基于 TestCenter 的测试管理方案

概述

TestCenter 是上海泽众软件科技有限公司开发,具有自主知识产权的产品。在测试的准备阶段,需要构建一个一般的测试流程。该测试流程即适用于手工测试,也适用于自动化测试,见下面测试流程图。

首先在需求模块中创建需求节点。选中需求节点,点击添加子节点。自动弹出添加子节点对话框。在对话框中依次填写需求信息,然后点击"确认"按钮,即可方便地创建子需求。

TestCenter 使用需求树描述测试需求,用以体现测试需求之间的包含和从属关系。相应地,也可以在 TestCenter 中实现对测试案例和缺陷的管理。并且在新版本的 TestCenter(version 5.0)中增加了评审模块,可以对测试需求和测试案例进行评审。具体包含的内容可以通过下一节了解。

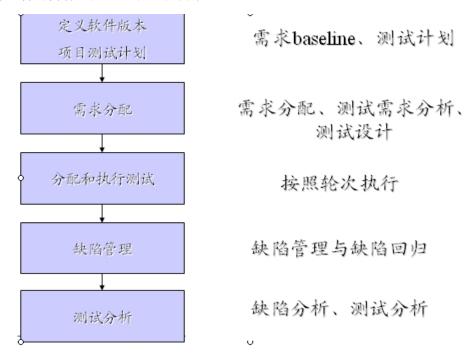


图 1 TestCenter 中的测试流程

需求导入与测试需求分析

在需求导入和测试需求分析阶段,需要做的工作如下:

1) 需求导入

TestCenter 管理的是测试需求,第一步是把 SRS 导入到测试需求模块,再进行测试需求分析工作。TestCenter 提供了插件,并且定义了需求模板,用户使用这些模板,就可以把 word、excel 格式的文件轻松的导入到系统中。

需求导入的步骤:

第一, 提供需求文档:

第二, 提供 TestCenter 需求导入文档模板;

第三, 应用需求导入文档模板,修改需求;

第四, 导入需求;

第五, Review 导入的需求,以保证正确性。

TestCenter 产品支持通过自定义的 excel 格式、word 格式来导入需求。 从需求管理系统导入测试需求,并且建立需求树,见下图:



图 2 需求覆盖率视图 (需求树)

上图是 TestCenter 测试需求管理的覆盖率视图。这个视图分为三个 tab 页来显示需求信息:详细信息输入视图(测试需求的详细描述和附件)、需求内容视图(显示需求的正文)、测试用例场景视图(显示与测试需求关联的测试场景)、需求链接视图(显示关联的测试案例)。

填写需求属性:

- ▶ 需求名称;
- ▶ 需求的描述:
- 需求的详细信息;
- ▶ 需求的附件:
- ▶ 需求的测试主题;
- ▶ 需求的种类;

- ▶ 需求的优先级;
- ► 需求的 RBT 值。

维护需求:

- ▶ 增加需求节点,包括子需求和兄弟需求节点;
- ▶ 删除当前选中的需求节点;
- ▶ 修改需求:
- ▶ 需求的不同视图察看;
- ▶ 搜索需求:
- ▶ 需求关联测试案例。
- 2) 测试需求分析

根据需求的理解,进行需求分析。

第一, 根据测试策略, 给需求增加测试项。测试项, 就是指这个需求如何被验证 的项目。

第二, 把需求模型根据测试项来转换为测试模型。

3) 评审需求

功能测试是基于需求的测试,如果没有需求,或者需求不符合进入功能测试的标准,测试工程师就无法根据需求来设计测试案例。

因此,进入功能测试阶段,需要进入一个准则,并且根据此准则进行需求评审。 需求评审的内容:

- ▶ 评审需求整体是否完善
- ▶ 评审需求的各个功能点是否按照要求进行描述
- ▶ 评审需求点的角色
- ▶ 评审需求点的前置条件是否清晰、明确
- 评审需求点的处理流程是否清楚
- ▶ 评审需求点的异常处理流程是否清楚
- ▶ 评审需求点的验证方法

从测试的角度看,需求是测试的重要输入,这个输入应该具备以下几个部分:

- ▶ 业务操作过程描述;
- ▶ 本功能的数据和约束规则:
- ▶ 本功能涉及的活动图;

几个部分对应表:

| 需求功能 | 对应测试对象 | 说明 |
|-------------|----------|----|
| 业务操作过程描述 | 测试脚本 | |
| 本功能涉及的活动图 | 场景(业务流程) | |
| 本功能的数据和约束规则 | 测试数据 | |

表 1 模型对应表

4) 测试案例关联

需求节点能够关联一个或多个测试案例。

关联的测试案例用来计算需求覆盖率: 当此节点关联的所有测试案例测试通过,我们就认为此需求被覆盖。否则认为没有覆盖。在执行一次测试计划之后,我们可以根据以上的规则来计算需求覆盖率。

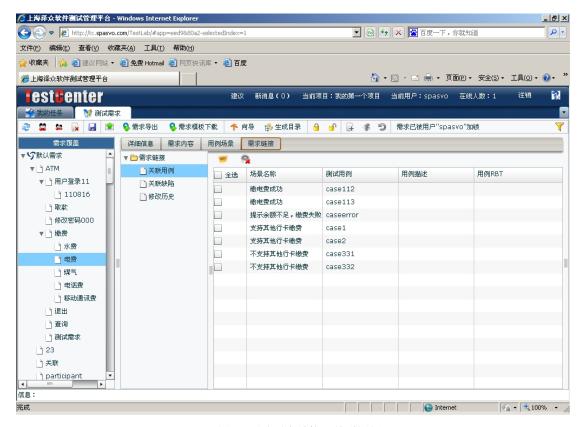


图 3 测试需求链接(关联用例)

在右边显示了选中的需求节点关联的测试案例列表,见上图。

5) 测试主题

要进行测试,就需要知道应用系统的需求是如何定义的,才能够进行功能测试。 但是,功能测试本身包括了众多的内容:

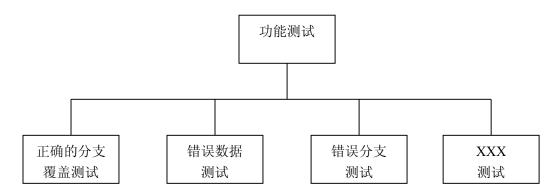


图 4 功能测试的范围

每一种测试需求,都使用不同的测试策略,使用不同的测试案例。

因此,每一种测试都包括一类测试需求。定义测试需求的前提是,根据应用系统的功能点来定义测试需求(还需要首先定义测试策略)。

在 TestCenter 中,用户可以定义不同的测试主题来对测试案例来进行划分。 在通过测试需求创建测试集的过程中,可以通过测试主题对测试案例进行筛选,使 得测试更具备针对性。

