展示各种复杂度和覆盖率的关系，更清晰的掌握各种程度覆盖率的分布。

图中点：代表每个函数，点击后可以看到相关信息。

纵向：代表复杂度不同级别，取各种覆盖率值，点击自动进行切换到相应的复杂度

横向：代表覆盖率不同级别，取各种覆盖率值，点击自动进行切换到相应的覆盖率

红圈 62：代表当前版本所选的复杂度类型中最高复杂度值

篮圈 100：代表当前版本所选的覆盖率类型中最高覆盖率值

4.4.3 精准测试的企业私有云-测试用例排行图

测试用例排行图，可直观展示参与测试工程师所执行的用例数、通过率和缺陷率，真实记录并分析每个测试用例的实效性。星云精准测试-测试工程师实效精准分析系统，将参与的测试工程师所执行的用例从逻辑覆盖映射到代码覆盖，真实记录并分析每个互联网测试参与者的工作实效。以逻辑覆盖为基准的而不是用例数量为考核标准。

测试用例排行图

测试用例列表

对于测试工程师利用tt录制的测试用例个数累计给出排行榜,快速的找到表现突出的工程师

Pass Fail NotRun

共15条记录

排名	用例量	所属工程师编号	工程师姓名	贡献度
1		79	测试a部014	38.6%
2		15	测试a部006	30.5%
3		10	测试a部001	26.6%
4		76	测试a部011	24.5%
5		14	测试a部005	23.7%
6		11	测试a部002	22.4%
7		77	测试a部012	16.5%
8		13	测试a部004	12.3%
9		12	测试a部003	10.9%
10		16	测试a部007	10.6%

Showing: 1 to 10 Total: 15 rows 10 records per page

<< < 1 2 > >>

图 4.4.3 测试用例排行图

4.5 知识库累积

4.5.1 精准测试数据的价值

星云测试采集的测试数据和插装后分析到的静态结构信息将作为大型企业系统大数据分析的基础数据。

星云精准测试-测试数据价值

（1）代码级的程序静态信息以及测试用例对应的海量动态测试的数据，这些多维度数据将作为大型企业系统大数据分析的基础数据。

（2）对本企业大量软件质量数据进行挖掘和分析，找到相关质量技术标准衡量的合理区间，避免常规错误。

（3）通过数据分析确定优异的开发方法和技术构件。

（4）通过质量大数据的分析结果，选择更加合理的技术方法，在设计阶段避免已知的缺陷。

4.5.2 精准测试智能回归测试用例智能选取

通过对测试数据分析，星云精准测试可自动筛选测试用例。在回归测试时，大大减少回归测试的时间及风险，如图通过对版本 a 和版本 b 测试数据分析，通过历史数据的比对，可在版本 c 上全自动得到回归优先级高的用例。

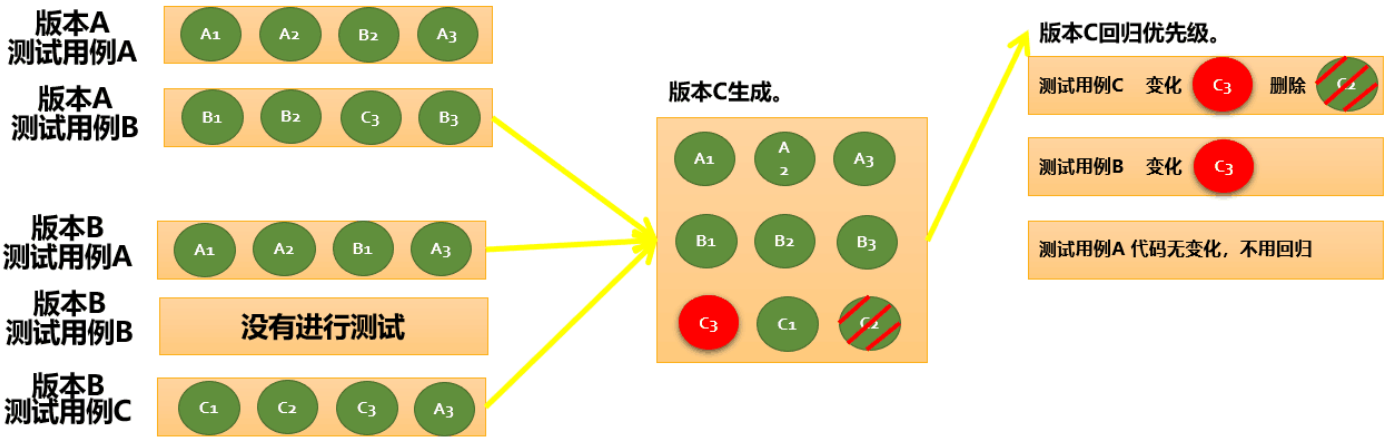


图 4.5.2-1 精准测试智能回归测试用例的选取

（1）适应快速的版本迭代周期，适应庞大的工程项目。

（2）在回归测试时，自动筛选测试用例，大大减少了回归测试的时间以及风险。

（3）降低了传统人工回归分析产生的测试盲点。

（4）精确计算回归用例的权重，测试人员在时间有限的情况下可以重点回归受改动影响最大的用例。

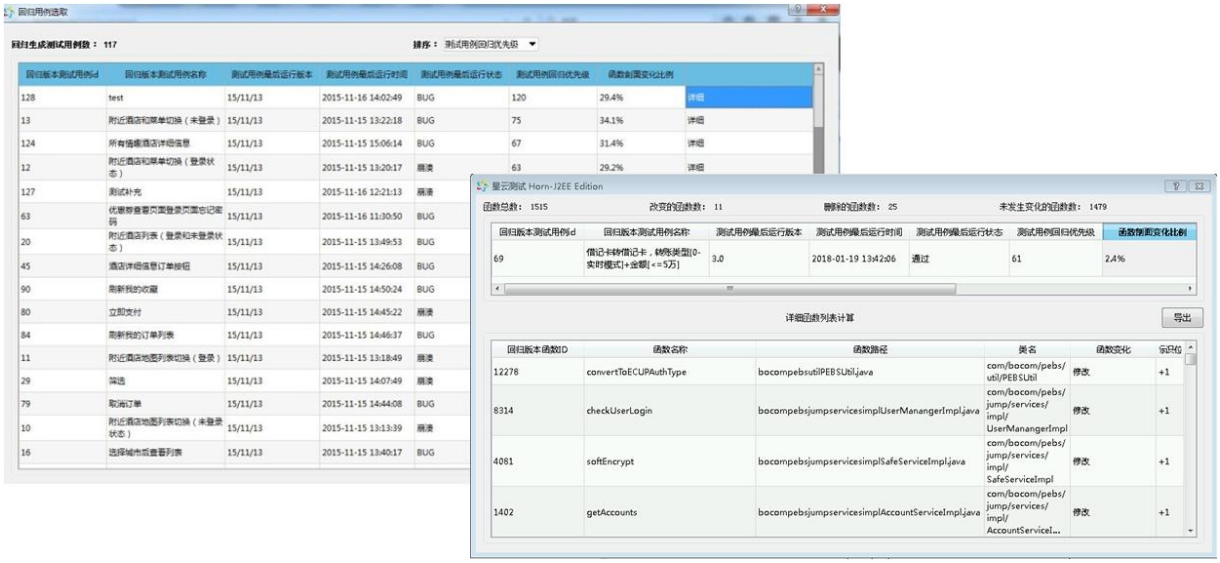


图 4.5.2-2 回归测试用例选取界面

4.5.3 精准测试在回归测试中的性能评估

回归性能，通常一般回归都是基于人对于系统的判断来做的，一般来讲国际上的统计每修改 6 行代码就会引入一个未知的缺陷。由人来进行回归用例集的判断，随着时间的延续，记忆将不可逆转地发生损耗并丢失，加之原团队人员的不断变更，老的系统维护越来越难，修改引入新的缺陷要越来越难风险。

通过星云精准测试企业离线平台，内置程序将持续、高效地全自动记录本企业自有的各系统测试用例和代码的关系数据，不用人工干预和记录。使用一段时间后，企业会得到越来越精确的数据，若加以有效利用，将发挥相关元数据及大数据的爆发性的价值。

