### 精准测试在智能机器人上的应用

随着工业生产规模的扩大及劳动力成本的提高，机器人作为提高工业生产自动化程度的重要设备，在诸多领域得以推广和应用，但工业机器人核心控制技术一直为国外厂商所垄断，机器人“灵魂“功能如何确保实现？仅凭手工黑盒测试，它的弊端是显而易见的。国内机器人厂商因为没有更合手的精准化测试工具监控、复现其运行结果，很大程度上钳制了研发效率。

目前，笔者观察到国内某个新近崛起精准测试化平台，他们也正在做机器人智能控制验证等人机交互的深度测试研究。可喜的是：在他们最近发布的一个测试实例中，我们可以看到：机器人中的监控、运动规划等多模块的设计要求，用户均可以从自己的客户端，看到测试用例、模块和代码的三向追溯。另外，他们还把用户系统覆盖率未达区域清楚的HIGHLIGHT出来，一目了然，大大加快了回归测试的效率。由于该技术比较前沿，大家如果感兴趣的话，可以随着笔者的测试过程记录，深入了解它的整体测试方法，以利大家工作中碰到问题，可以拿出来参考、验证。（具体平台名称就不展开了，自己可以找一下）。

## 一、案例背景介绍：

该WIFI智能小车由电机、小车车体、控制芯片、WIFI收发模块、电机驱动、舵机、电源、摄像头等部件构成。WIFI智能小车利用笔记本或手机等设备通过无线信号连接到小车WIFI路由器中，通过应用软件显示路由器上摄像头上采集到的视频信号，在通过这些智能的终端设备发送控制指令到路由器，通过路由器将指令传送给单片机进行处理。然后通过单片机控制电机驱动，驱动电机转动、舵机转动，从而实现控制小车的运动及视频采集。

## 二、测试工具与方法

选用某平台工具（简称TT）对WIFI小车机器人的Android手机控制平台进行黑盒手工的测试，达到小车控制与代码逻辑分析。

## 三、静态测试与代码审查

静态测试内容里面要求采取多种的测试方法，例如‘低复杂度的强制要求’一般需要通过满足一定的度量指标来实现，度量指标包括圈复杂度、嵌套深度等等，除此之外静态测试还要求一些其它的测试方法，例如如果要满足更高的安全完整性等级的话，静态测试内容还需要包括运行时错误检测等，一般需要使用可靠性测试性的测试。

静态分析：是对代码的机械性的和程序化的特性进行分析，包括控制流分析、数据流分析、接口分析和表达式分析。

代码审查：是检查代码和设计的一致性、代码执行标准的情况、代码逻辑表达的正确性、代码结构的合理性以及代码的可读性。代码审查应根据所使用的语言和编码规范确定审查所用的检查单，检查单的设计或采用应经过评审并得到委托方的确认。

采用TT代码插装技术生成被测程序的项目框架，并进行静态的代码扫描。

TT的代码插装技术是针对语言对代码进行特定规则的划分，并对其划分的模块加入了探针代码，生成被测项目的函数调用图和控制流程图，以及整个项目的代码量、复杂度与代码注释度等信息，配合动态测试来实现功能与代码的关联。

**函数调用图：**

TT展示一系列关于软件系统的整体信息。如：类或者函数以及类的成员函数的总数目，调用关系或者类的继承关系的深度、层次结构、语句总行数和总体复杂度，整体的测试覆盖率（分累积的结果和最后一次运行的结果，可选择语句、分支和MC/DC测试覆盖率标准）、整体的性能分析结果以及各模块所占的用时比例、以及全局变量和静态变量的分析结果等；同时，又给出了各个模块具体的信息，包括：各模块的源码行数和复杂度、测试覆盖率分析结果、扇入扇出信息，高亮显示一个模块及其所有相关的模块，或者以任何一个模块为根生成局部子树等。



函数调用聚集图：

以类对函数进行分组，通过图表把同一类的函数聚集在一起进行展示。



类继承图：

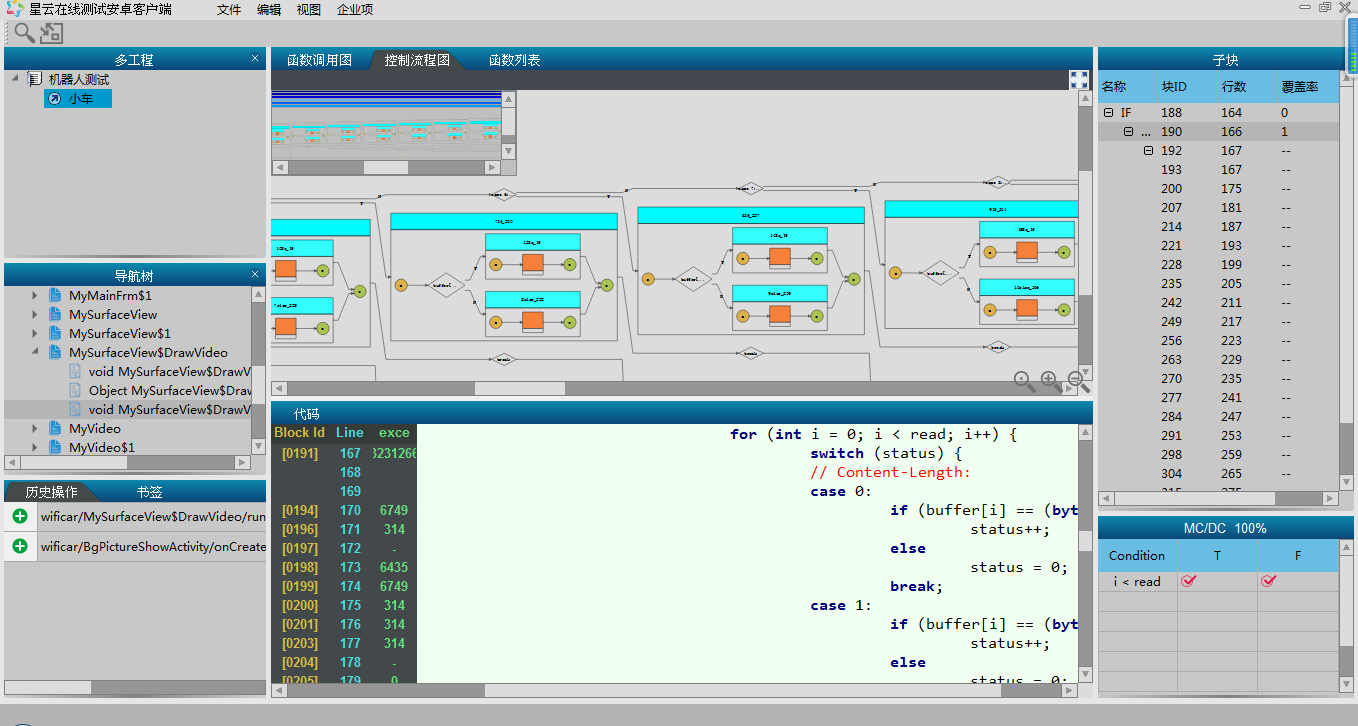
显示的是当前项目所有类的集成和派生关系。



**控制流程图**

通过函数的if-else，while，for，do-while，switch-case等控制语句结构绘制组成的流程关系的展示图；配合下方的源代码展示界面，显示能清晰查看函数内部运行逻辑和结构、条件的真假运行状况、MC/DC的满足率等。