6) 测试需求关联测试版本



在测试计划的模版,分成三个层次来管理: 1)发布版本; 2)测试计划; 3)测试执行轮次。

发布版本: 即被测试软件的版本号,它对应了一系列的需求项和需求项的版本。

测试计划:针对需要被测试的版本,规划测试的阶段,如:功能测试、系统测试、验收测试等。

测试执行轮次(简称测试轮次): 在测试计划之下,即本次测试计划需要执行的过程。

7) 创建测试集

TestCenter 支持从需求出发来创建测试集。

一次测试,都具备明确的测试目标、测试范围。对测试目标和范围的描述,都是定义自需求,如下图:

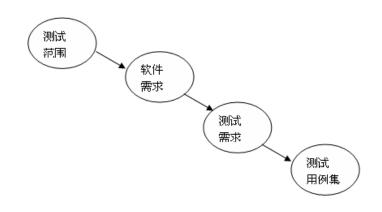


图 5 需求出发创建测试集

测试案例的集合, 称为测试集。测试集组成测试计划。

根据测试案例执行结果,可以计算需求覆盖率:

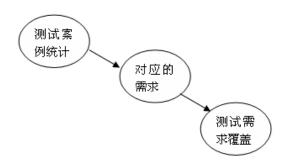


图 6 计算测试需求覆盖率

测试用例设计

1) 设计标准的测试案例

TestCenter 的测试案例定义:测试案例是面向具体的测试目标,定义确切的操作过程对被测试系统进行输入测试案例输入数据、获取返回数据与信息,定义校验规则进行校验,并且能够明确判定测试结果的对象。

测试案例特性:

▶ 面向具体的需求功能点

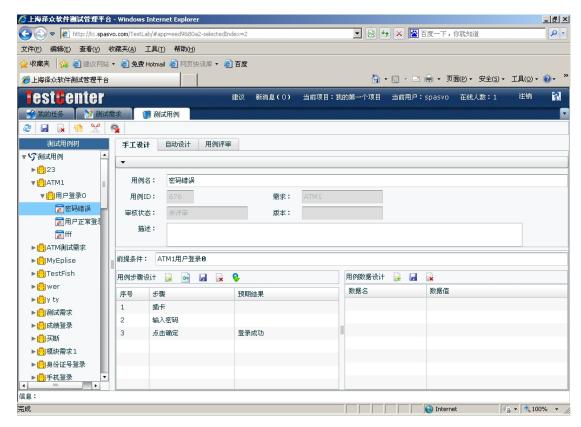
所有的功能测试案例都是面向具体的功能,不允许有测试案例是不面向功能 的。

在 TestCenter 中,理论上不允许存在测试案例,它与任何测试需求树不关联。如果存在这样的测试案例,将会通过需求来创建的任何测试集,都无法包含这个测试案例,也无法根据测试结果来计算需求覆盖率,无法评估系统的风险。

当然,在 TestCenter 中,可以只定义测试案例不定义需求,这样的测试案例,需要用户自己手工维护测试集来管理,并且不考虑测试覆盖率的计算。

▶ 具备具体的操作过程

测试案例必须明确具备操作过程定义,如下图,定义了测试用例执行的步骤:



很多操作界面,都具有明确的屏幕控制逻辑和操作路径,并且这些路径在一个 界面上都不是唯一的。但是对于一个测试案例,必须只能够对应唯一的路径。

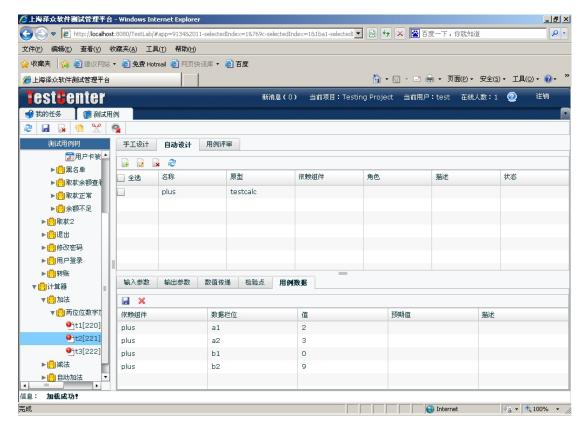
不允许一个测试案例具备不明确的操作路径,或者涵盖多个路径的情况出现。

▶ 明确的输入数据

测试案例的输入数据是唯一的,不允许出现不明确的数据。例如,不能够有测试案例在描述的时候,同时要输入多个数据。

每个用例,只能够具有一个数据组合。

如果特殊情况下,需要循环很多次来执行,那么他具备的循环测试是确定的、 循环的数据也是确定的。



如上图,是测试用例的自动测试的属性,它包括:

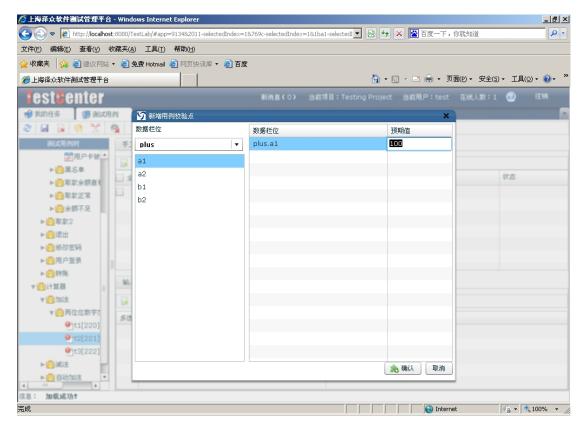
- 1. 测试用例包含的测试脚本,以步骤的方式体现;
- 2. 测试用例的数据,与每个测试脚本的参数对映,有明确的输入数据、验证数据等;