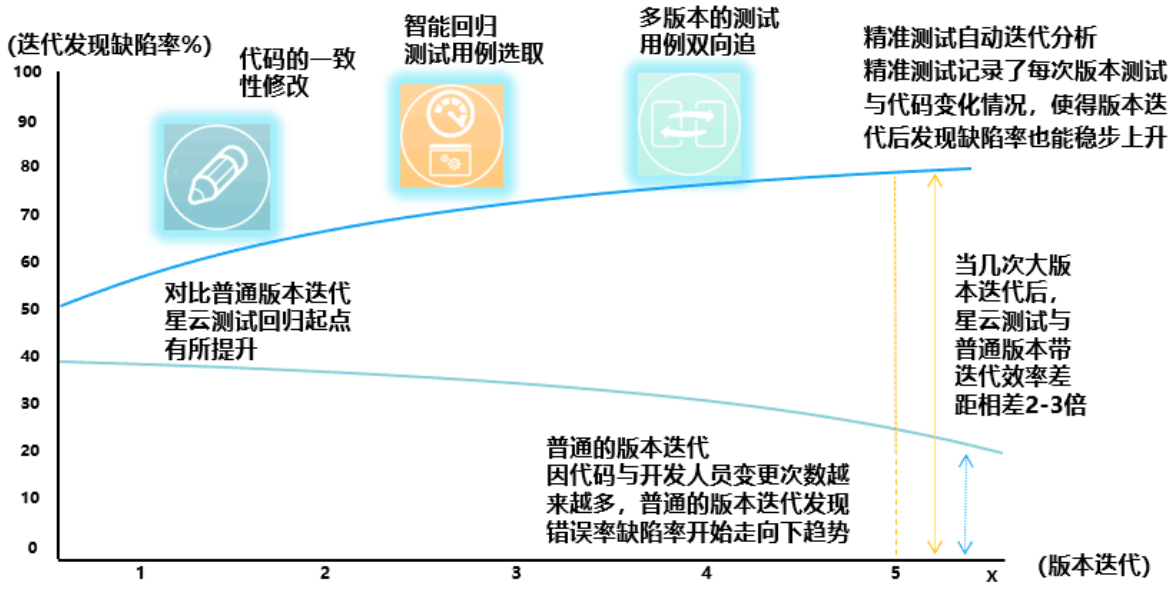


图 4.5.3 智能回归测试用例选取的性能评估

第五章 精准测试的管理报表分析

星云精准测试报表体系，是根据大型企业实际工作中的诉求而设计。报表涉及内容

很多，包括项目信息、测试过程及总结信息、团队效率信息等，数字化展示企业分布式开发与测试过程、验收和维护全过程。它深度解决软件测试短板，数字化跟踪、追溯、输出开发与测试每一步信息。

它把原来很多需要人工录入的数据，由平台代为自动化机器处理，确保所有的数据都是可信的、不可篡改的。它自动把测试用例、测试设备、测试数据、测试人员、测试时间等信息进行关联分析，企业管理人员可以利用机器原生数据，对测试全过程进行精准的数字化管理。