TT可视化的控制流程图，对主要的控制逻辑语句对应有清晰的图元显示，并支持嵌套显示以及串联显示。当点击控制流图的每个图元可以清晰看到对应的代码段以及代码段的执行次数、覆盖率情况。当条件语句成为选中热点后，可以清楚的展示条件语句的各个子条件的各种组合执行的真假情况。



**代码复杂度分析和安全检查**

复杂软件不稳定，也经不起不可预测的行为。所以，我们努力使软件的复杂度变小。如果有条件采用某种自动化工具，可以通过工具对软件设计或/和代码进行控制，用图形化的方法反映出软件结构中的控制流和数据流，通过连结数/调用数、节点数、嵌套深度等这样一些结构关系的检查，获得复杂度的度量，将会获得很好的效果。

TT在对代码的分析过程中，TT可以直接给出代码复杂度的计算结果，通常复杂度越高的软件模块更容易引入缺陷，也更加容易引入安全性问题，高度复杂的数据结构难以彻底测试，可以采用TT等复杂性评估技术来标示出需要进一步改进的区域，以便提升软件的安全性。



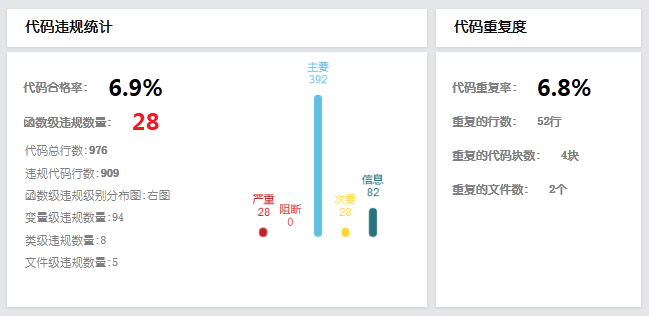
**TT可维护性检测**

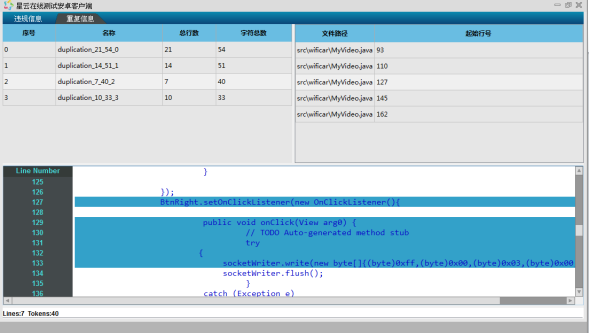
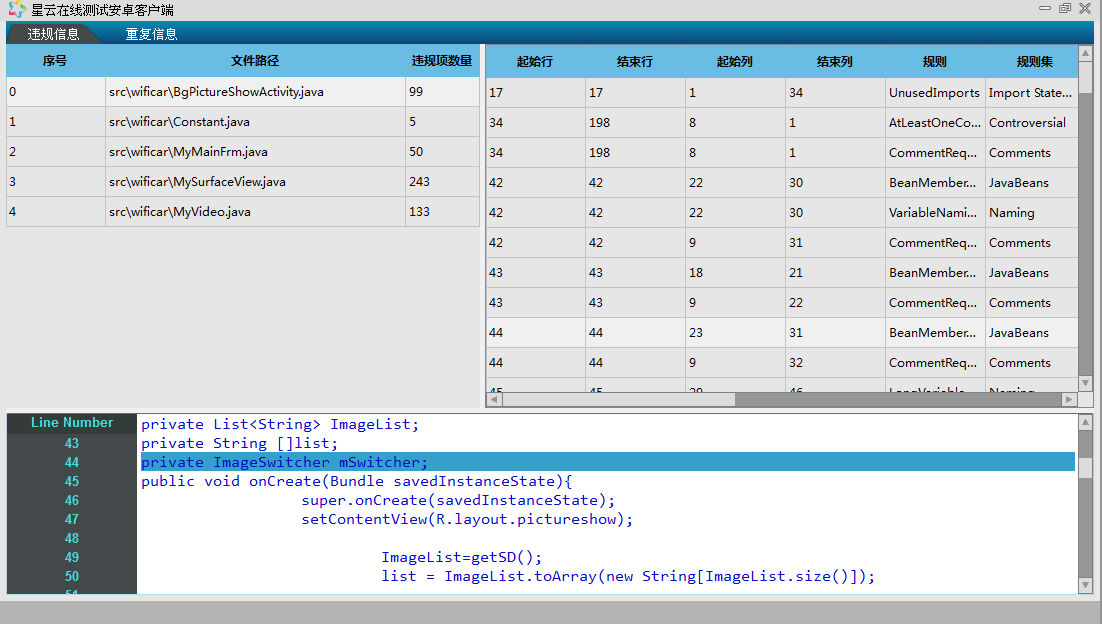
TT通过代码的行数、复杂度、注释行数等信息的结合计算，对代码的可维护性进行了打分，良好的可维护性是对项目后期维护的一种保障。



**TT违规与重复代码检测**

TT的静态扫描采用PMD代码分析技术，对原代码中的静态违规、重复等代码进行检查并可视化的展示。





## 四、动态测试

以前的动态测试大部分是通过人工的方式实现，但是由于动态测试巨大的工作量，以及测试用例难以重用，测试覆盖率无法计算的困难，因此在测试解决方案中还是需要选择合适的动态测试工具来执行软件单元测试。又因汽车行、机器人行业对需求和测试有可追溯性的要求，所以最好测试工具可以和相关需求管理工具集成，例如能够从需求管理工具中将需求导入，再将测试用例和需求链接起来，并且实现数据的双向同步。