▶ 明确的预期结果和校验规则

在测试过程中,每一步操作,都可以定义预期结果。例如,在输入数据之后,被测试系统执行检查,会进行相应的提示,可以把这些提示信息作为预期结果来处理。

整个测试案例,需要定义一个或者多个校验规则。校验规则用来判断整个测试案例执行完成之后,用例对应的功能是成功还是失败。

总的原则是:测试案例需要能够来自己定义自己的规则,能够自己判定是否正确,而不是需要通过其他测试案例或者手段来检查本用例的执行结果是否正确。



如上图,在自动化的测试用例中,可以对测试用例中的每一个取值设置一个预期结果,在自动执行过程中,会对这个值进行自动验证。

▶ 支持业务流程的测试

TestCenter 的测试案例,支持测试一个业务流程,包括一个或者多个操作过程。

例如,当我们测试一个 ATM 更改密码的功能,就需要:第一步,使用更改密码的功能去修改密码;第二步,退出卡;第三步,插入卡,输入密码,通过查询校验来确认密码修改成功。

在这个过程中,每个功能对应到一个操作步骤,而测试案例应该包括整个过程。

2) 添加业务组件

应用系统是由单个的操作功能组成的,在 TestCenter 中,我们把这个单个的操作对应到业务组件。

▶ 业务组件定义

对应一个操作过程,具备按照一步一步地描述方法来进行描述,它的步骤是明确的。

业务只是一个流程,是没有数据的,不能够执行的。一个业务组件,唯一的对应了一组业务操作数据。我们把交易分支加上它的一组数据。

▶ 操作过程描述

通过 step by step 的方式,描述操作过程。便于测试脚本编写人员能够根据这个过程来编写测试脚本。

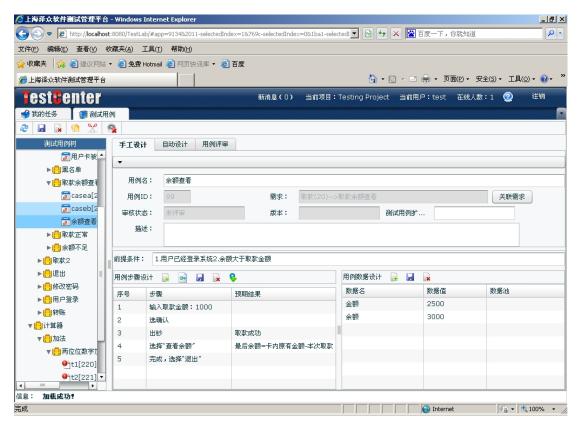


图 7 录入测试案例步骤

如上图,显示一个组件用来修改用户密码的步骤描述。

▶ 自动测试属性

如果业务组件支持自动测试,那么这个业务组件会对应测试脚本。 它的自动测试属性包括:

- ◆ 业务组件的名称。这是业务组件的唯一标示信息。
- ◆ 业务组件的版本号。可以与被测试系统一直。
- ◆ 业务组件的测试脚本。测试脚本对应到一个具体的文件,在 TestCenter 上可以显示这个文件的内容。

点击组件的"脚本"页,可以查看脚本内容,如下图。

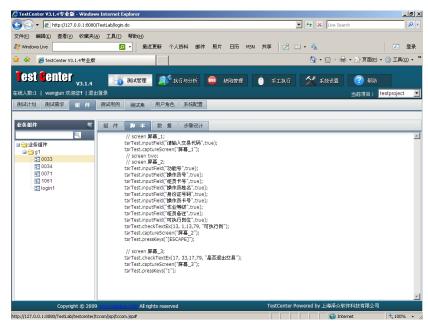


图 8 查看测试脚本

在 TestCenter 中,脚本信息不能修改,只能够察看。

业务组件的对象库。对象库在业务组件中,只体现为一个文件,但不可察看。 业务组件的参数模板。业务组件对应的测试脚本,是一个被参数化之后的测试 脚本。

如下图,是参数模板信息:



图 9 参数模板信息

右边显示为组件的数据模板信息,按照脚本的使用顺序,从上到下排列。

▶ 业务组件管理

业务组件采用目录结构来存放,在 TestCenter 中,显示为一个树状结构。 用户可以增加、修改、删除这些业务组件。

自动化业务组件是实现自动化测试的基础。通过业务组件,测试设计工程师可以轻 松的搭建测试案例,实现测试案例的可配置。

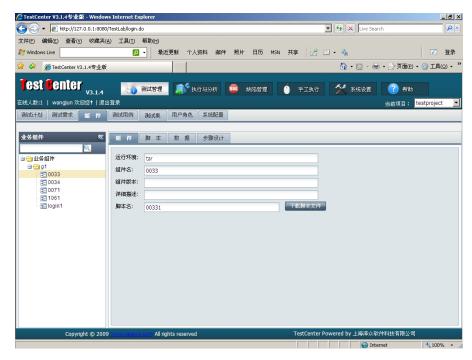


图 10 组件管理

如上图, 左边显示的是基于目录管理的组件管理。

右边显示了组件的基本信息,包括运行环境(对应的自动测试工具类型)、组件名、版本号、详细描述信息。

3) 手工测试案例设计 被测试案例描述测试过程、测试数据。 如下图:

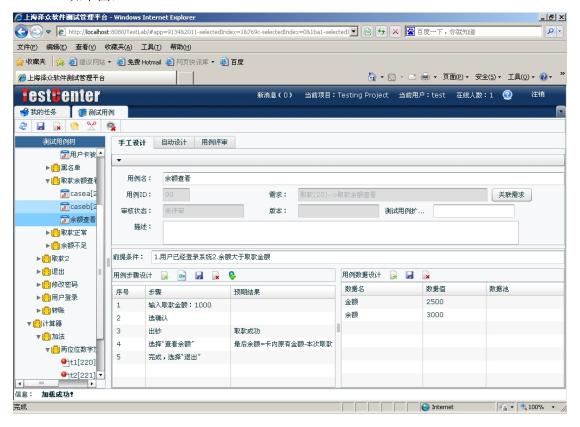


图 11 测试案例信息

右边显示了测试案例的操作步骤、基本描述、前置条件、审核状态。

- ▶ 描述:测试案例的基本描述,说明测试案例的测试目标。
- ▶ 前置条件:描述本测试案例需要运行,需要的先决条件。例如,需要测试从ATM 取款的交易,前置条件必须是卡号存在,余额大于取款金额。
- ▶ 审核状态:测试案例都具备一个"审核"的状态,审核人可以由 TestCenter 配置来决定,审核状态为"通过"和"未审核"。

4) 测试案例组件化

TestCenter 的测试案例采用组件化的方法,也就是通过配置业务组件来实现测试案例。这种方法的最大优势是把宝贵的测试设计工程师从繁杂的测试步骤设计工作解脱出来,去做更有价值的测试案例设计。

TestCenter 的测试案例组件化实现了测试设计与具体实现分离,能够大幅度提高测试设计人员的工作量,提升工作效率。

如下图所示:

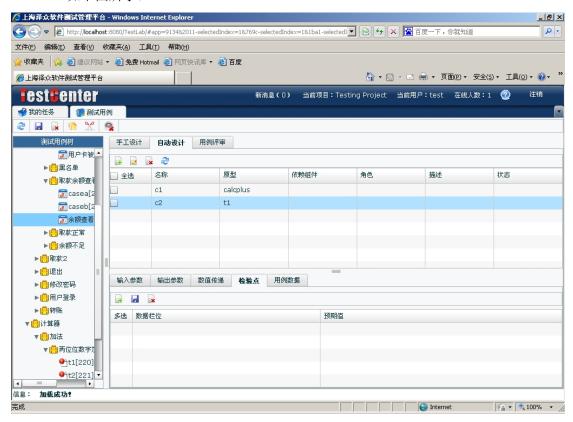


图 12 组件和用例相关联

右边的测试案例"配置"页,显示了测试案例是由业务组件(测试脚本)calcplus、t1配置而来。

说明:

▶ 组件增加与修改

在测试案例中,点击"增加"按钮,就可以给当前的测试案例增加组件。 如图:

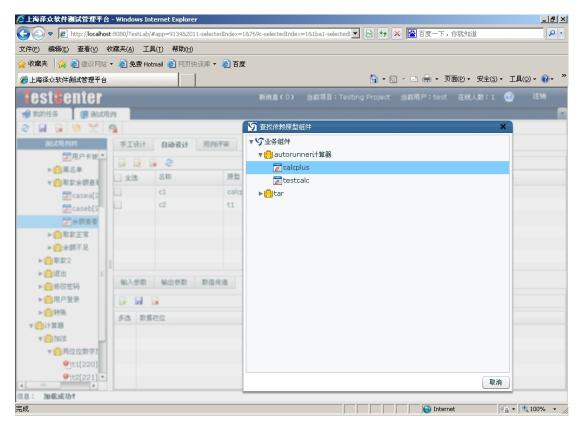


图 13 设置组件原型

弹出了增加组件对话框,在这个对话框上,我们能够通过"原形"按钮来选择 具体的组件原形;

通过点击"修改",也可以修改当前选中的业务组件。

▶ 依赖关系

也可以通过点击"依赖组件"按钮来选择当前组件依赖的其他业务组件。

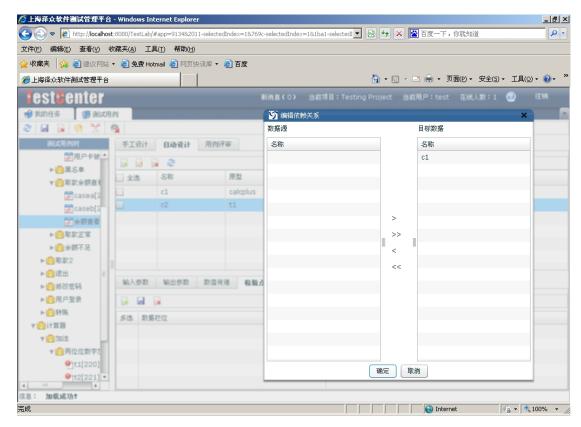


图 14 创建组件依赖

如上图,是定义依赖规则对话框。

测试案例中的依赖关系,就是定义了一个业务组件执行的顺序关系:哪个在前、哪个在后。具体的先后次序关系,是由测试案例设计人员根据具体的测试需求来决定的。

▶ 组件删除

选中测试案例中的某个组建,点击"删除"按钮,就可以删除这个组件。

▶ 组件刷新

选中测试案例中的某个组件,点击刷新,就可以来刷新当前测试案例中的组件。此功能主要是用来:当当前选中的测试案例依赖的组件发生变化,就需要使用原来的业务组件的数据模板来更新当前测试案例数据。

当组件被更新,如果数据模板也需要更新,那么测试案例就一定需要重新刷新; 如果只是更新了测试脚本,就可以不刷新。

5) 准备测试案例数据

当测试工程师配置完成测试案例的组件之后,会自动生成一个测试案例数据,如下图:

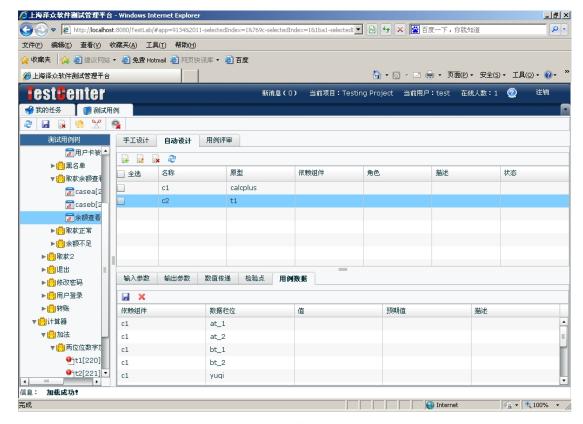


图 15 测试数据

点击"数据"页,就可以查看到测试案例的数据信息。此时,所有的数据都是测试案例的业务组件模板中的缺省数据,使用这个数据进行测试是不行的,测试工程师需要根据测试案例的数据设计来修改这些测试数据,从而保证测试数据的正确性。

测试案例的数据是与测试案例紧密相关的: i)测试案例的组件配置决定了测试案例数据的格式; ii)测试案例的数据是测试案例紧密不可分割的一部分,是保存在TestCenter测试案例数据库中的。

6) 测试案例自动化

测试案例是否是自动化,与测试案例本身没有太多关系,只需要满足以下条件:

- ▶ 测试案例由业务组件构成:
- ▶ 业务组件是自动化的,也就是具备测试脚本的属性。

也就是当测试工程师设计测试案例的时候,不用关心是否是自动化用例,只关注于使用的业务组件就可以了。

7) 测试案例参数与校验规则

在测试过程中,需要进行参数管理,有几种情况:

- ▶ 需要把前面操作过程的输出参数传递给后面的测试过程作为输入;
- ▶ 需要给一个测试案例定义参数,以保证这个测试案例能够在不同的数据环境下使用。例如,测试案例需要一张车票信息,在不同的测试环境下,车票与根据售票地点相关的;因此就需要把车票信息作为测试案例的参数。这个参数使得这个测试案例能够在测试环境下使用。
- ▶ 测试案例需要输出参数,给后续的测试案例。 如下图:

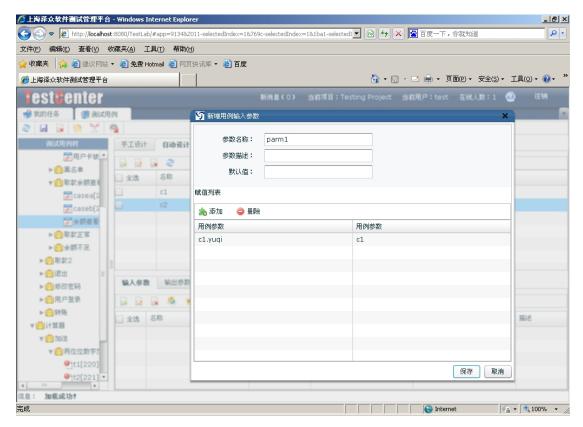


图 16 用例输入参数添加

表示了测试案例具有输入参数、输出参数、参数传递等功能。

▶ 输入参数:

定义了输入参数的名字、输入参数传递给哪个组件的数据。

▶ 输出参数:

定义了输出参数的名字,从哪个参数的数据传递给输出参数。

▶ 参数传说:

定义了数据从哪个组件的哪个参数, 传递给其他组件(名称)的参数(参数名)。

8) 创建测试案例集

测试案例是依赖于一个具体的测试需求,对于一次测试而言,需要的是对众多的需求点进行测试。因此,就需要把很多测试案例集中到一起来管理、运行。

多个测试案例组成的集合,就是测试集。

测试集的创建方式:

- ▶ 手工创建,手工在测试集中增加测试案例;
- ▶ 通过测试需求来创建。通过需求节点来创建测试集,能够把测试案例和需求覆盖关联到一起,也能够保证测试覆盖和不发生测试案例遗漏。如下图,为测试集 setaa:

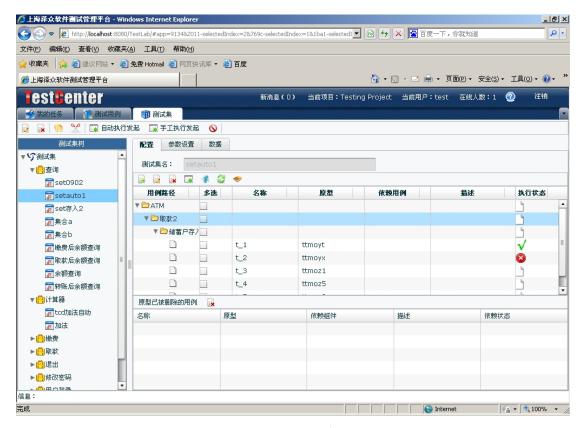


图 17 测试集

它采用树的形式来显示测试用例,能够清楚的知道测试用例所属的树形结构。

- 9) 与第三方测试设计工具接口
 - ▶ 生成测试案例的接口,包括接口类和接口文件说明;
 - ▶ 能够生成手工测试步骤描述;
 - ▶ 能够生成与业务组件关联的测试案例:
 - ▶ 能够生成业务组件来对第三方测试脚本进行封装;
 - 能够支持业务流方式的测试案例和单个面向功能操作的测试案例;

10) 审核测试案例

测试案例作为主要的测试对象,在测试案例设计完成之后,需要进行评审,作为质量控制点。

审核的规则为:

- 测试经理有权限对测试案例的审核状态进行操作,达到审核通过、取消审核等目标:
- ▶ 测试案例在测试设计完成之后,均为未审核状态;
- 测试案例评审通过之后才能够修改为审核状态;
- ▶ 通过审核的测试案例不能够删除、修改。

11) 测试案例风险值

需要为需求定义风险值 RBT (Risk business testing value)。 建立测试案例评估模型:

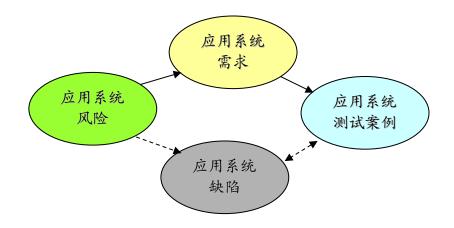


图 18 风险传递

如上图,应用系统的风险最终体现在缺陷。应用系统的风险,从软件工程的角度, 首先传递给应用系统的需求,需求再传递给测试需求和测试案例。

因此,只需要定义需求的风险,通过一个算法就可以计算出测试案例的 RBT 的值:

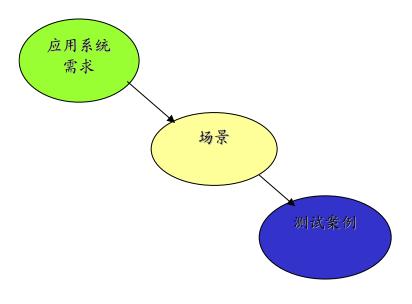


图 19

具体的方法,是根据测试工程师输入的需求的 RBT 值,计算场景的值,再根据场景来计算测试案例的 RBT 值。

在执行一个测试之后,就可以估算到:

- ▶ 本次测试(所有测试案例)的 RBT 值与整个系统(测试需求 RBT 值综合)的覆盖率;
- ▶ 本次测试(所有测试案例)的 RBT 值与所选需求的覆盖率。

通过这两个覆盖率,可以估算本次测试活动的覆盖率指标,并且通过这个指标来评估测试的风险情况。

建立测试计划与测试数据场景

1) 创建测试计划

测试计划是用来定义一个测试活动如何开展的对象。它从属于发布版本。 测试计划包括多个执行轮次,每个执行轮次代表了一次执行的信息。 执行轮次可以包括多个不同的测试集。测试集代表了一类测试用例的集合,便于我们分配执行用例。