5.1 项目指标

此项中将展示项目中各个指标汇总信息，如：程序代码信息汇总，测试漏洞、程序覆盖率指标、代码违规统计、代码重复度、程度复杂度指标、程序 Crash 情况等。

图 5.1 项目指标汇总图

5.1.1 程序代码信息汇总

程序代码信息汇总中显示当前代码基本信息、代码注释比例、代码可维护性。它能有效地检查出代码的整体状况，并指出相应的薄弱点。

图 5.1.1 程序代码信息汇总

上述指标，需要使用到以下数据并对应映射表来确定：可分析性、可修改性、稳定性、可测试性的最终等级。

5.1.2 程序覆盖率指标

程序覆盖率指标：该值是程序测试过程中代码段的覆盖率数值

图 5.1.2-1 程序覆盖率指标

级别	低	中	优秀	高质量	超高质量
覆盖率百分比（%）	0~30	30~50	50~70	70~90	90~100

图 5.1.2-2 覆盖率百分比

程序复杂度指标：给出程序复杂度相关信息，以及文件、类、函数三个级别复杂度分级柱状图。（基于 CC0 计算）CCO 表示的是圈复杂度

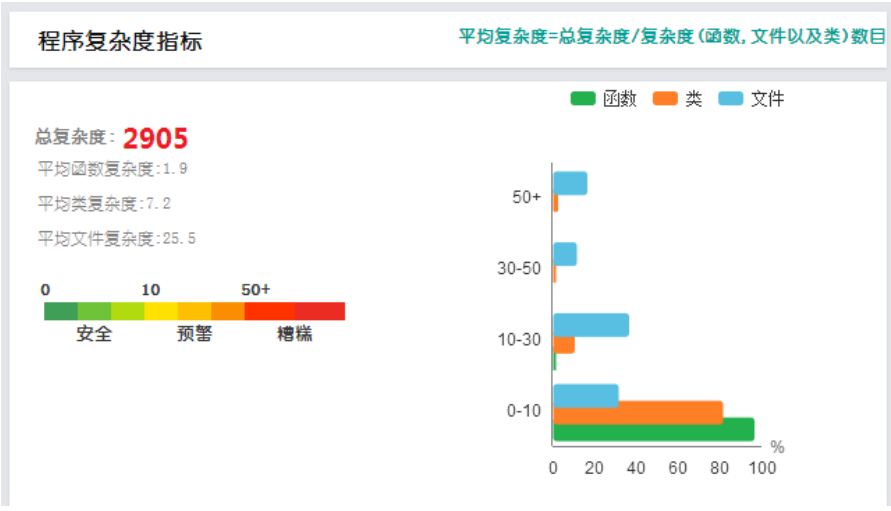


图 5.1.2-3 程序复杂度指标

5.2 测试用例-按日趋势图

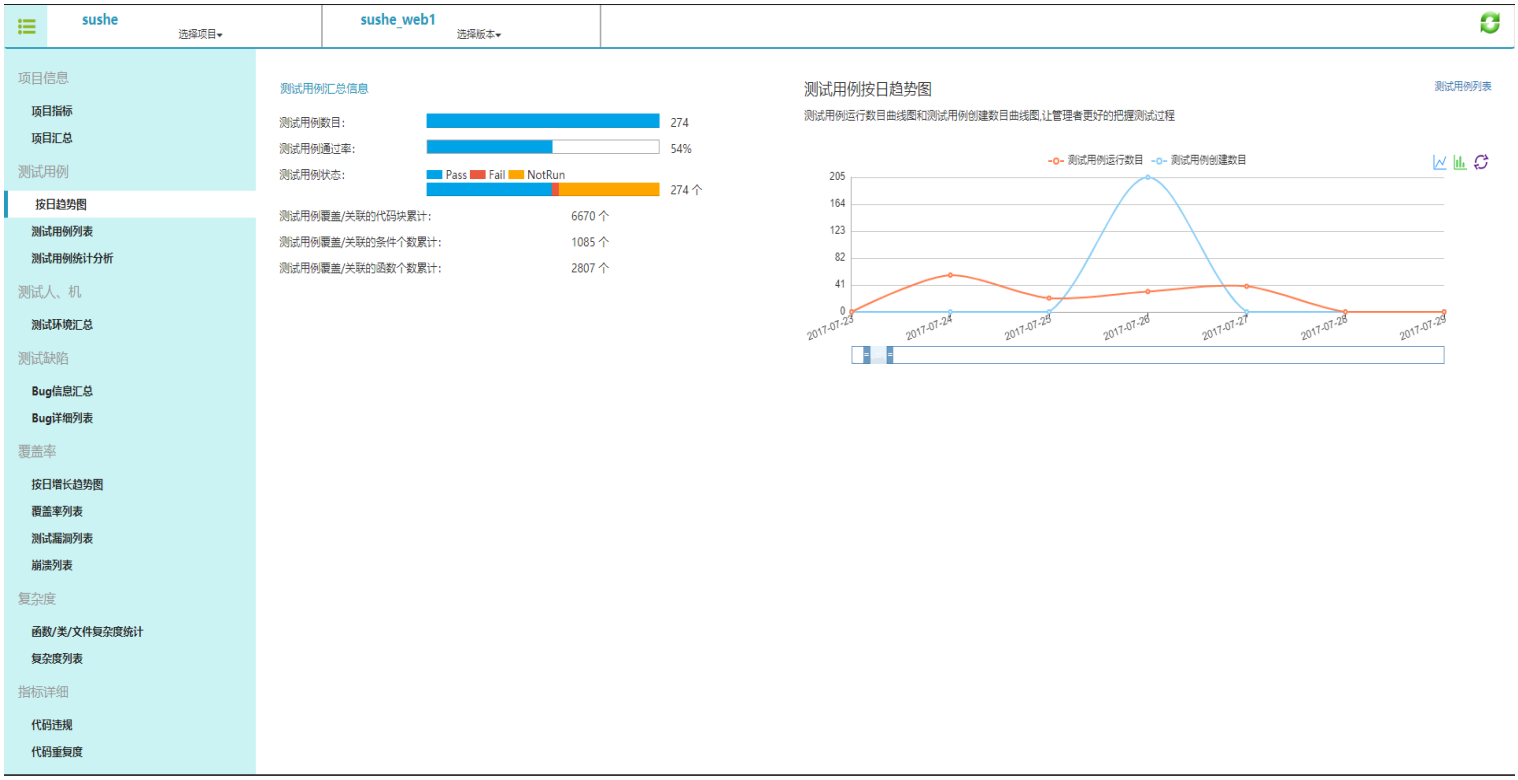


图 5.2 按日趋势图

5.2.1 测试用例汇总信息

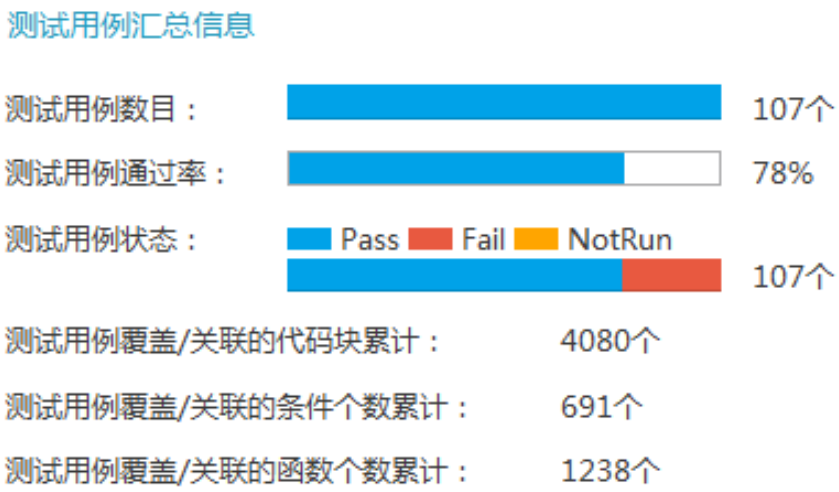


图 5.2.1 项目信息

5.2.2 测试用例按日趋势图

折线图 和柱形图 可以在不同的图形示例中切换，还原 显示默认折线图

纵轴坐标：代表数量，取值范围为自动适应最高值

横轴坐标：代表日期时间

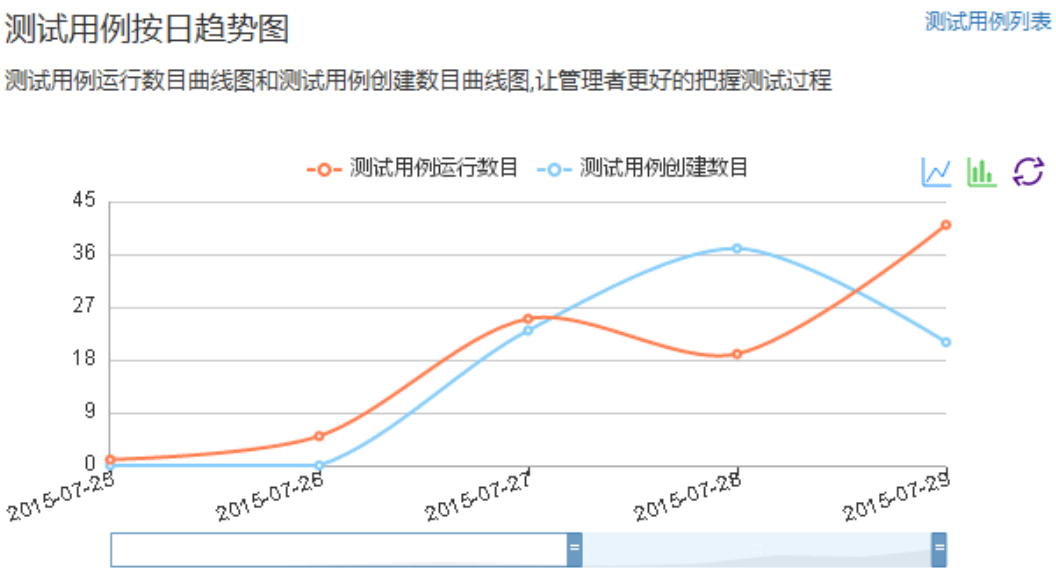


图 5.2.2 测试用例按日趋势图

5.3 测试用例-测试用例列表

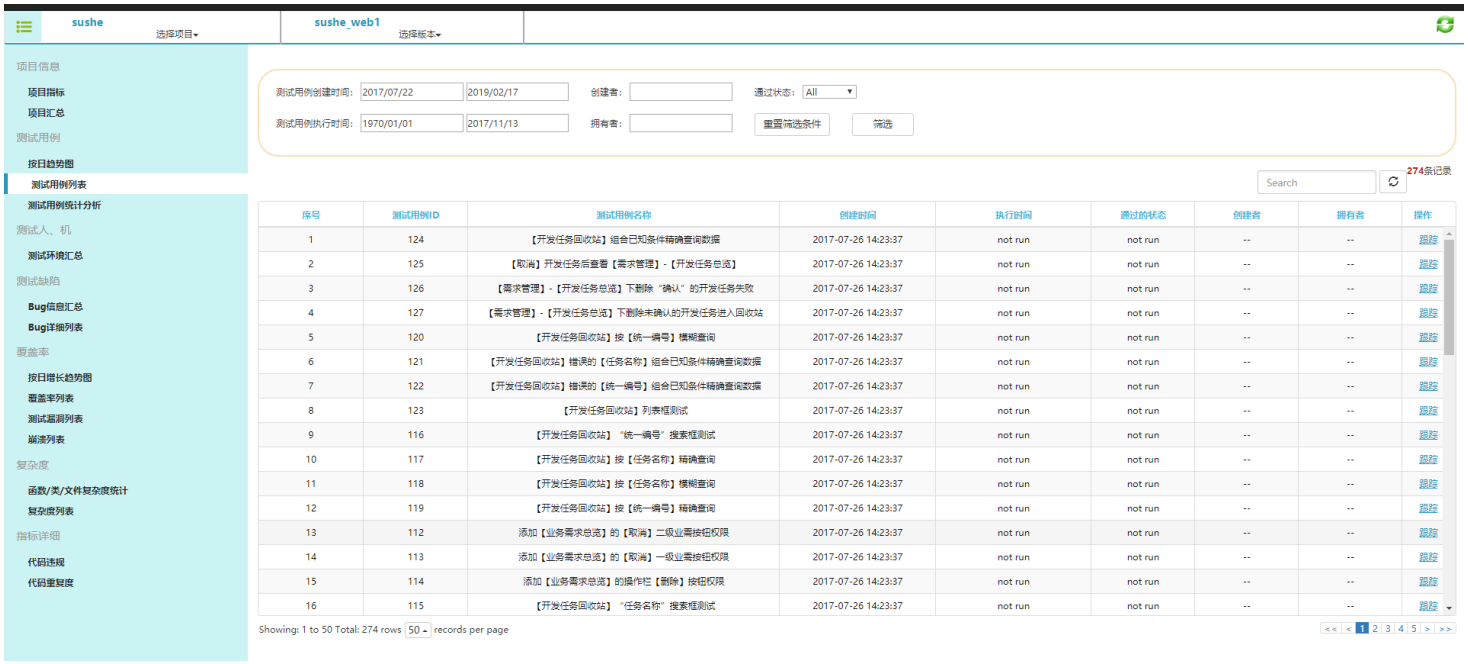


图 5.3-1 测试用例列表的详细信息

显示制作的测试用例的详细信息，包括测试用例的名称、创建时间、执行时间、关联函数、覆盖率占比、运行状态、测试人员等

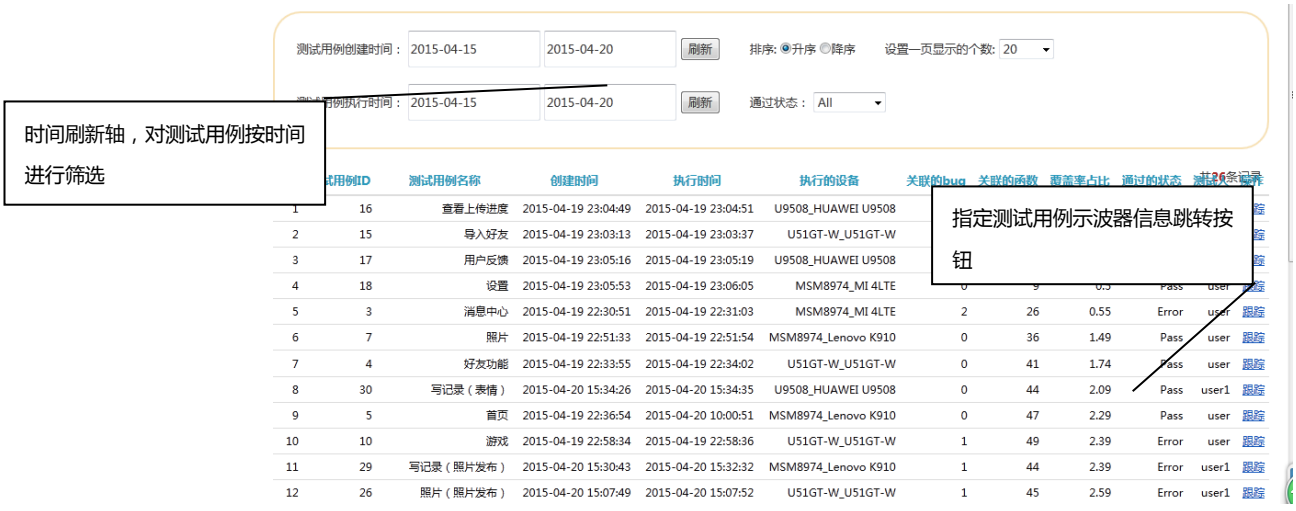


图 5.3-2 测试用例排行与贡献

5.3.1 星云精准测试软件示波器（测试用例跟踪）

5.3.1.1 测试用例描述

对选定测试用例表述其详细信息，包括测试用的所属模块、日期、录制人、执行时间等。

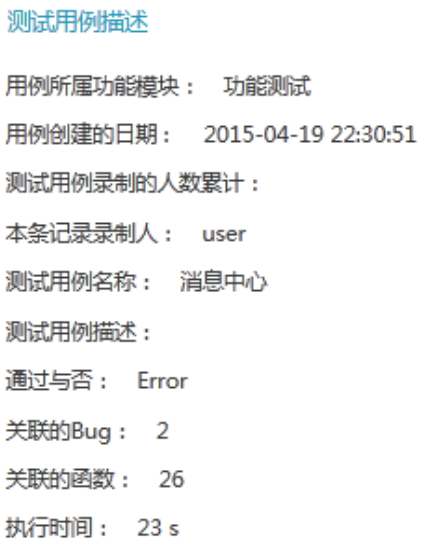


图 5.3.1.1 测试用例描述图

