TestCenter 测试管理软件

技术白皮书
Version5.5

上海泽众软件科技有限公司

2018 年 1 月

目 录

1. 问题提出与说明

1.1. 背景

随着软件规模的发展和对软件系统的依赖，人们发现：软件的质量对应用系统的影响日益增加，质量存在问题的软件会导致帐务出错，客户信息丢失，用户的服务出错。

因此，提高软件的质量成为一个重要的问题。而测试正是提高软件质量的有效手段。数据显示，在一个软件开发过程中，测试占到整个工作的 40%－60%。所以，如何能够在比较低成本的情况下大幅度提高测试的质量，对软件的最终质量起到非常重要的作用。

另一方面，当应用软件投入使用之后，随着应用的不断发展和变化，将会提出大量的新增需求。新功能对用户非常重要，能够给用户不断发展的业务提供更强大的支撑。

当开发人员修改软件的功能、增加软件功能，新增功能部分导致原有系统运行不够稳定的几率必然增加，可靠性降低：由于修改一个小错误造成大量业务无法正常运行的情况。这就需要做大量的回归测试来保证系统的可靠性，通过回归测试验证以往的功能是正确的、可靠的。

传统的测试工作是采用文档方式来进行管理的，我们经常使用 word、excel 等来编写和管理测试用例。这种方式在测试发展的初期非常普遍，它适用于对应用系统的质量要求比较低、测试工程师人数比较少、对测试用例的要求比较低的情况。随着软件质量越来越关键，以往的方法出现了问题：

1.缺陷管理的力度不足。对测试过程中产生的缺陷，没有进行登记、编号，并且采用标准化的流程进行跟踪，无法确保每个缺陷都已经被关闭。遗漏的缺陷对软件的正常使用是非常重大的威胁。

2.测试用例缺乏规范性。使用 word 或者 excel 来编写测试用例，使得测试用例的规范性无法得到保证，因此会导致：1）无法在测试工程师之间共享测试用例。当我们需要重新进行一次测试的时候，就需要重写测试用例，造成成本增加；2）无法对测试用例进行质量控制，造成测试用例本身的质量难以控制；3）造成了测试用例对人员的依赖，测试人员的流动会造成测试用例失效；4）测试用例

作为一种资产被浪费。

3.测试过程难以进行管理。测试工作需要遵循测试流程，以保证每个需求都可以被覆盖；每个测试用例可以被执行。通过 word 和 excel 方式的方法，无法对流程进行管理。

4.测试需求管理没有建立。基于 word 或者 excel 格式的需求，不是一个条目化的需求，无法对每个需求项进行跟踪管理，如对需求项进行变更管理。另外，当需求发生变更，也无法确定哪些测试用例需要重新设计、哪些测试用例需要回归。

5.自动测试无法实现。自动测试是测试发展的一个方向，通过自动测试我们能够大幅度的提升测试覆盖率，减小测试的颗粒度，对于提升软件的质量非常重要。显而易见，word 格式的测试用例无法实现自动化测试。