图 20 测试计划

2) 察看测试计划执行进度

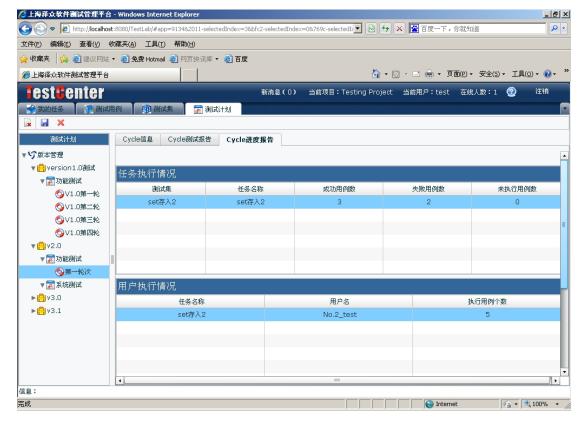


图 21 测试执行进度

自动测试执行

执行测试计划,如下图所示。

- ▶ 选择要执行的测试计划,通过上下文菜单选择自动执行;
- ▶ 在弹出的自动执行窗口上输入一个执行的名称;
- ▶ 打开 Test Agent, 使 Test Agent 连接测试管理系统服务;
- ▶ 点击测试器情况 Tab 页,点击加载按钮得到执行器列表;
- ▶ 选择本地执行器,点击开始。

| 运行名称: | 计划名称 | ⊼: |
|-----------|-------------|-------------|
| run101 | plan01 | |
| 运行描述: | | |
| | | |
| 息运行情况 测试器 | 器情况 | |
| 开始时间: | 运行时间: | 剩余时间: |
| | 0天00时00分00秒 | 0天00时00分00秒 |
| 总用例数: | 成功用例数: | 失败用例数: |
| | 0 | 0 |
| | 用例进度: | |
| 总组件数: | 成功组件数: | 失败组件数: |
| | 0 | 0 |
| | 組件进度: | |
| 运行记录: | | |
| | | _ |
| | | |
| | | V |
| | | |
| | | |



图 22 自动执行对话框

触发自动执行后,测试管理系统将驱动本地的测试工具执行测试计划中的自动化测试脚本。在执行的过程中,运行对话框的总运行情况,Tab 页不断刷新以显示执行的情况。例如:执行组件数、案例数,成功、失败的案例数、组件数,运行时间等等。

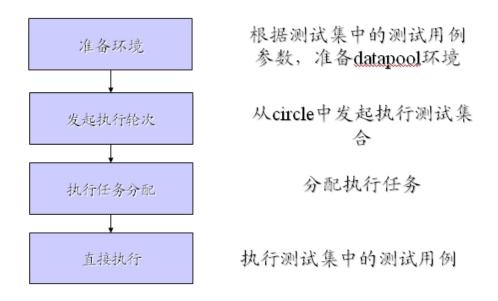
执行完毕,测试管理系统将结果保存在执行报告里。也可以通过执行日志查看简要 的测试结果,见下图。



图 23 计划执行日志

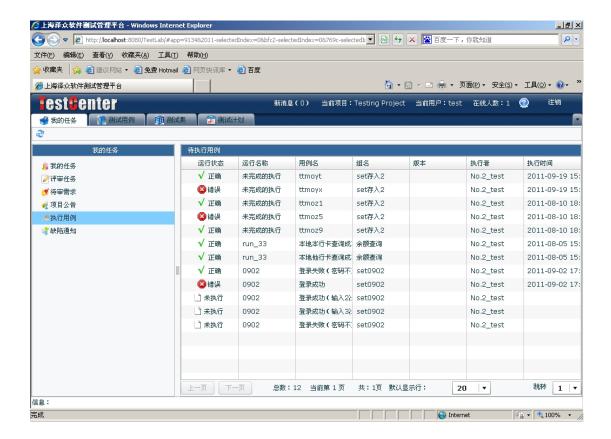
手工测试执行

TestCenter 能够管理测试执行的过程,流程如下:



对于不同规模的测试执行团队,可以支持不同的执行流程。

分配执行任务之后,测试工程师可以在他的面板上看到分配给他的测试执行任务列表,如下图:

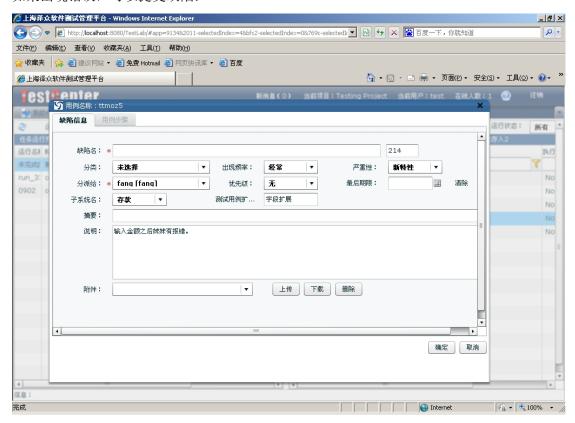


点击执行任务,可以执行这个任务:



执行任务,就需要根据执行的步骤来依次执行。对于出现的错误,需要在"实际结果"栏目中填写。

如果出现错误,可以提交缺陷:



填写缺陷的内容,可以提交缺陷。缺陷中的关联用例,就是提交缺陷的关联测试用例。

测试分析与报告

测试结果评估通过分析统计报表来实现。

1) 测试报告

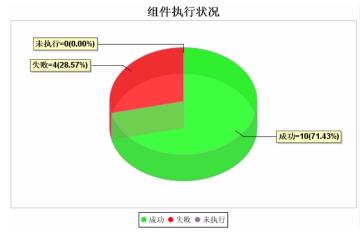
▶ 自动测试报告 在执行完毕后创建自动测试报告,使用表格和图表的形式展示执行的数据统计结果,如下图所示。



A 运行基本信息

| | 运行总数 | 成功数 | 失败数 | 未执行数 | 验证失败数 | 成功率 |
|--------|------|-----|-----|--------|-------|-------|
| 组件 | 14 | 10 | 4 | 0 | | 71.4% |
| 用例 | 7 | 4 | 3 | 0 | 0 | 57.1% |
| 需求 | 0 | 0 | 0 | 0 | 0 | 0.0% |
| 需求运行总数 | | | 项 | [目需求总数 | 需求覆 | 盖率 |
| 0 | | | 1 | | 0.0% | |

B 执行统计表格



C 执行统计饼图

图 24

自动测试报告经过统计得到各种指标反映执行结果,包括:组件的执行成功率、用例的执行成功率、测试需求的成功率和覆盖率。

▶ 自动运行日志

在执行结束后记录自动运行日志,在日志中列出所有执行过的案例列表和组件,并 且保存执行过程中的截屏。通过自动运行日志,分析执行失败的原因,判断是否存在缺 陷,见下图。也可以通过手工的方式验证错误的组件。