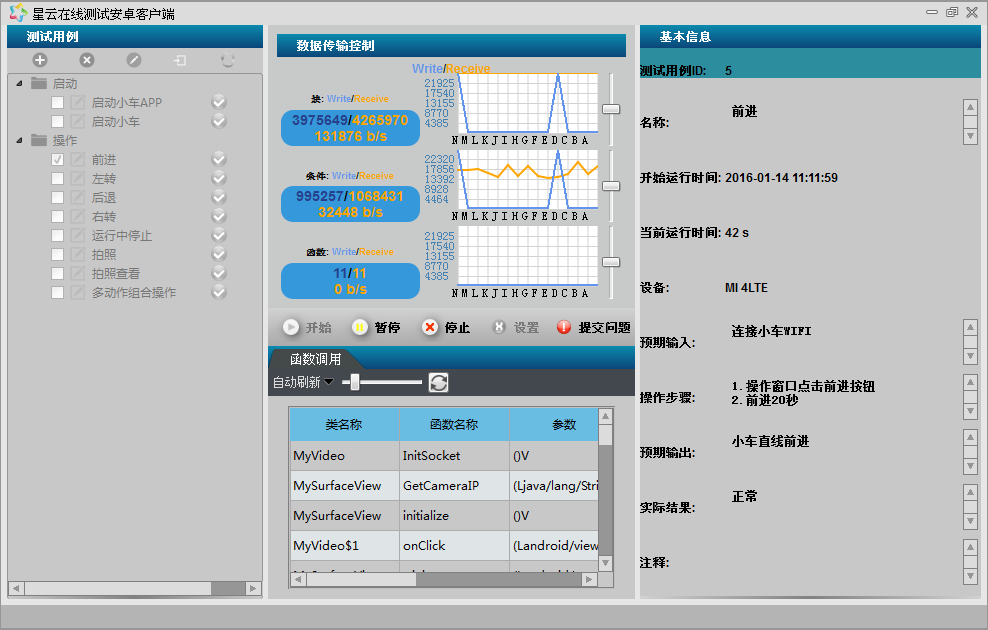
### 1．示波器技术

TT首先在软件测试中使用了示波器技术，在实际测试的过程中，测试员可以看到类似于心电图的数据获取模式，让软件测试过程因此变得可视、精准。

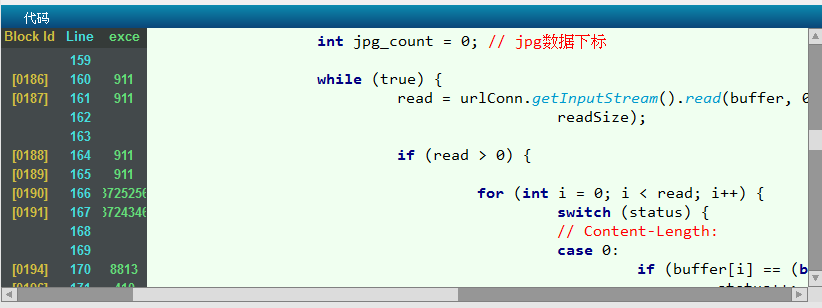
TT示波器采用测试离线测试技术使得源代码分离户外测试变的可能，在实际测试中，我们把小车机器人放在户外的草坪中，通过手机与示波器进行控制进行测试，测试完毕后把测试数据拿回实验室与代码进行对接并分析，从而使得在保护源代码的情况下对户外代码测试变得可能。

TT示波器的Console模式优点：

* 可视化观察程序运行指标，辅助测试人员对被测试程序进行分析，例如等价类的辅助划分、探索性测试等。
* 示波器可以实现WildTest（场测），无需与PC的连线通过wifi就可以实现高速获取被测数据。



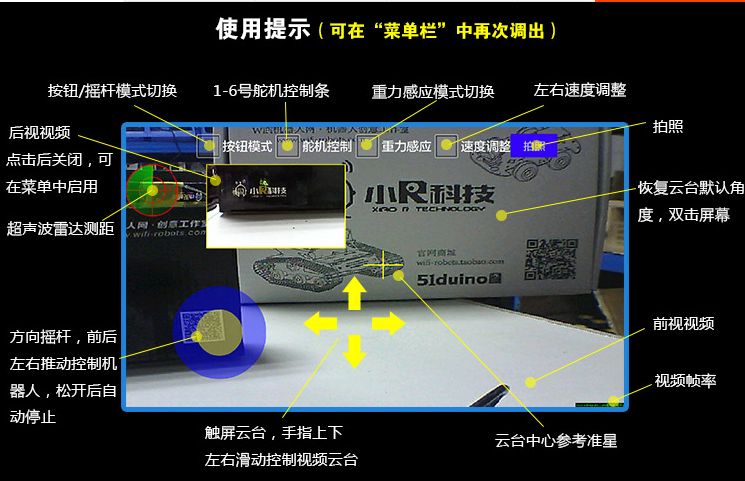
在实际运行中，我们虽然未对小车机器人进行操作，但示波器还是从小车机器人那里接收到了大量的数据信息，我们通过函数的追溯查看其代码的运行次数，发现在run函数中存在的while死循环的代码，这就导致了产生大量数据的原因。

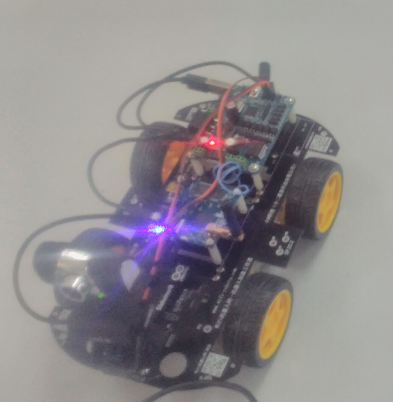


### 2．测试用例与代码追溯

功能测试：是对软件设计文档中的功能需求逐项进行的测试，以验证其功能是否满足要求

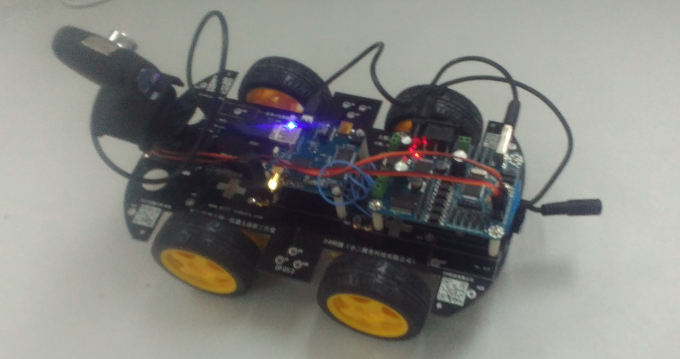
在实际测试中，我们首先对小车机器人的方向进行功能上的测试，使用前进、后退、左转、后转、停止等按钮，逐次对小车进行移动。