5.3.1.2 录制记录

记录示波器最后的接收的块、条件、函数的总信息，分最新一次和上一次，用于等价类测试比对。

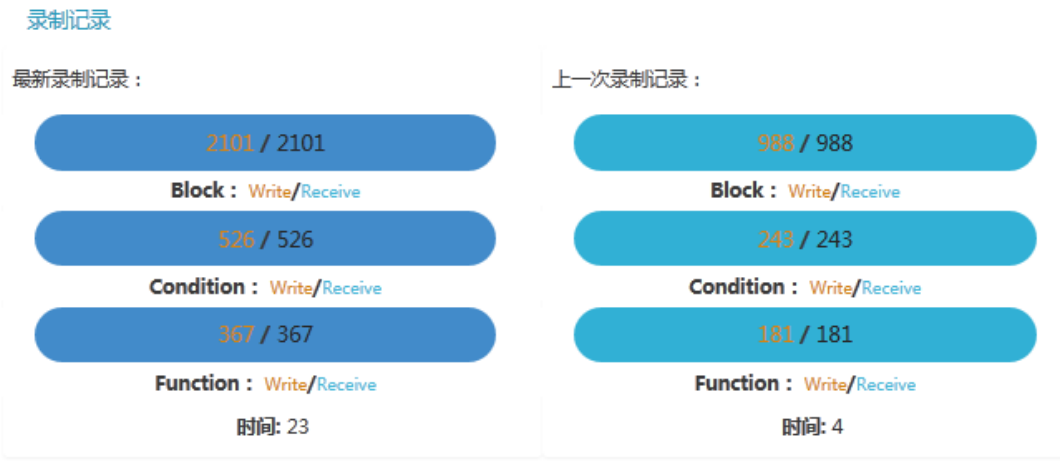
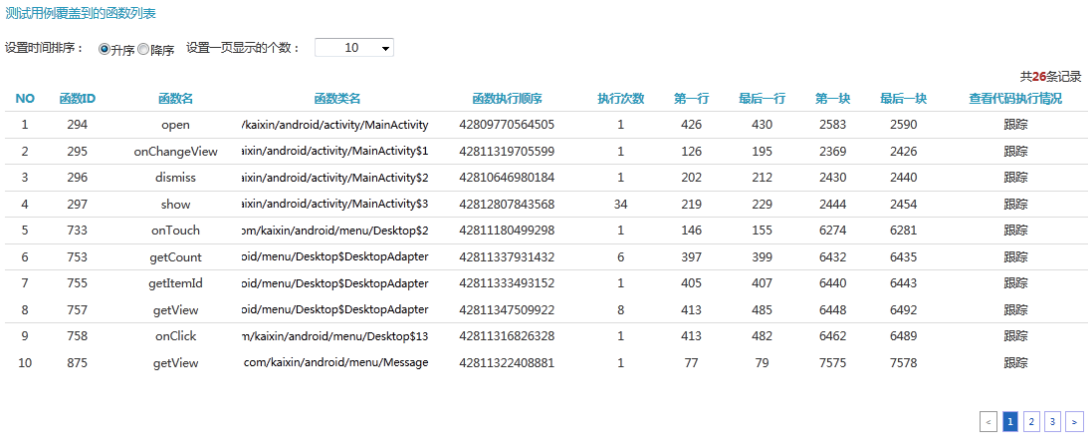


图 5.3.1.2-1 最新与上一次软件示波器接收信息对比

按时间倒序排列示波器接收到的函数，记录程序最后运行的函数状况，用于定位程序错误和测试用例运行过程中的逻辑分析。



NO	函数ID	函数名	函数类名	函数执行顺序	执行次数	第一行	最后一行	第一块	最后一块	查看代码执行情况
1	294	open	/kaixin/android/activity/MainActivity	42809770564505	1	426	430	2583	2590	跟踪
2	295	onChangeView	ıixin/android/activity/MainActivity$1	42811319705599	1	126	195	2369	2426	跟踪
3	296	dismiss	ıixin/android/activity/MainActivity$2	42810646980184	1	202	212	2430	2440	跟踪
4	297	show	ıixin/android/activity/MainActivity$3	42812807843568	34	219	229	2444	2454	跟踪
5	733	onTouch	ɔm/kaixin/android/menu/Desktop$2	42811180499298	1	146	155	6274	6281	跟踪
6	753	getCount	ɔid/menu/Desktop$DesktopAdapter	42811337931432	6	397	399	6432	6435	跟踪
7	755	getItemId	ɔid/menu/Desktop$DesktopAdapter	42811333493152	1	405	407	6440	6443	跟踪
8	757	getView	ɔid/menu/Desktop$DesktopAdapter	42811347509922	8	413	485	6448	6492	跟踪
9	758	onClick	ɲ/kaixin/android/menu/Desktop$13	42811316826328	1	413	482	6462	6489	跟踪
10	875	getView	com/kaixin/android/menu/Message	42811322408881	1	77	79	7575	7578	跟踪