| ∓h∖∓ | 行报表图 | | | | | | |
|------|---------|------------|------|--------|-----------------------|------|--------|
| PULE | 运行人 | 所属计划 | 运行名称 | 用例名 | 结束时间 | 运行状态 | 所有记录 ▼ |
| | mark | plan2 | run9 | 一本通连跑2 | 2010-04-13 17:46:48.0 | 运行正确 | |
| E | mark | plan2 | run9 | 一本通连跑3 | 2010-04-13 17:46:48.0 | 运行正确 | _ |
| + | mark | plan2 | run9 | 一本通连跑7 | 2010-04-13 17:46:48.0 | 执行失败 | 提交BUG |
| | 用例組件截图 | | | | | | |
| | 组件 | 学 名 | 截图名 | 3 | | 截图 | |
| | T10 | 061 | 屏幕_ | 3 | 点 | 击查看 | |
| | T10 | 061 | 屏幕_ | 4 | 点 | 击查看 | |
| | T10 | 061 | 屏幕_ | 5 | 点 | 击查看 | |
| | T10 | 062 | 屏幕_ | 1 | 点 | 击查看 | |
| | T10 | 062 | 屏幕_ | 2 | 点 | 击查看 | |
| | T10 | 062 | 屏幕_ | 3 | 点 | 击查看 | |
| | T10 | 062 | 屏幕_ | 4 | 点 | 击查看 | |
| | T10 | 062 | 屏幕_ | 5 | 点 | 击查看 | |
| | T10 | 062 | 屏幕_ | 6 | 点 | 击查看 | |
| | T10 | 062 | 屏幕_ | 7 | 点 | 击查看 | |
| | T10 | 062 | 屏幕_ | 8 | 点 | 击查看 | |
| | 用例验证表达式 | | | | | | |
| | 用例名 | | 效验名 | | 表达式 | 校验结 | 果 |
| # | mark | plan2 | run9 | 一本通连跑8 | 2010-04-13 17:46:48.0 | 执行失败 | 提交BUG |
| Ŧ | mark | plan2 | run9 | 一本通连跑 | 2010-04-13 17:46:48.0 | 执行失败 | 提交BUG |
| 1 | mark | plan2 | run9 | 一本通连跑9 | 2010-04-13 17:46:48.0 | 运行正确 | |
| E | mark | plan2 | run9 | 一本通连跑1 | 2010-04-13 17:46:48.0 | 运行正确 | _ |

图 25 点开案例可以看到执行的组件截图以及校验点情况

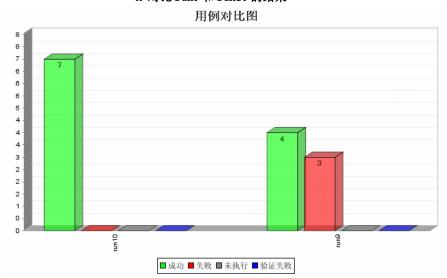
如果确认有缺陷,将缺陷提交到缺陷跟踪系统,然后该缺陷将于当前案例关联。

▶ 测试对比报告

对比多次执行结果,进行差异分析,评估缺陷修复情况,见下图。

| 选择用例 | 计划名称 | 运行名称 | 运行时间 | 手工/自动 |
|----------|-------|------------|-----------------------|--------|
| | plan1 | run1 | 2010-04-13 16:40:58.0 | 自动 |
| V | plan2 | run10 | 2010-04-13 17:52:47.0 | 自动 |
| | plan2 | run11 | 2010-04-13 17:57:17.0 | 自动 |
| | plan1 | run2 | 2010-04-13 16:43:32.0 | 自动 |
| | plan1 | run3 | 2010-04-13 17:10:22.0 | 自动 |
| | plan1 | run4 | 2010-04-13 17:11:48.0 | 自动 |
| | plan1 | run5 | 2010-04-13 17:19:21.0 | 自动 |
| | plan1 | run6 | 2010-04-13 17:27:30.0 | 自动 |
| | plan1 | run7 | 2010-04-13 17:28:32.0 | 自动 |
| | plan1 | run8 | 2010-04-13 17:31:18.0 | 自动 |
| V | plan2 | run9 | 2010-04-13 17:46:48.0 | 自动 |
| | plan1 | run1 | 2010-04-13 15:14:12.0 | 手动 |
| 页一寸 页一 | | 当前第1页,共1页, | 总共11条记录。 | 跳转至: 1 |

A 对比 run9 和 run10 的结果



B 对比结果显示 run9 有 3 个用例失败 run10 都运行通过了 图 26

2) 缺陷报告

▶ 状态统计表、图

按照缺陷的各个状态统计缺陷的分布情况。通过状态统计图表,可以方便地找到解决缺陷、遗留缺陷、重开缺陷等等处在不同状态上的缺陷数量,见下图。开发人员重点关注已分配的缺陷。测试人员也可以通过这个统计图表分析缺陷的走势。

| Bug 状态统计 | -表 | | | | | | 忌出pDE |
|----------|----|-----|------|------|-----|-----|-------|
| 新建 | 三 | 己分配 | 正在修改 | 修改完成 | 己驳回 | 己重并 | 己关闭 |
| 30 | 4 | 9 | 4 | 16 | 19 | 1 | 352 |

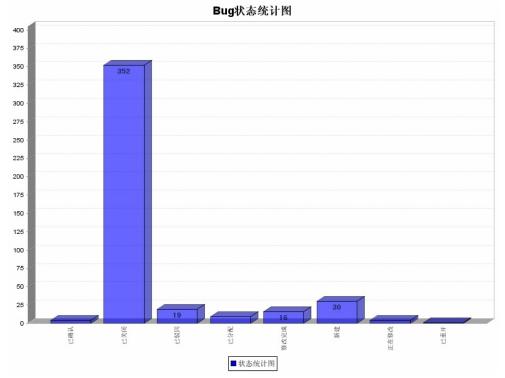
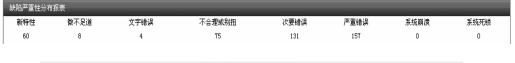


图 27 缺陷状态统计图、表

▶ 严重性统计表、图

严重性统计图、表统计不同严重性的缺陷数量。利用严重性统计图、表可以增强对严重性高的缺陷进行跟踪和解决,如下图所示。



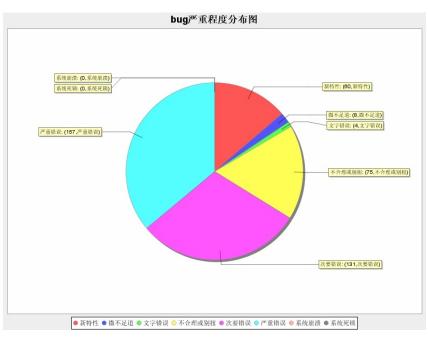


图 28 严重性统计表、图

测试人员报告缺陷数

通过人员报告缺陷数统计表,了解每个测试人员报告缺陷的情况(见图 48)。

| 测试人员报告Bug数目统计表 | | |
|--------------------|-------------|----------|
| 测试人员登录名 | 测试人员 | 报告的bug数量 |
| zhangjf | zjf | 191 |
| mark | mark | 8 59 |
| wxgt lixs | ***g 李晓松 | 38 |
| zhangpj 1 i kun | 张配杰 李昆 | 42 |
| likun wangyf | 学院 | 44 53 |