1.2. 解决方案

企业可以建立一整套软件测试体系，包括：需求管理、测试分析、测试管理、缺陷跟踪，并且把这个过程纳入整个软件项目开发和软件产品开发过程。

实际上，在 CMM 的规范中，测试本身就是 SQA 的一部分。

测试管理软件是支撑 SQA 的重要工具，它应该包括以下功能：

- 管理测试需求

测试管理软件能够对测试需求进行条目化管理，按照需求树的方式来组织测试需求。

测试需求支持导入、导出，能够从任何一个需求树节点来导入、导出需求。

测试需求关联到测试用例、关联到本需求所产生的缺陷。

测试需求还应该能够关联到场景。

每个需求项（需求条目）可以具有详细的描述信息，也具备频度、状态（已建议、已修改……）等状态。

- 管理测试用例

能够建立与需求同构的测试用例树。

支持测试用例的描述，如：测试目标、所属需求、测试步骤、测试数据，与自动化相关联的信息等。

- 执行测试

能够把测试用例按照一个测试目标进行组织，形成测试集，并且能够执行测试集。支持测试执行流程。

- 管理缺陷

能够对缺陷管理过程进行工作流的管理。

支持自定义管理流程、自定义缺陷状态、自定义用户角色。

支持对缺陷进行检索、合并、导出；

支持缺陷的各种报告。

- 管理测试计划

支持测试计划管理，每个测试计划都包括测试资源、测试任务、测试轮次管理等，同时支持通过测试计划查看测试报告。

- 查看测试报告

能够根据自定义的格式生成测试报告。

2. 概述

1、本技术白皮书适用于上海泽众软件科技有限公司自动测试管理工具（TestCenter Version5.5）。

2、本技术白皮书是上海泽众软件科技有限公司测试管理工具（TestCenter）的技术说明，也是技术谈判的主要内容，是采购方询价、系统选型以及系统测试和验收的主要技术依据。

3、本技术白皮书是根据信息产业部颁布的有关技术体制和技术政策并结合上海泽众软件科技有限公司的实际情况制定的。本技术白皮书没有提出而信息产业部的技术体制以及技术标准已有具体规定的内容，应按信息产业部的技术体制以及技术标准执行，如果存在不一致应以信息产业部颁布的最新技术体制及技术标准内容为准。

4、本技术白皮书在内容或技术指标上如果存在错误（包括印刷错误），经双方确认后可对该错误内容或技术指标进行修正。

5、测试管理工具（TestCenter）版本升级之后，上海泽众软件科技有限公司有权对本技术白皮书进行修改，并不需要主动通知用户。

6、本技术白皮书以下内容为用户重点考察内容：

软件的功能、性能、技术指标和环境要求；

设备容量计算和配置方法；

所提供的数据库的功能和性能指标；

软件安装要求；

提供软件的接口、协议等工程技术要求；

乙方供货范围、交货能力和时间、运输、安装、调测验收和培训等项内容的日程安排；

其他有关技术资料。

7、本软件对涉及专利、知识产权等法律条款承担有限责任。

8、本技术白皮书提供了对上海泽众软件科技有限公司的测试管理工具（TestCenter）的相关技术描述，由于用户使用造成损失，上海泽众软件科技有限公司不承担责任。

9、本技术白皮书以中文编写，未经上海泽众软件科技有限公司同意或授权的其它语言或形式的技术白皮书无效。

10、本技术规范书的解释权归上海泽众软件科技有限公司。

2.1. TestCenter 产品定位

TestCenter 是测试管理工具，它基于 B/S 体系结构，管理测试需求、测试构建、测试计划、缺陷管理、测试资源管理等模块，并且提供多任务的测试执行（包括手工执行和自动执行），最终生成测试报表。

TestCenter 可以和本公司的测试工具 AutoRunner 集成，也可以和其他测试工具集成，提供强大的自动化功能测试。

使用用户是所有需要提高软件开发质量的软件公司、软件外部企业，以及提供测试服务的部门。

2.2. 测试管理的概念

- **测试要求和测试计划**

任何一次测试都需要测试计划，测试计划明确了一次测试需要的内容：

1、测试目标。明确测试所要达到的目的，比如测试某个子系统的所有正常

的交易是否能够正常执行。

2、测试范围。定义那些测试主题是应该被测试的，哪些不需要包含在测试范围内。

3、测试环境。包括测试的硬件环境、网络环境、操作系统、数据库环境、应用系统环境、应用系统数据环境。最关键的是设置应用系统的数据环境，以保证测试用例能够在这个环境下正确执行，保证预埋数据能够保证测试用例正常执行。

4、测试过程计划。一次大规模的测试会划分成多个阶段，如需要测试不同营业日的应用，就需要对测试过程进行计划，协调自动测试执行过程和手工修改环境的过程，以保证测试能够正确执行。