图 5.3.1.2-2 软件示波器接收函数图

5.4 测试缺陷-Bug 信息汇总

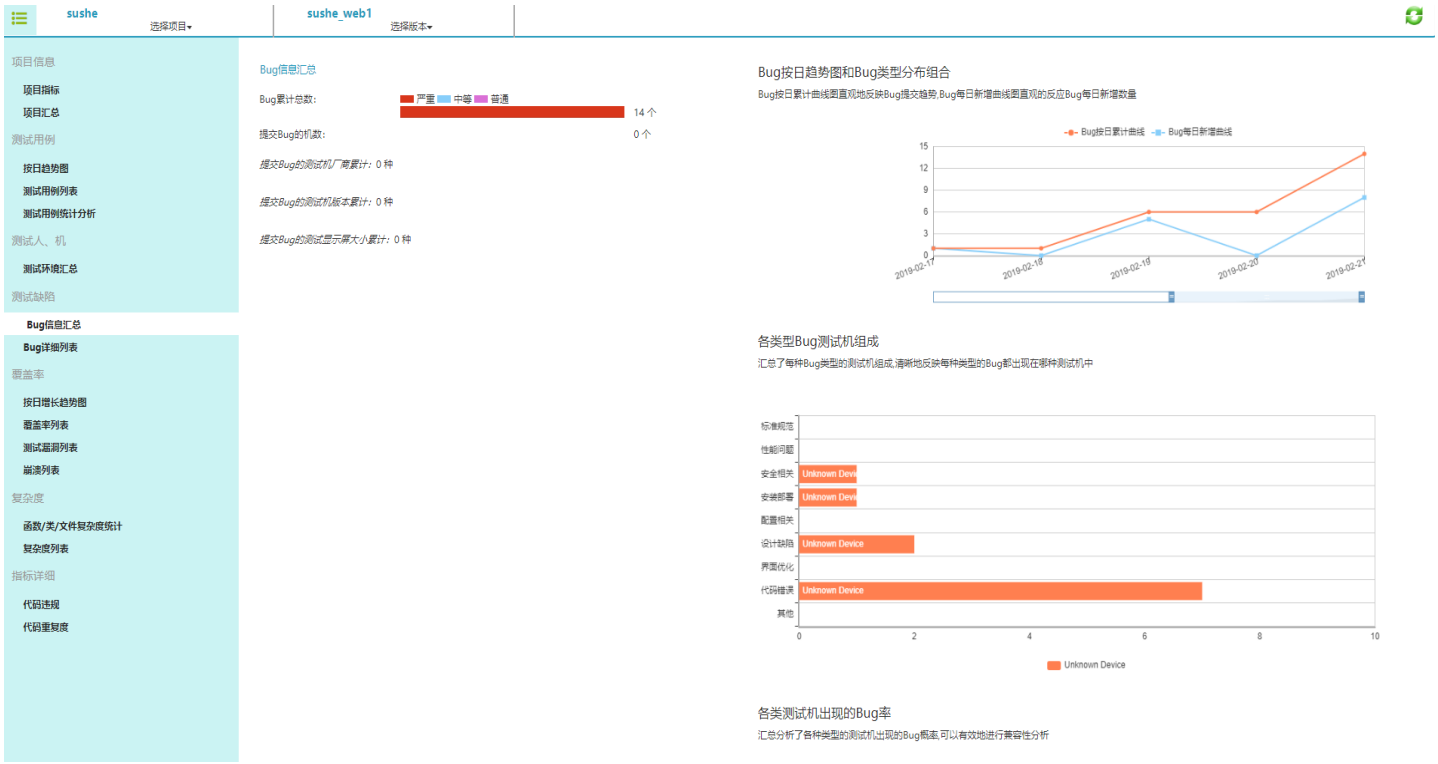


图 5.4Bug 缺陷汇总

5.4.1 Bug 按日趋势图和 Bug 类型分布组合

Bug 按日趋势图直观地反映 Bug 提交趋势。

点击折线图上的点会显示相应的 Bug 组成。

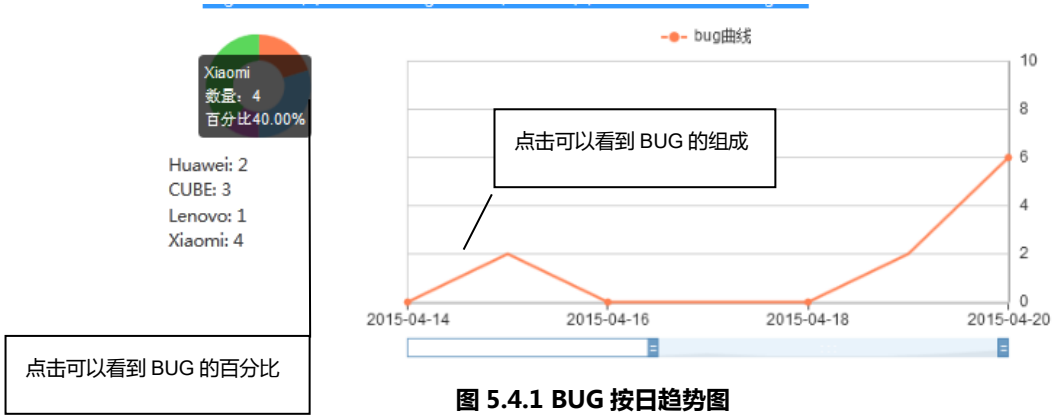


图 5.4.1 BUG 按日趋势图

5.4.2 Bug 提交排行榜

给出提交 Bug 的测试工程师的排行榜。

彩条图可以选择按照 Bug 级别显示，也可以选择按照 Bug 类型显示。

图 5.4.2BUG 提交排行榜

5.5 测试缺陷-Bug 详细列表

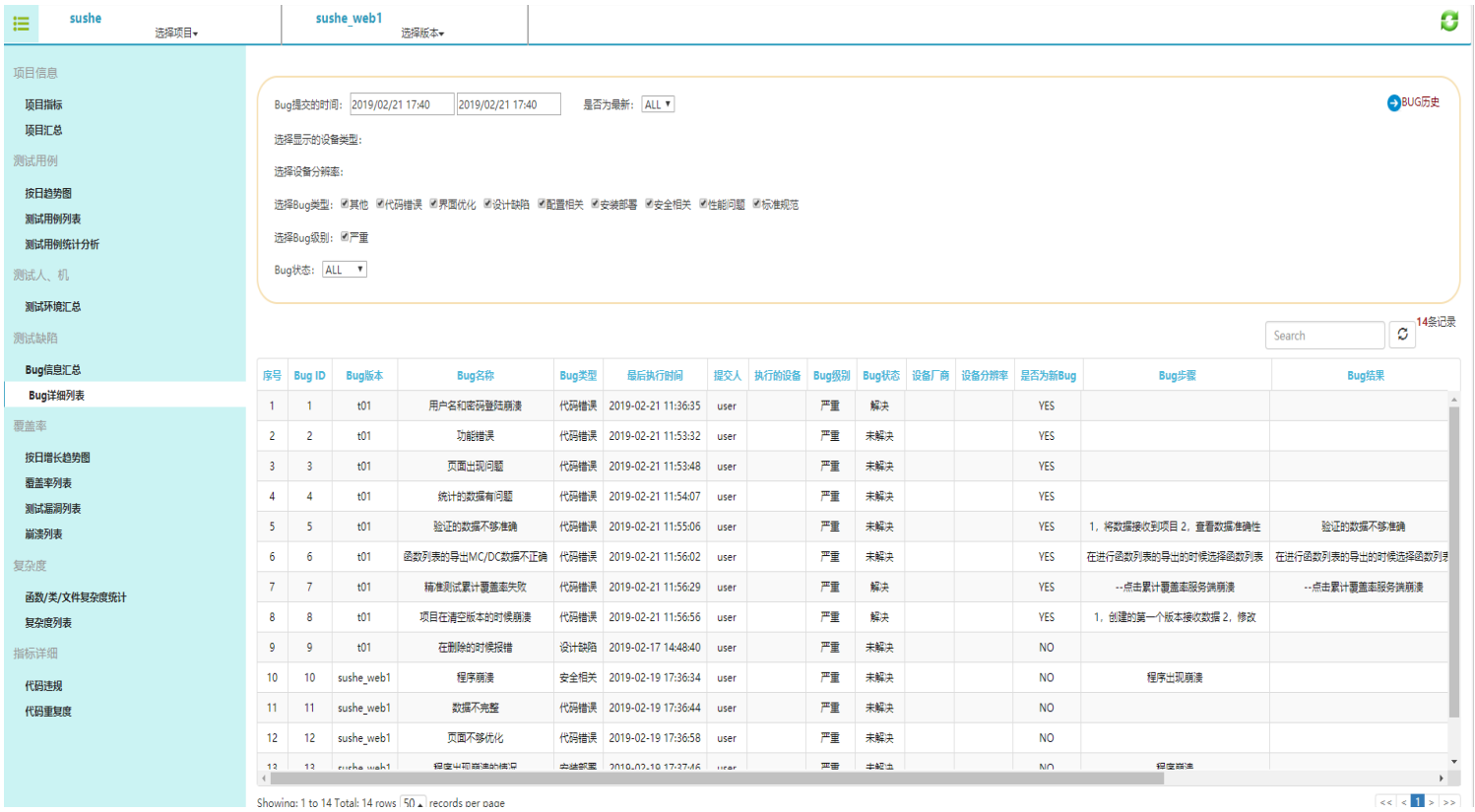


图 5.5-1 bug 详细列表

显示提交的 BUG 的详细信息、包括提交人、BUG 类型、对应的测试用例等。

图 5.5-2BUG 详细信息图

5.6 覆盖率-按日增长趋势图

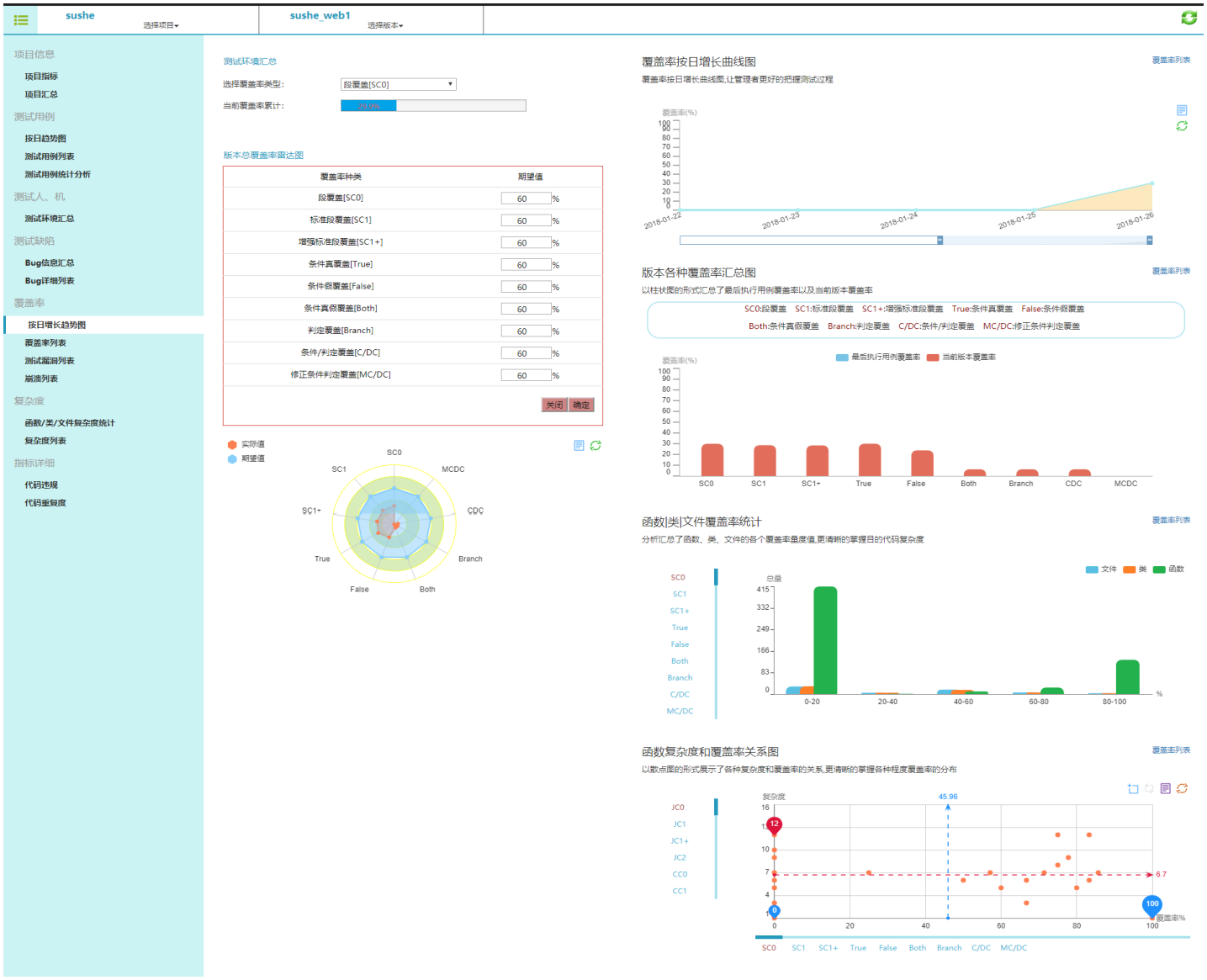


图 5.6 按日增长趋势图

5.6.1 覆盖率信息汇总

可以查看当前版本覆盖率的信息，可以选择多种覆盖率进行查看以及各个测试模块占覆盖率的比。

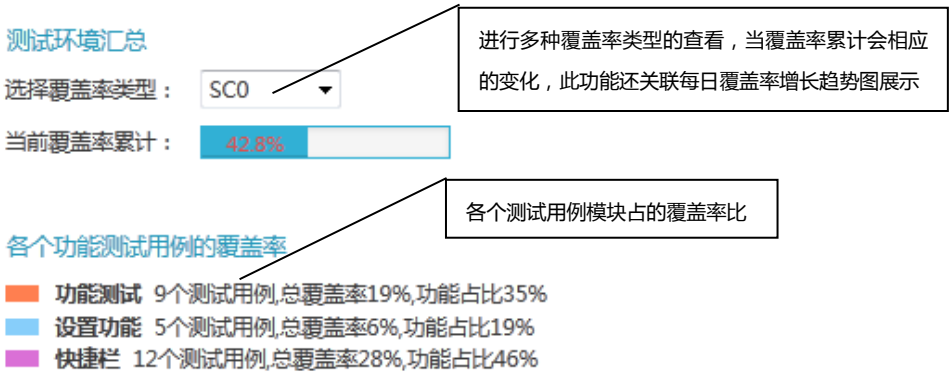


图 5.6.1 覆盖率信息汇总图

5.6.2 覆盖率按日增长曲线图

覆盖率按日增长曲线图，让管理者更好的把握测试过程。

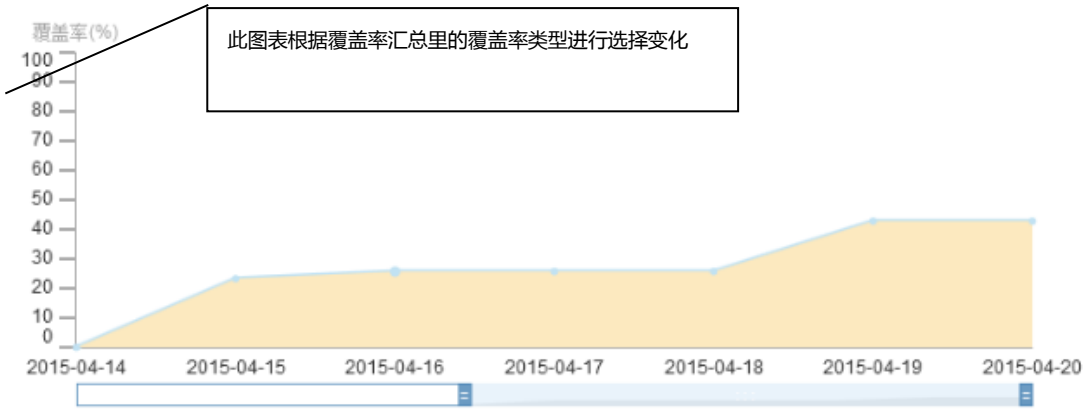