小车方位移动测试

其次我们在对小车机器人进行空转测试，逐次把小车左侧速度与右侧速度一直将为0，运行小车移动，发现小车在马达发动时，会一直进行长响音。



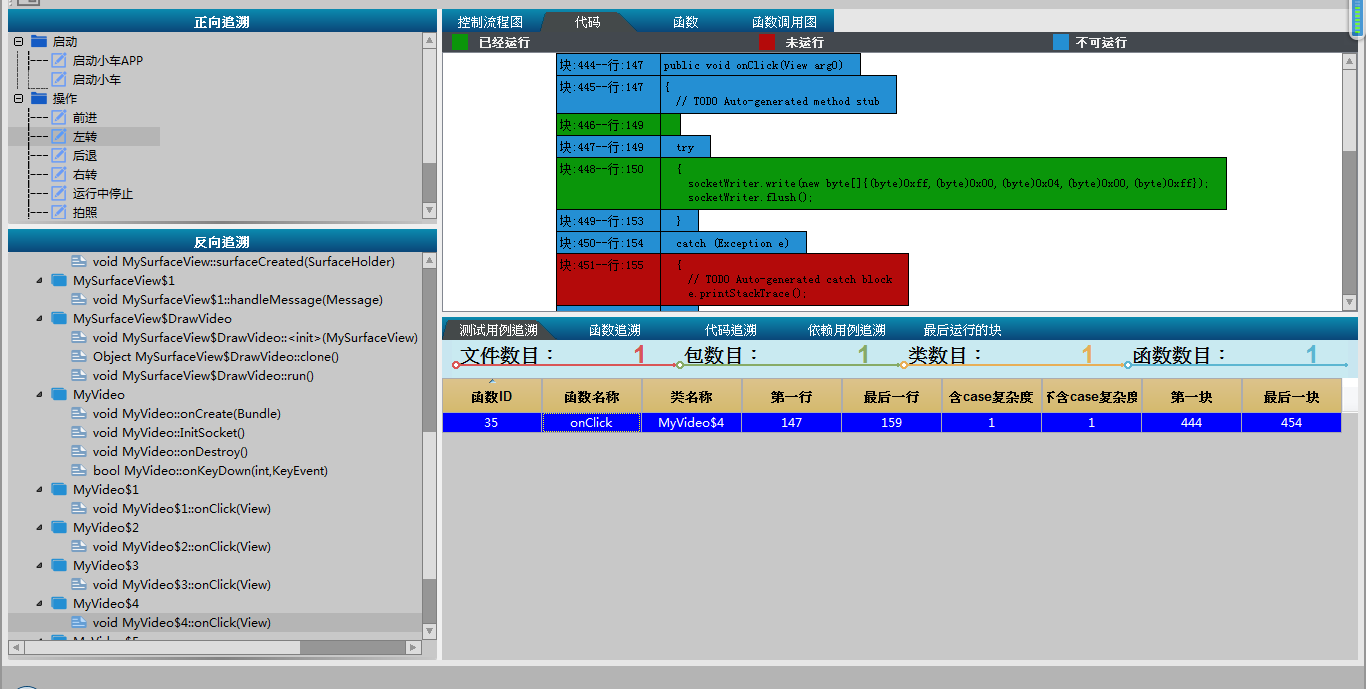
空转测试

通过这些场景的模拟测试，测试数据会通过示波器进行接收并关联相应的测试用例。

TT通过技术手段自动建立的测试制品（测试用例）与开发制品（代码逻辑）之间的双向追溯 。

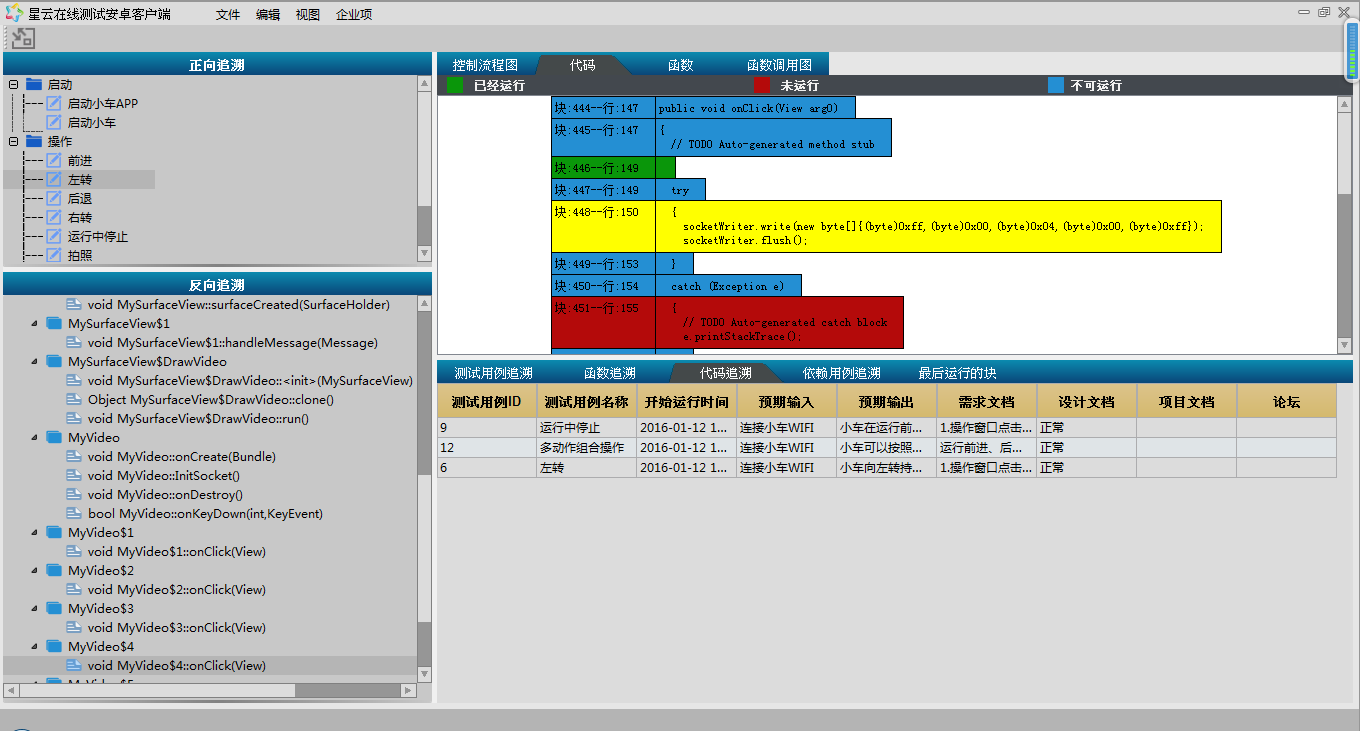
**正向追溯优点：**

* 迅速定位缺陷对应的代码执行逻辑，帮助开发快速修复缺陷，可追踪难复现缺陷。
* 精确，详尽的记录测试用例运行的情况，为精准软件测试提供大量原生分析性数据。
* 可以进行事后的缺陷分析、追踪，辅助开发进行功能实现确认。