图 29 人员报告缺陷数统计表

▶ 定期缺陷统计图

生成定期缺陷统计图来表示每日或每周新增的缺陷数,从而对当前系统质量做出评估,见下图。

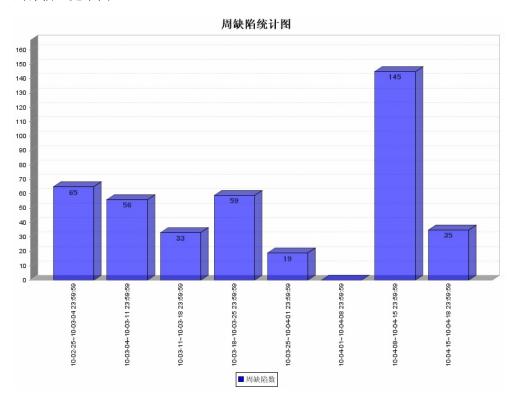


图 30 定期缺陷统计图

▶ 缺陷新增、关闭图形

通过缺陷新增、关闭图形分析缺陷的发展趋势, 见下图。

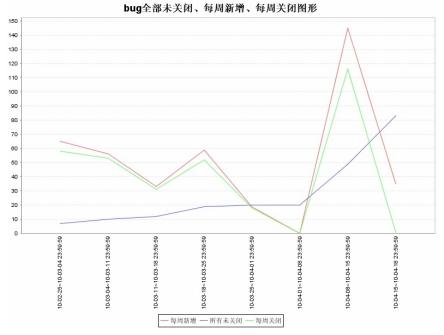


图 31 缺陷新增、关闭图形

测试度量

测试质量的度量通过三个覆盖率指标体现,见下图:

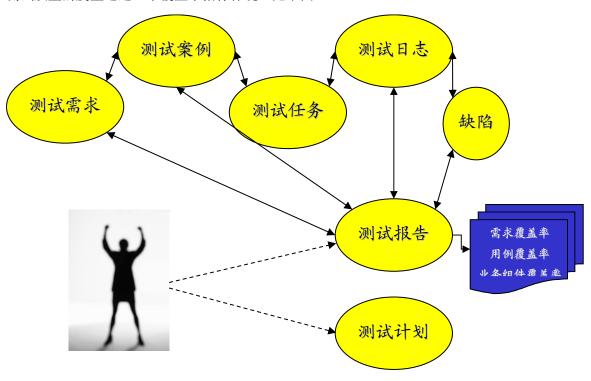


图 32 测试度量

1) 需求覆盖率

需求覆盖率,是衡量了本次测试对整个需求覆盖的比例。

TestCenter 支持对不同的需求制定不同的 RBT (Risk Business Testing)的值。例如:对于活期存款的需求,可以定义非常高的 RBT;对于教育储蓄需求,设定的 RBT 就相对低。

不同需求的 RBT,反映了这个需求对整个应用系统的风险影响,主要体现在需求使用频度和复杂度方面。

测试案例是附属于需求,因此测试案例也具备 RBT 的属性。

需求覆盖率=SUM(执行通过的测试案例 RBT)/SUM(需求 RBT)。

测试需求覆盖率,能够很好的反映本次测试对需求覆盖的情况。

2) 用例覆盖率

用例覆盖率 = 执行通过的测试案例个数/本次执行的测试案例的个数。

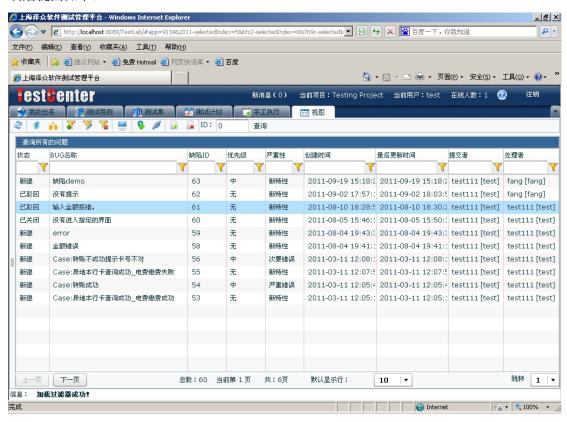
3) 业务组件覆盖率

业务组建覆盖率 = 执行通过的组件个数 / 总的组件个数。

缺陷管理

TestCenter 的缺陷管理,包括: 1) 缺陷视图; 2) 提交缺陷; 3) 缺陷合并; 4) 缺陷过滤器; 5) 缺陷报告与分析: 6) 定义缺陷流转流程: 7) 缺陷流转。

缺陷视图如下:



缺陷视图, 支持查看缺陷、操作缺陷、导出缺陷等工作。

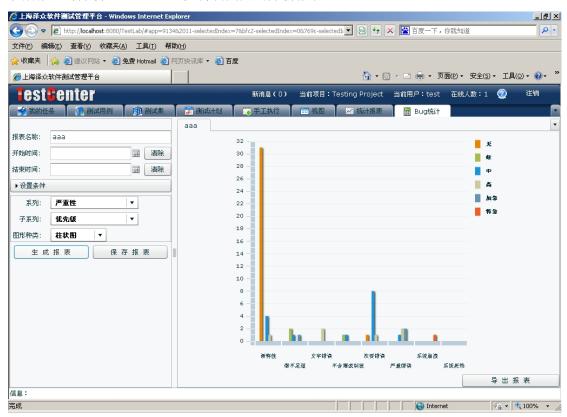
提交缺陷。可以从缺陷视图的界面上发起。

合并缺陷。是可以把多个缺陷合并成一个缺陷,一般用于缺陷提交重复的情况。

缺陷过滤器。用户可以定义公用的缺陷过滤器和个人的缺陷过滤器。缺陷过滤器主要用来过

滤到自己关心的缺陷,屏蔽掉不关注的缺陷。TestCenter 的缺陷过滤器支持图形的自定义和输入 sql 语句处理。

缺陷报告与分析。TestCenter 支持自定义的分析报表。



如上图,用户可以输入各种组合条件,生成不同图形类型的报表,主要支持:柱状图、表格、饼状图、折线图等。