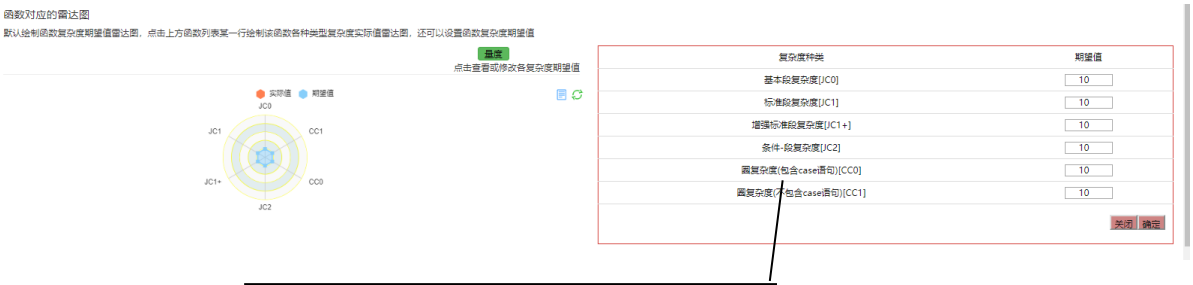
图 5.6.2 每日覆盖率增长曲线图

5.6.3 雷达图

根据项目的需要，测试人员可以自己设置覆盖率的上限，通过雷达图展示是否达到预期。

覆盖率指标是否要每项都到 100%才算测试结束？在覆盖率达标方面，用户可以按每个应用的实际情况进行达标线设置。

给出数字化覆盖率展示的用意在于：让测试人员通过观察，能更好的补充完善测试用例。雷达图的目的，是为了当测试人员观察是否达到预期。



测试人员自己设置覆盖率最大最小值 ,由右侧雷达图进行覆盖率展示

图 5.6.3 覆盖率总雷达图

5.6.4 函数|类|文件覆盖率统计

分析汇总了函数、类、文件的各个覆盖率量度值，更清晰的掌握目标代码覆盖率。

图内容：显示当前版本的文件、类、函数在各个覆盖率之间的分布。

纵向：代表覆盖率的不同级别，取各种覆盖率值，点击后自动进行切换到相应的覆盖率

横向：代表覆盖率%区间，取值 0-20、20-40、40-60、60-80、80-100

总量：代表当前版本的函数总量

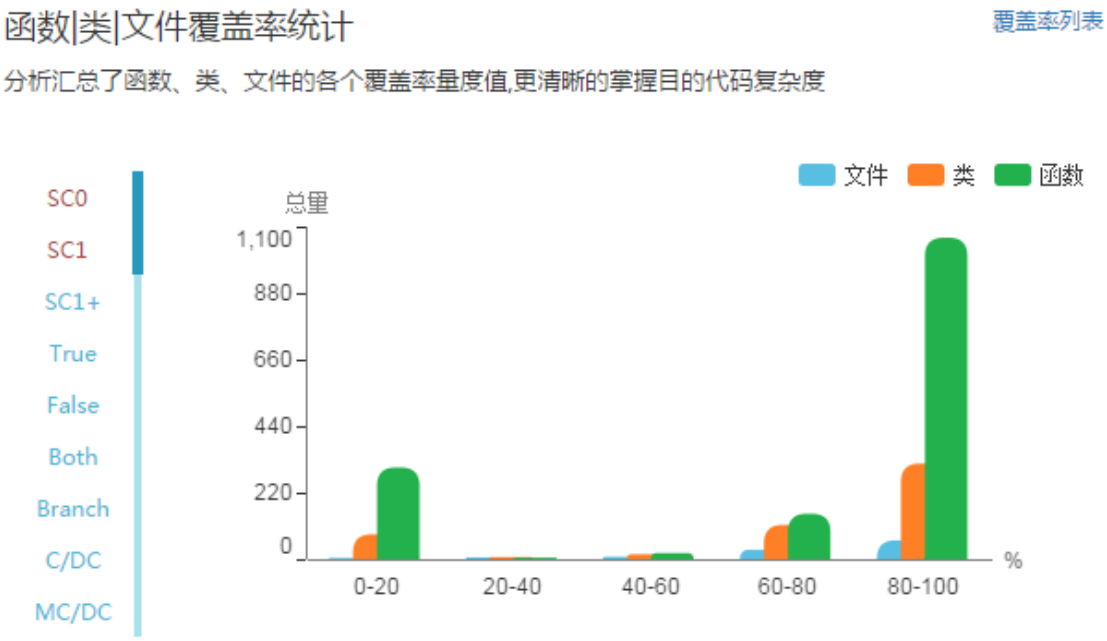


图 5.6.4 函数|类|文件覆盖率统计图

5.7 覆盖率列表

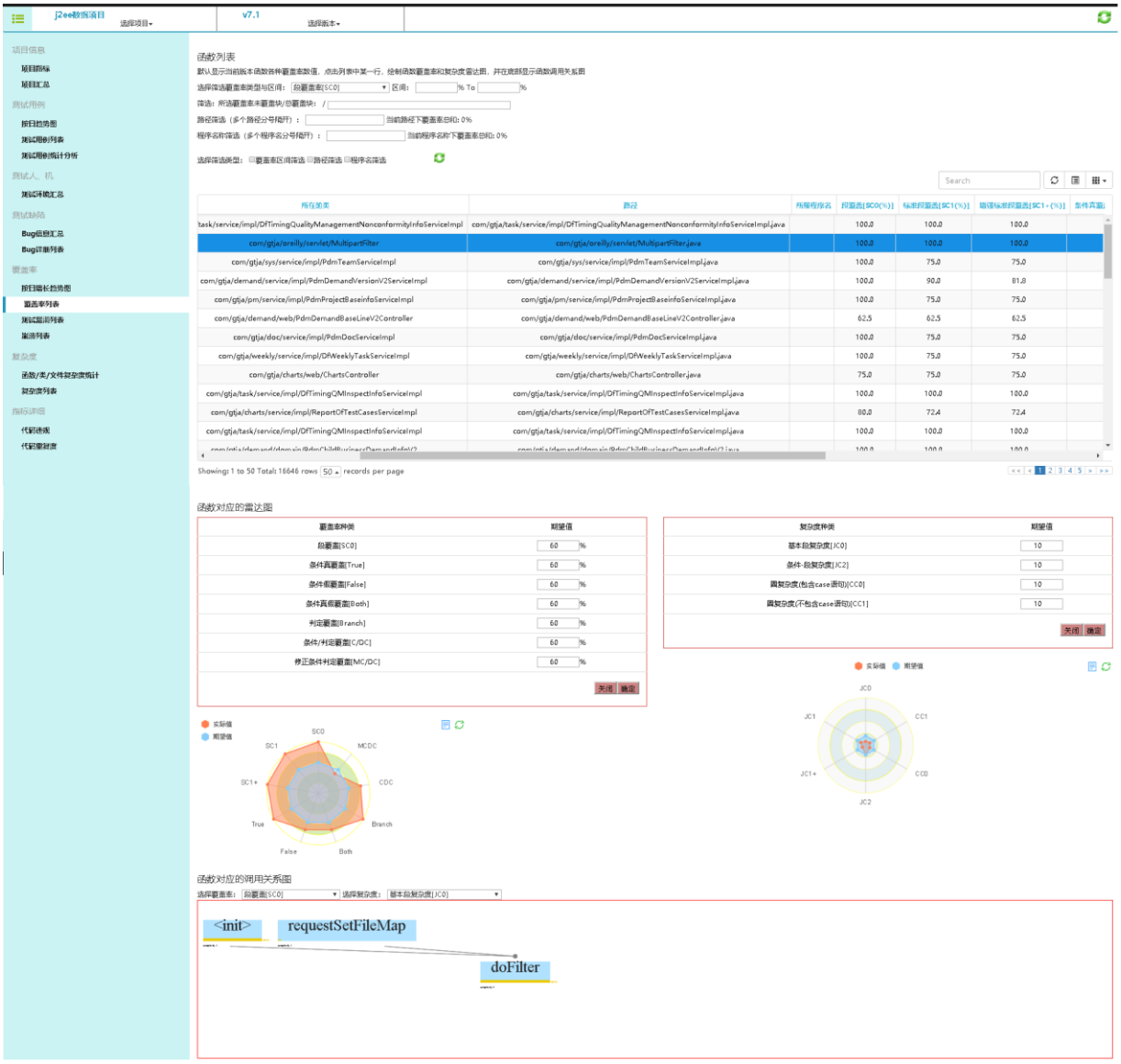


图 5.7 覆盖率列表

5.7.1 覆盖率列表与单函数的覆盖率、复杂度雷达图

通过对单个函数的覆盖率的雷达图设置，用数字化的形式展示核心模块的测试充分度。

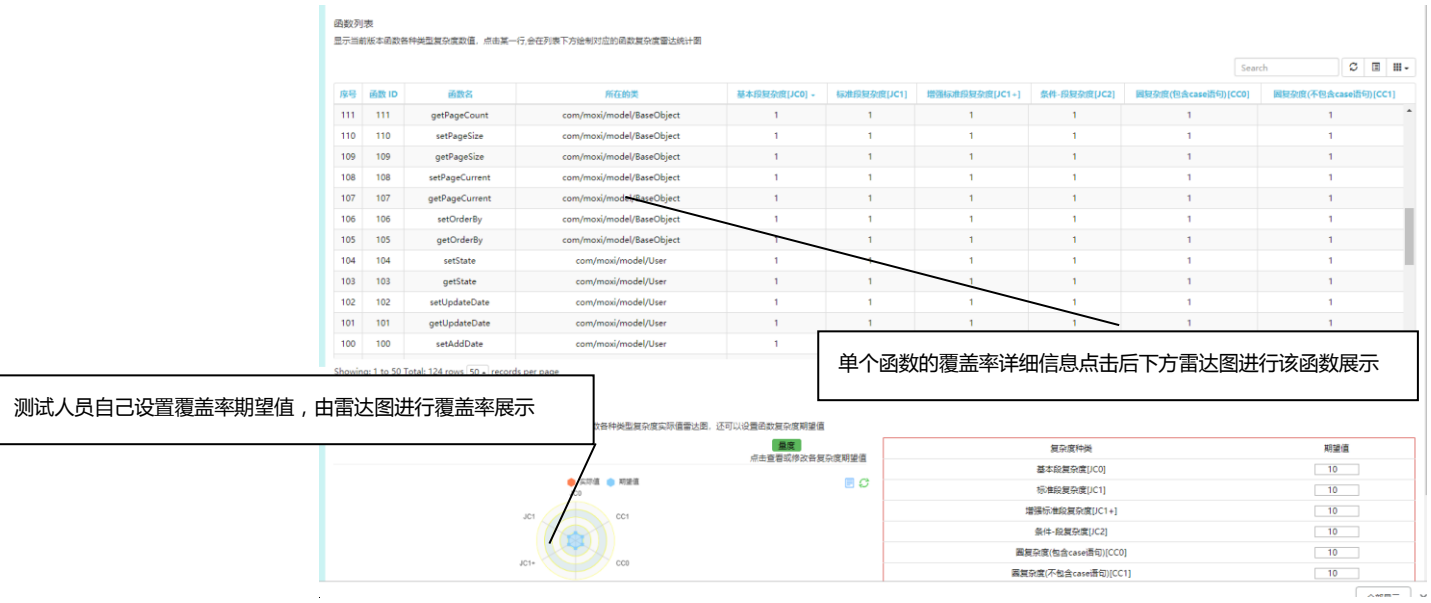


图 5.7.1 盖率列表与单函数的覆盖率、复杂度雷达图

5.7.2 函数对应的调用关系图

选择函数列表的中的函数，对应展示该函数的调用关系，总 3 层。