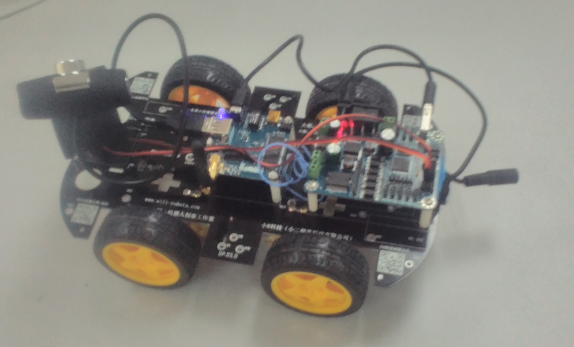
**反向追溯优点：**

* 辅助进行一致性修改
* 辅助进行回归测试用例自动选取

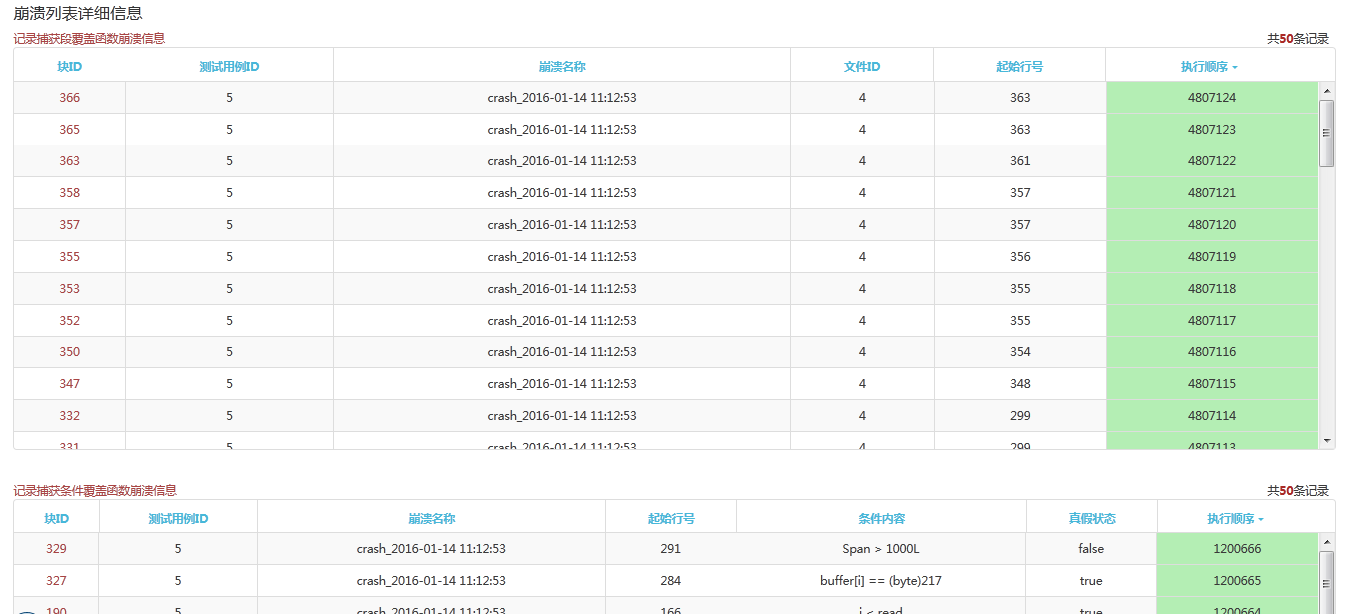


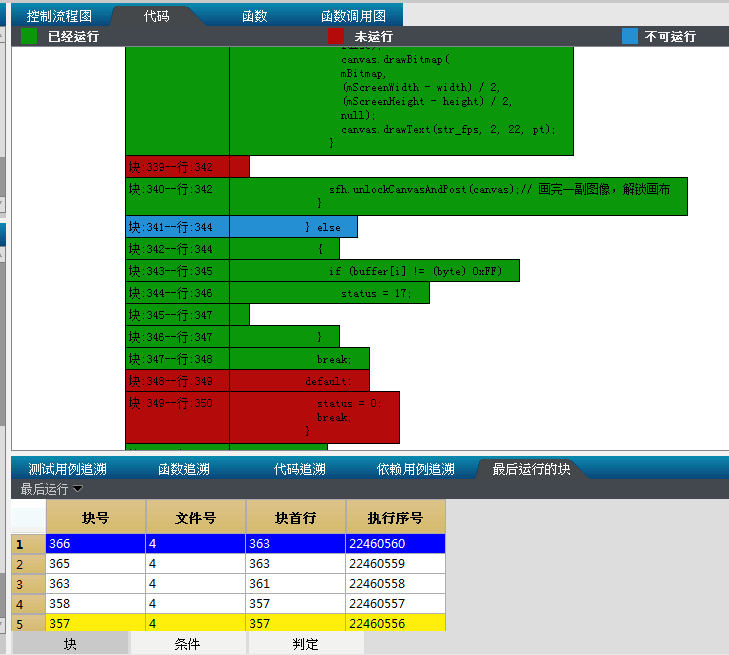
### 3．代码级BUG追溯

我们在进一步对小车进行移动过程中摄像头进行测试，在测试中，我们对摄像头进行了各种角度的转动并进行了拍照，完成后进行对照片的查看，但是在测试过程中，当我们读取拍摄照片时候发现无法读取现象，通过TT的崩溃列表查看到有崩溃信息，对其进行查看，直接定位到代码最后的运行过程。



摄像头转动拍照测试



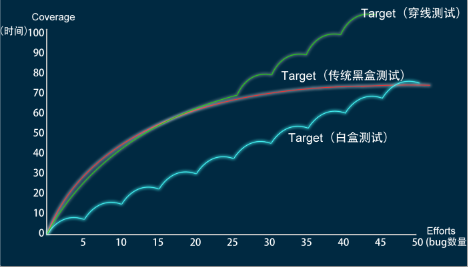


### 4．覆盖率与可视化分析

**覆盖率技术**

测试覆盖率是测试界公认的最佳的是测试结项的可用指标。在黑盒测试中，测试人员需要充分了解需求后逐渐细化到小功能，进行测试用例的设计，然后通过测试过程中出现的问题以及和客户的交流再加以完善，这样的过程过度依赖个人经验和能力等因素，并且也不一定能达到完美的效果。在白盒测试中，测试流程需要对每条代码、每个语句、每个判断进行运行分析，测试周期冗长，大多企业无法承受。

TT把大量复杂的运算放在后台，用户界面简单易用。首先，测试人员用传统黑盒测试方法把基本的功能都点测一轮，排除掉大约70%左右的初级BUG；随后，利用TT云平台进行数据分析与排查。第三步，根据分析数据报表，快速、有针对性地补充测试用例，达成覆盖率管理的实际要求，做到有理有据、心中有数。



TTC 覆盖率形式多样，最高支持航天航空标准MC/DC的100%覆盖率要求，并且提供了全新的覆盖率每日增长趋势图。项目管理者可以通过该报表清晰的观察整个测试进度情况，TTC平台的覆盖率趋势图对于团队的质量控制具有很好的指导意义，它能够让高级管理人员对测试进度进行预判，也能够对测试效率进行有效的识别，例如通过对覆盖率增长曲线的拟合，可判断按照目前进度能够在上线日期到达前能够一个合理的测试水准；通过覆盖率增长是否走平，可判断是否达到黑盒测试的极限区，主动地进行测试策略的调整。





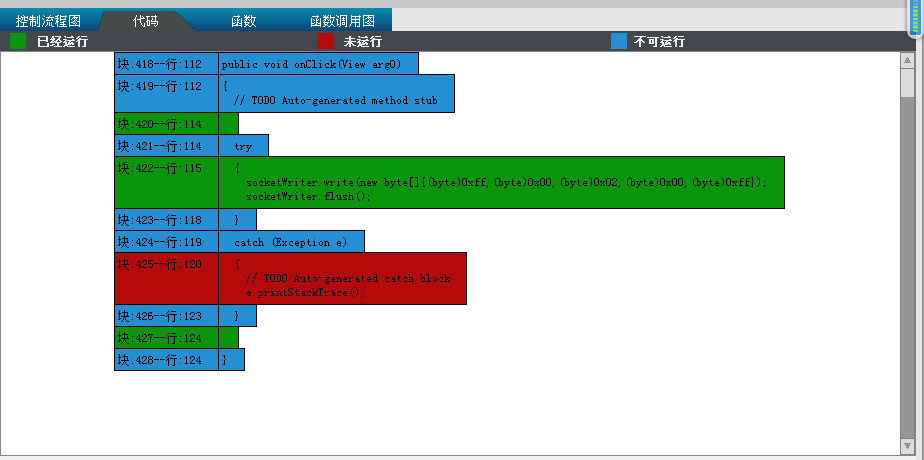
同时通过每日的覆盖率增长情况、函数|类|文件复杂度和覆盖率关系图了解那些高复杂度的函数的覆盖率指标，这些测试数据的展示分析能够让项目管理人员更好的把控整个测试的节奏，进行测试规划。

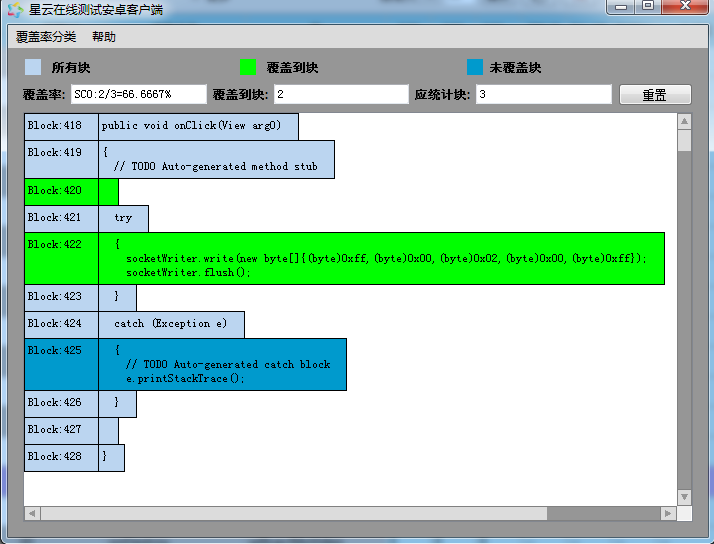


**覆盖率可视化技术：**

TT支持对各种覆盖的计算的可视化，对每种覆盖率的计算结果给出可视化的直观结果展示。覆盖率可视化功能更直观、更清晰的向用户展示哪些块被覆盖，哪些块没被覆盖到，让用户更有效的进行测试补充、达到预期覆盖。

如：我们查看测试用例中的后退用例，查看到其函数覆盖率为66.6%，通过覆盖率可视化进行查看分析得出，其中未跑的代码为catch异常捕获代码，在实际测试过程中因为没有运行出测试异常，以至于该代码无覆盖。





### 5．测试漏洞

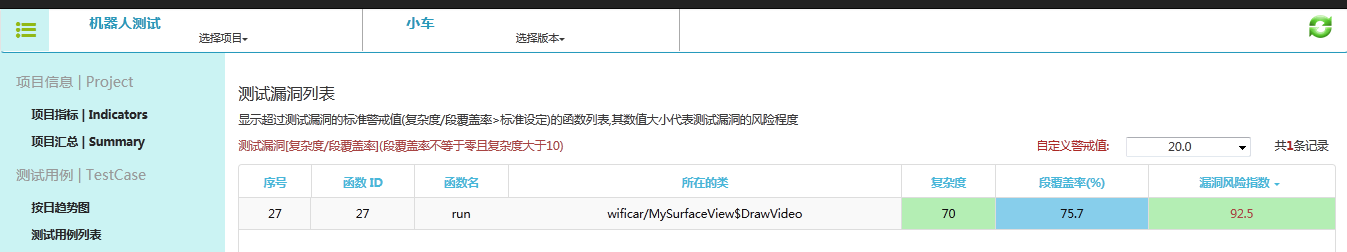
TTC通过静态、动态指标的综合分析，在大量的程序中，通过计算直接筛选潜在的高危的测试漏洞。

针对这些高危测试漏洞分析的优势：

大大提高了废弃代码的排查率：针对这些无用的代码进行相应的注释与删除提高软件的后期维护性。

提高了测试和开发的互动性：测试人员可以通过测试漏洞分析，直接对高危的核心模块功能与开发合作完成测试。

核心模块的测试覆盖率提升：通过漏洞分析检查出核心模块的测试遗漏点，通过可视化的图形查看，进行测试用例补全，消除测试遗漏点与盲点。



如实际测试过程中函数ID27，复杂度为70，在测试过后，段的覆盖率为75.7，通过覆盖率可视化查看覆盖率的覆盖情况，并通过分析进行测试用例的补充。

## 五、测试管理

TT提供了实时的测试报表，使用者只要通过浏览器登陆报表系统，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。你可以很直观地和大家一起推动任务进展、有针对性的进行能力提升规划。

测试团队、开发团队，项目委托方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。

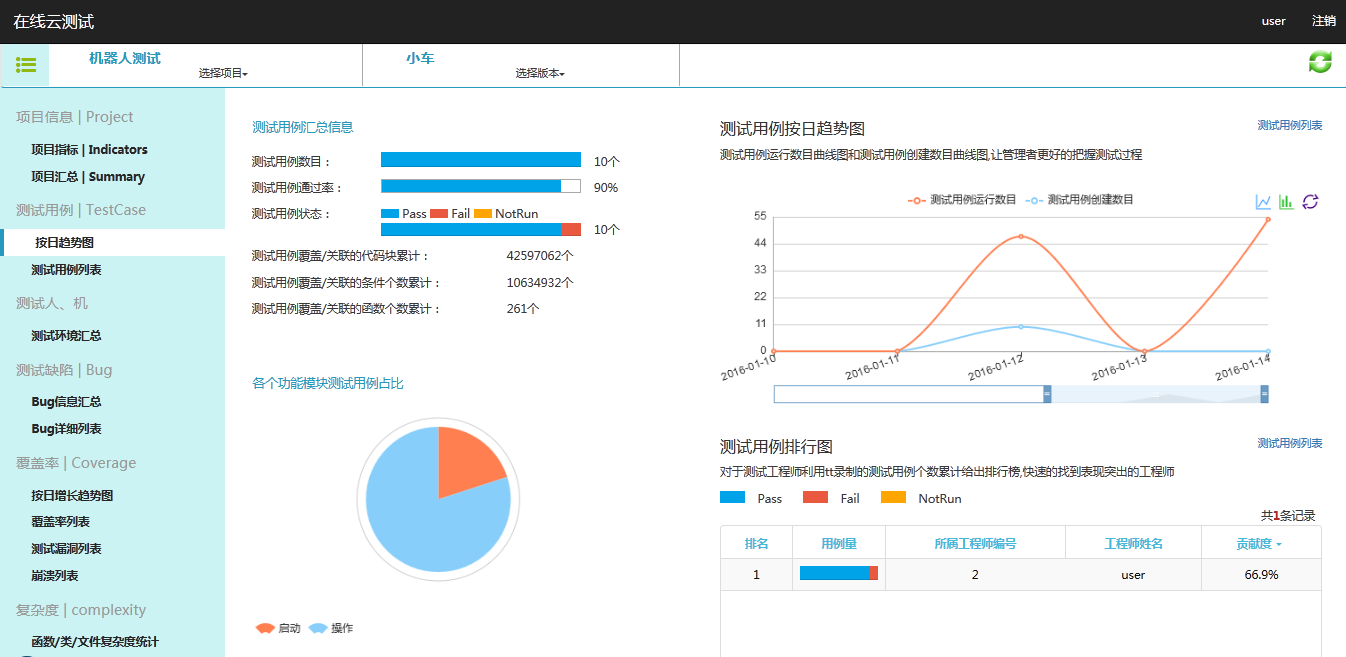


### 1．测试团队人员分析

在以往的测试中，评价一个功能测试团队和测试人员，主要看他的寻找BUG的能力，但是在实际中，因测试项目的质量以及测试人员对业务的理解和测试人员的工作年限，不光只能靠BUG来进行评定。