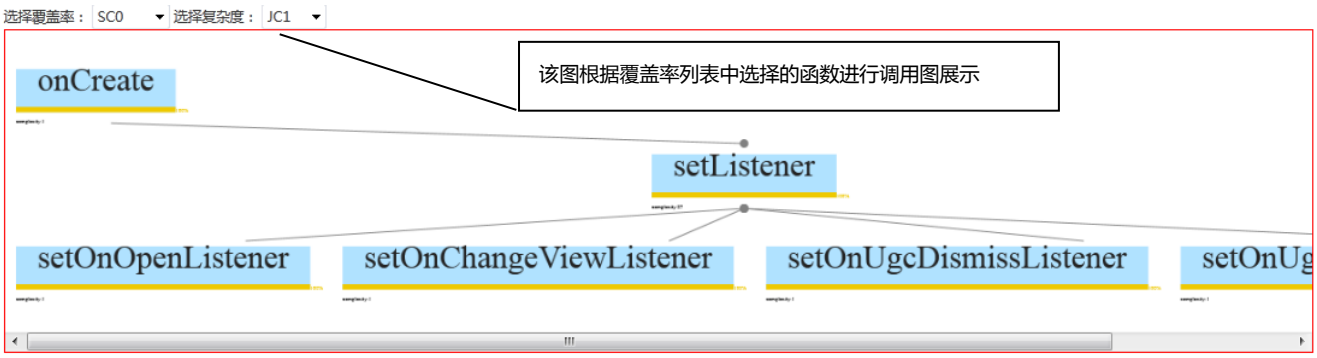


图 5.7.2 选择单函数调用关系图

5.8 复杂度-函数|类|包复杂度统计

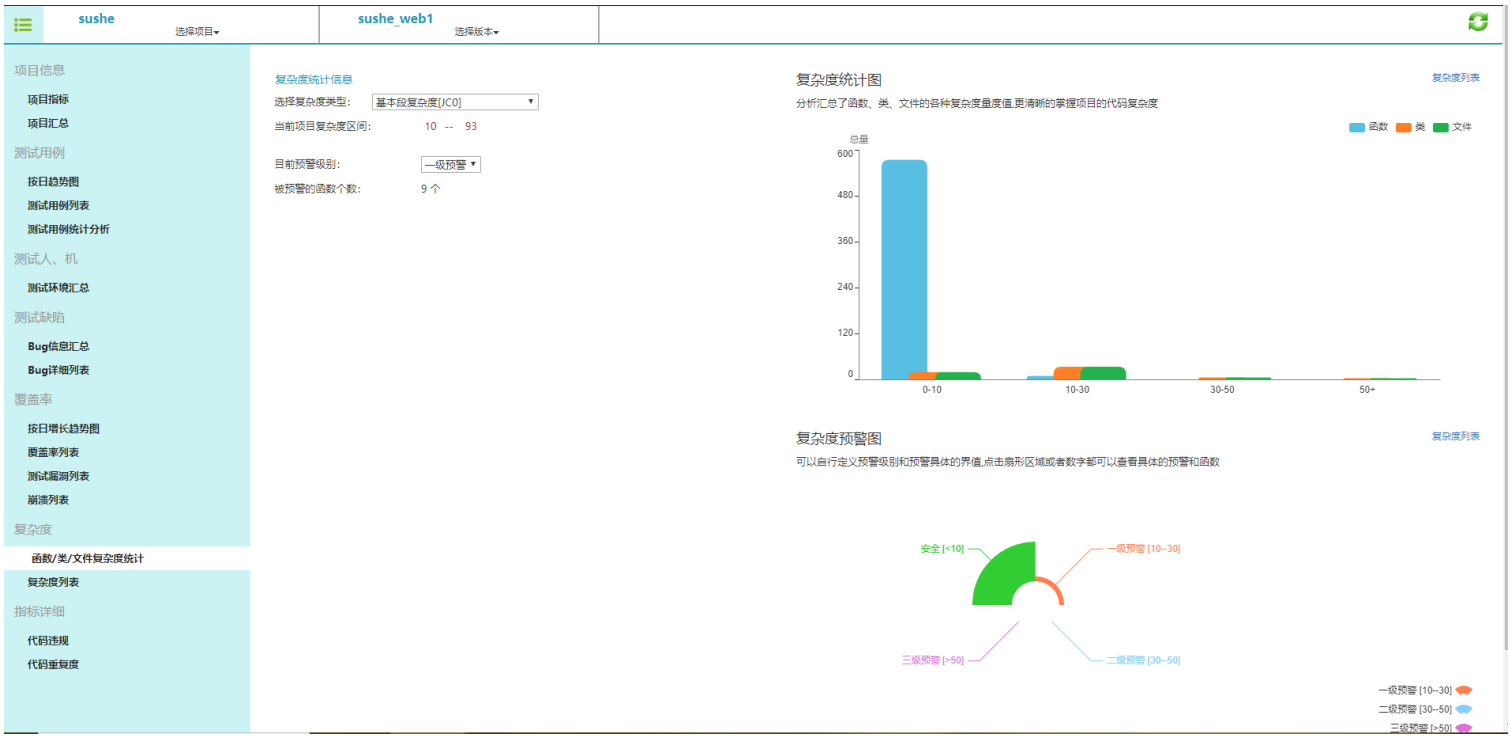


图 5.8 函数/类/文件复杂度统计

5.8.1 复杂度统计信息

为了正对复杂度的风险，星云精准测试给出了预警报告表和复杂度详细列表。

对于安全系数高的客户，测试人员可以要求开发进行重新设计降低风险。

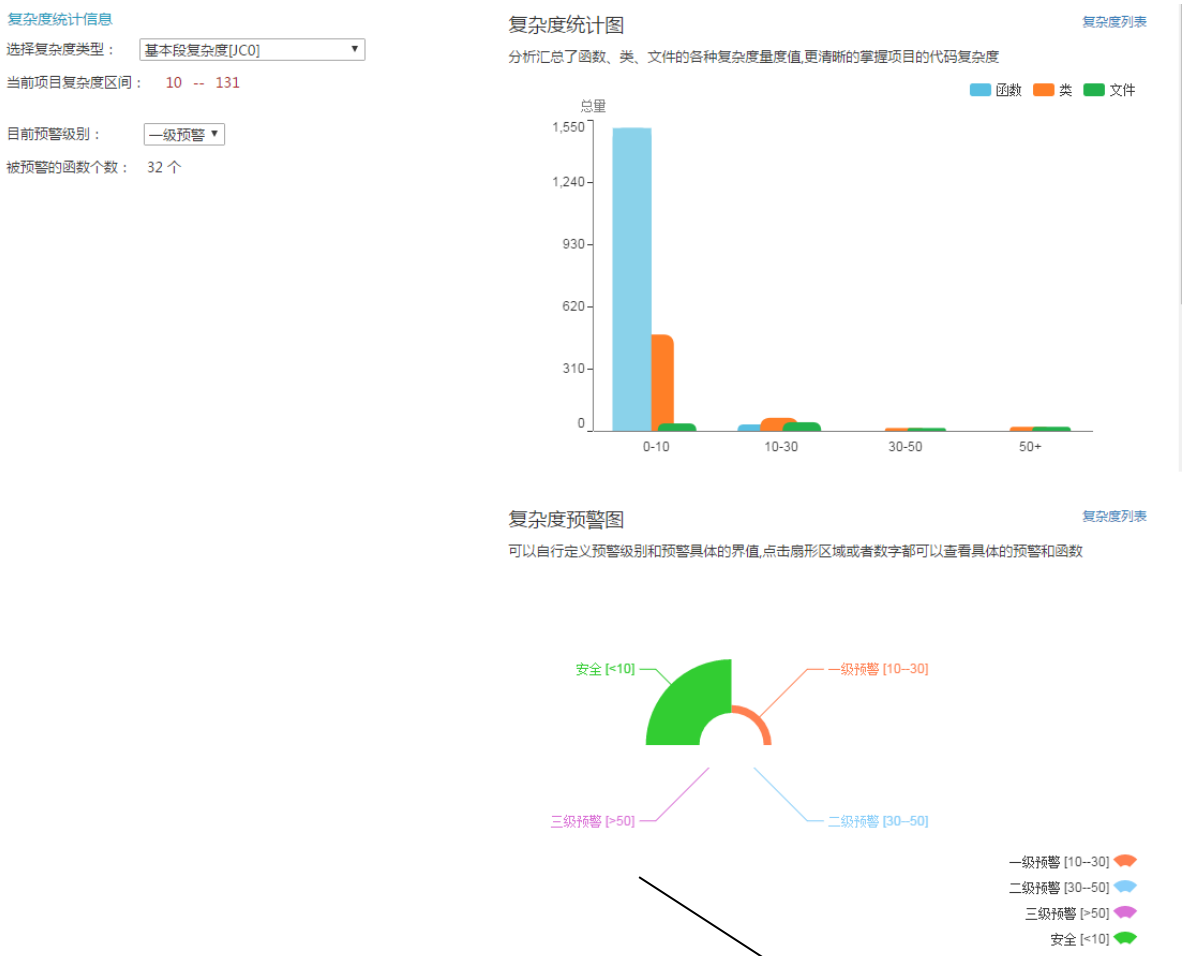


图 5.8.1 复杂度预警

5.8.2 复杂度列表

星云精准测试报表中，展示所有函数的复杂度信息。

点击某一函数，会在列表下方绘制对应的雷达统计图。

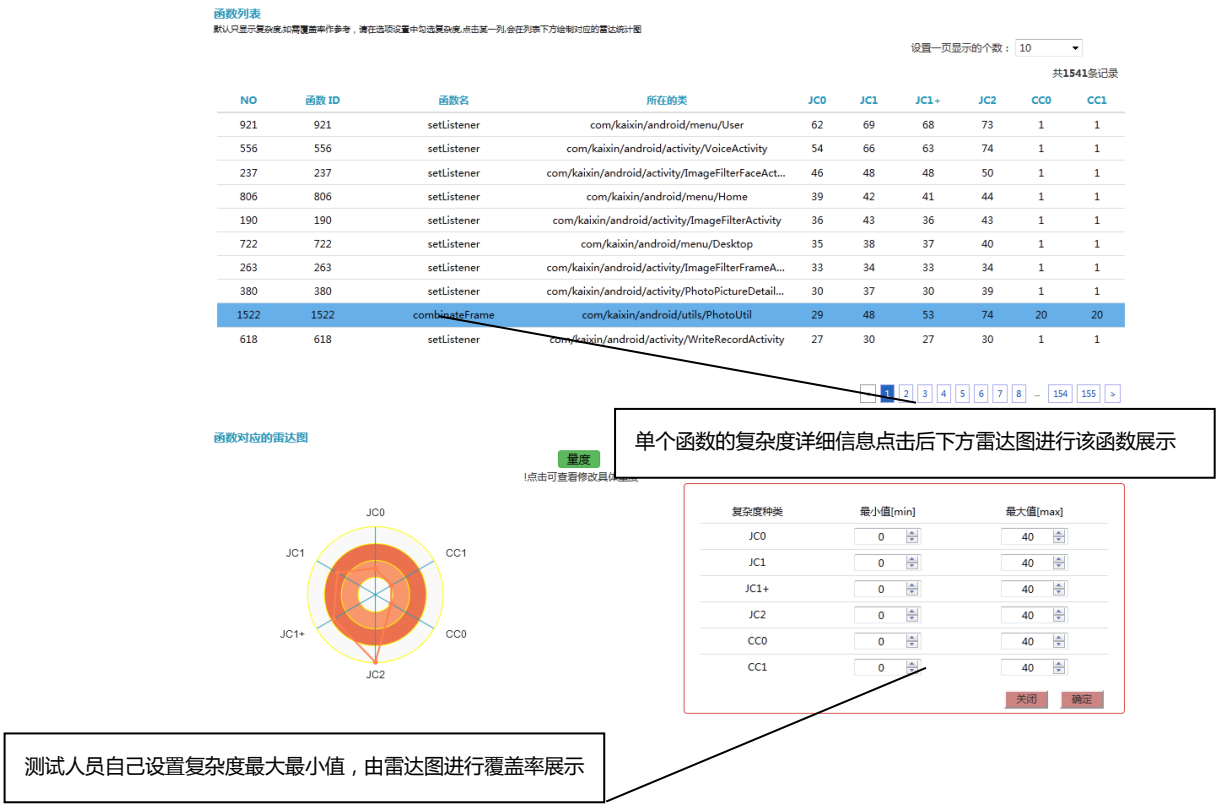

函数列表
默认只显示复杂度,如需覆盖率作参考，请在选项设置中勾选复杂度,点击某一列,会在列表下方绘制对应的雷达统计图
设置一页显示的个数： 10
共1541条记录
NO 函数 ID 函数名 所在的类 JC0 JC1 JC1+ JC2 CC0 CC1
921 921 setListener com/kaixin/android/menu/User 62 69 68 73 1 1
556 556 setListener com/kaixin/android/activity/VoiceActivity 54 66 63 74 1 1
237 237 setListener com/kaixin/android/activity/ImageFilterFaceAct... 46 48 48 50 1 1
806 806 setListener com/kaixin/android/menu/Home 39 42 41 44 1 1
190 190 setListener com/kaixin/android/activity/ImageFilterActivity 36 43 36 43 1 1
722 722 setListener com/kaixin/android/menu/Desktop 35 38 37 40 1 1
263 263 setListener com/kaixin/android/activity/ImageFilterFrameA... 33 34 33 34 1 1
380 380 setListener com/kaixin/android/activity/PhotoPictureDetail... 30 37 30 39 1 1
1522 1522 combinateFrame com/kaixin/android/utils/PhotoUtil 29 48 53 74 20 20
618 618 setListener com/kaixin/android/activity/WriteRecordActivity 27 30 27 30 1 1
1 2 3 4 5 6 7 8 ... 154 155 >
函数对应的雷达图
量度
!点击可查看修改具体
JC0
JC1
CC1
JC1+
CC0
JC2
复杂度种类 最小值[min] 最大值[max]
JC0 0 40
JC1 0 40
JC1+ 0 40
JC2 0 40
CC0 0 40
CC1 0 40
关闭 确定
单个函数的复杂度详细信息点击后下方雷达图进行该函数展示
测试人员自己设置复杂度最大最小值，由雷达图进行覆盖率展示

图 5.8.2 复杂度列表