TT报表通过对测试人员运行的测试用例、测试用例的覆盖率、测试用例BUG等关联，直接反映测试人员的测试状况，避免测试与开发关系沟通问题，以及有针对性的对测试人员进行指导。

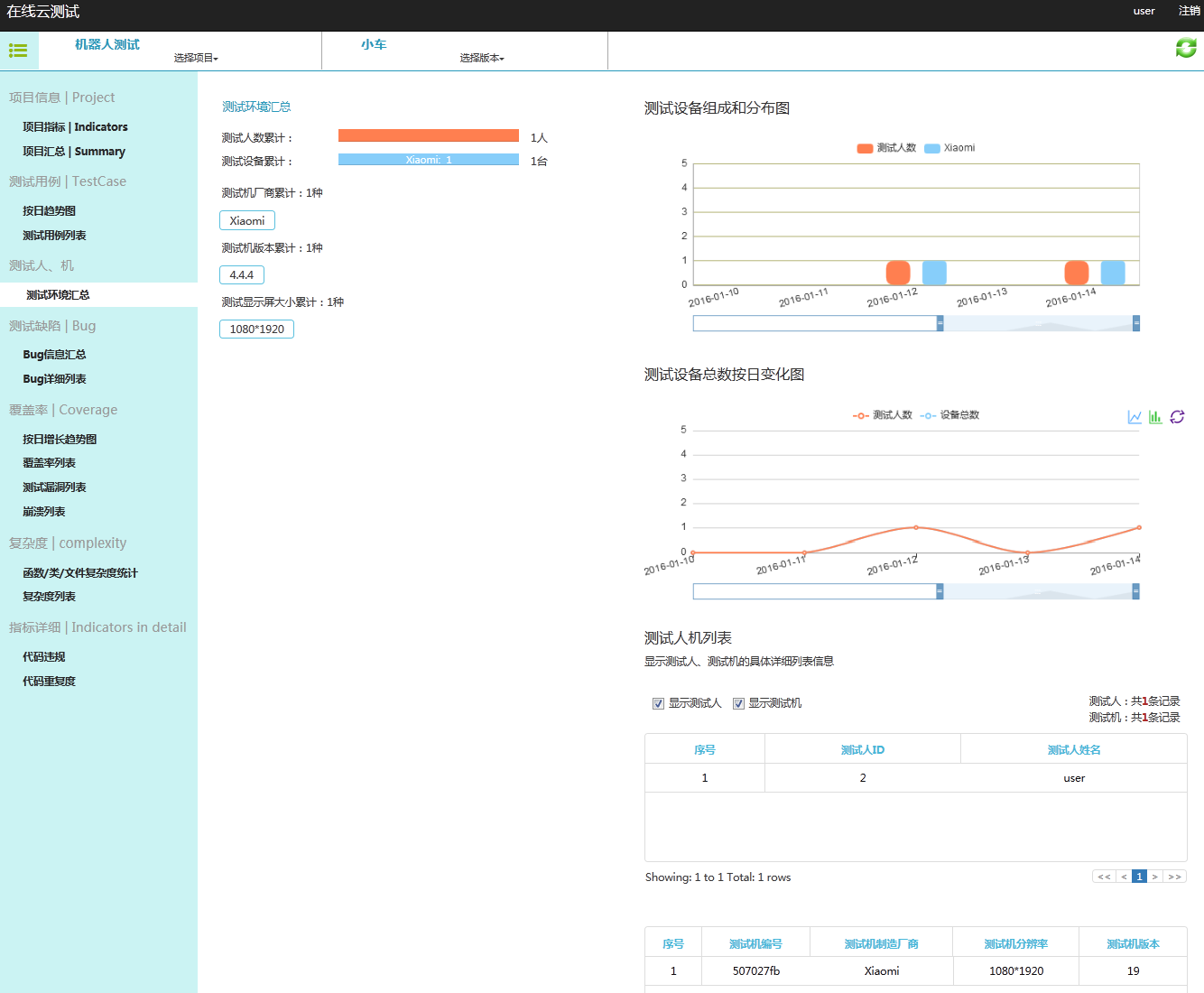
如：某个测试人员设计的测试用例很全面，运行遍历后，TT测试的报表覆盖率也很高，但是却始终发现不了BUG，这时我们可以判定为2种情况：第一种，该测试人员测试的项目确实没有很大的问题，第二种，测试人员对业务的理解有可能存在偏差，虽然运行了大部分的功能，但BUG也包括友好度、逻辑输出等，这些都是业务理解层面的，针对这种情况，可以对该测试人员进行业务上的指导。



### 2．测试设备分析

在android很多测试场景中，测试人员在测试过程中没有发现任何问题，但是客户在使用过程中缺平凡出错，这些问题有不少是因为系统或兼容性导致。

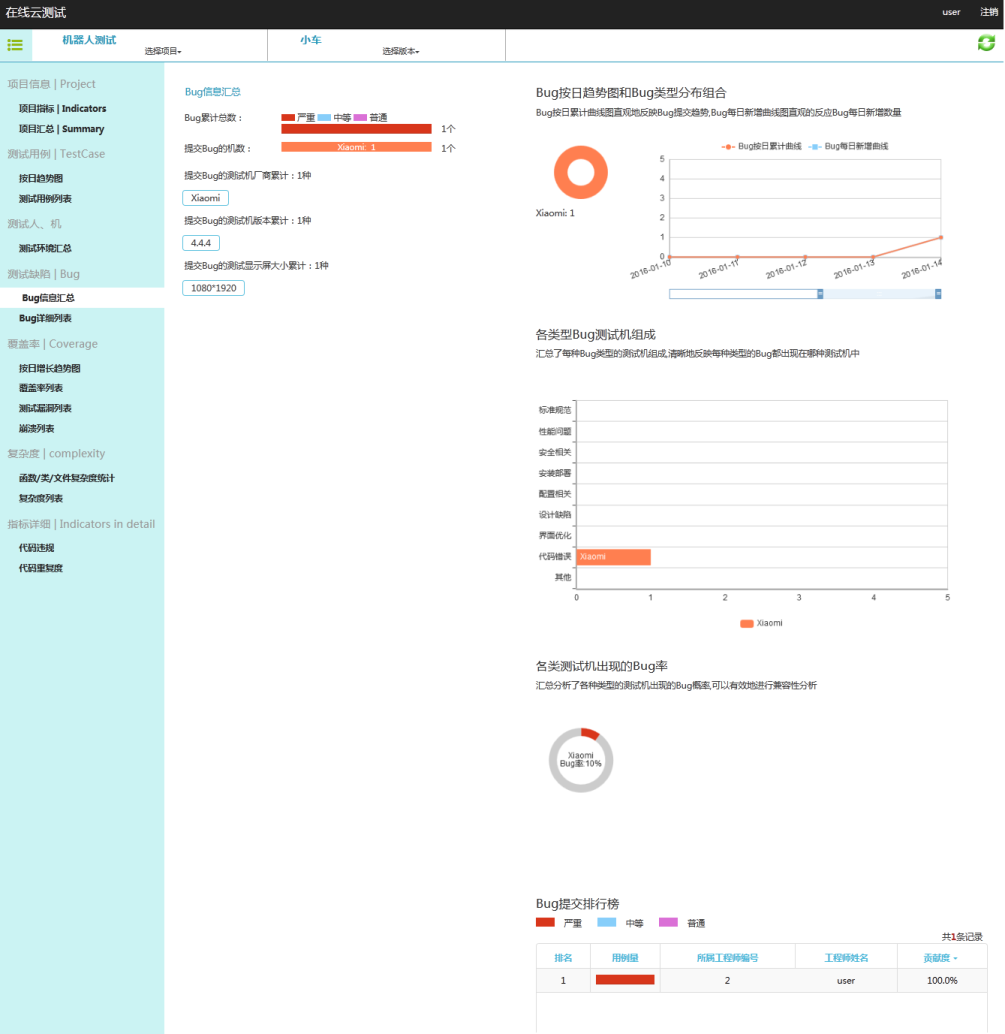
TT报表会在测试人员在测试过程中原生态的记录该测试过程中使用的测试设备与系统信息，并和测试用例、BUG等进行关联，可以有效地管理整个测试设备的使用以及对应情况。



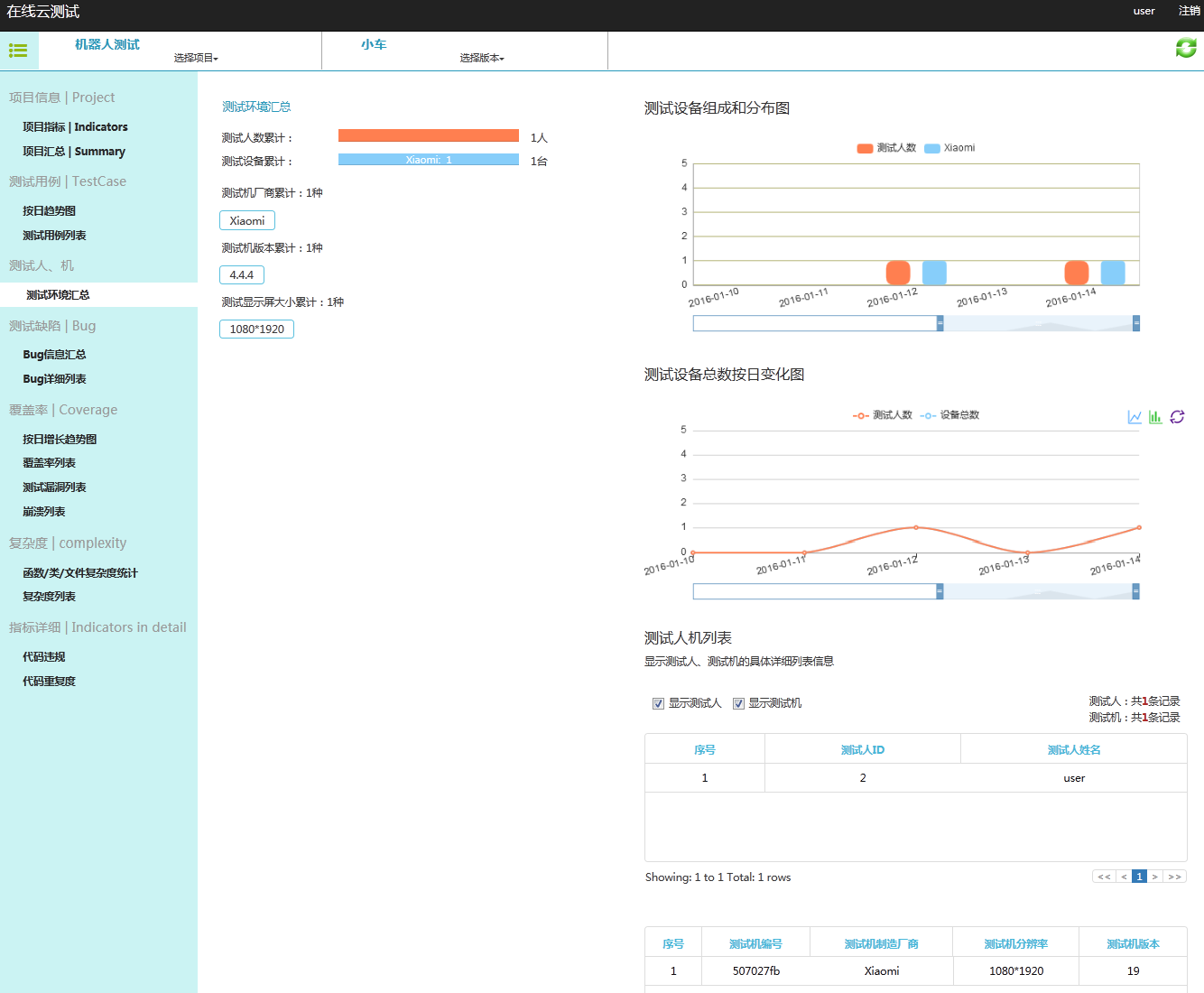
### 3．BUG管理

在TT中，BUG不像传统缺陷管理工具一样是单独的一个功能，TT通过把BUG和测试用例以及代码、设备等关联，这使得BUG分析更加精准。

如：BUG和测试用例关联，TT通过测试用例的历史版本和每个版本中关联，可以查看到版本迭代过程中，该测试用例的BUG修复情况，这种关联有效地避免了版本迭代中重复BUG的提交以及重复BUG的出现。







## 六、回归测试用例选取算法

TT打破传统白盒测试必须对上一个版本进行完全测试后才能回归的弊端，采用所有版本测试用例分析选取技术，实现自动化的版本对比，分析出修改、增加、删除的代码以及这些代码相关联的测试用例，快速做测试用例回归测试、快速补充测试用例、删除无效测试用例。

TT回归的优势在于：

* 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。
* 降低了传统人工回归分析产生的测试盲点，函数代码的修改可能导致调用该函数的代码产生错误，所以需要测试其它代码。如项目工程比较大，人工分析不可靠也耗精力。
* 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。
* 适应快速的版本迭代周期，适应庞大的工程项目。