5、测试过程中的环境备份与恢复。在测试过程中，一个阶段的测试完成之后，可能会需要把所有的测试环境（应用系统数据）备份下来，便于下一个阶段的测试。因为下一个阶段的测试需要上面的测试环境，这样，在下一个阶段自动测试执行过程中，如果出现了错误，可以使用备份的数据重新执行此阶段的测试，而不需要执行所有的测试过程。

6、测试验证。测试结果是否正确，除了可以通过自动测试工具之外，也可以通过用户自定义的工具/程序来验证。在测试计划中，要考虑到其他的测试验证方法和手段，并且在测试过程计划中加入这个部分。

测试计划中的测试目标和测试范围，都来自于测试要求。测试要求就是要确定测试的目标和测试范围。

- **测试需求**

功能测试的目标是验证软件是否满足定义的需求。

软件的需求，往往包括每个独立的、具体的、操作的功能，也包括由操作功能组成的一个业务流程，并且明确定义业务流程正确性的标准。

测试需求，就是针对软件需求的两个部分进行测试，定义明确的测试方法来测试各个业务流程的各种业务情况下是否正确、定义明确的测试方法来测试操作流程是否正确。

测试操作流程的部分，基本上通过手工测试可以完成，自动化的测试要定义几个主要的测试流程进行测试。

- **测试用例**

测试用例是具体测试需求的实现，即通过业务组件实例的流程和相互关系来定义测试过程、验证标准。

测试需求定义了“做什么”，测试用例是“如何做”。

- **业务组件**

应用系统，是由具体的界面交互流程组成，我们把每一个界面交互流程，成为一个业务操作过程，对应了测试中的一个对象，我们称为“业务组件”。

一个具体的业务功能，往往包括多个业务组件。例如：网上订机票系统，包括多个业务组件组成。

业务组件，核心是描述了一个界面交互的过程：一系列的输入和输出界面和组件。对于输入和输出数据来说，它还是一个交互过程数据的模版，用户只要填写这个模版得到一组数据，就可以成为一个业务组件的实例。

在 TestCenter 中，业务组件由：测试脚本（bsh/java）、测试脚本相关的资源文件（xml）、业务组件数据模版（csv）组成。

- **测试集**

测试集，就是根据测试要求，对测试需求进行筛选，最终得到一组对应的测试用例的集合，我们称为“测试集”(TestSet)。

执行测试集（可以手工执行和自动执行），就会帮助测试人员完成一次测试。

- **测试执行**

测试用例集合建立的目标是测试执行。TestCenter 支持自动执行和手工执行。自动执行需要连接到自动测试工具，通过运行测试脚本来实现测试用例执行；手工执行是通过人工的方式来执行测试用例。

- **测试报告**

测试执行之后，需要了解测试执行的信息，包括：那些测试用例被执行了、那些没有执行、那些执行过程中出错了、那些成功了等等。

在这个结果上，我们还需要知道那些测试集合执行覆盖了多少业务组件，覆盖业务组件的百分比是多少；覆盖了多少测试需求，比例是多少。

2.3. 业务提供

所谓业务提供，就是指使用 TestCenter 能够完成哪些工作，适合完成哪些工作：

2.3.1. 适用性说明

TestCenter 适用于：

1、功能测试。TestCenter 的核心是完成功能测试管理，包括支持开发/定义测试用例的流程、跟踪测试需求、执行测试用例、得到测试报告、测试日志分析和测试报告。

2、软件开发过程中的功能测试。TestCenter 提供了组件化的方法和对业务流程测试组件化的支持，能够快速的根据应用系统的修改来构建测试用例，满足开发节奏的要求，和开发同步进行每日测试，提供测试效率和开发效率，并且提高测试质量。TestCenter 还提供了单独执行某个测试用例的功能，满足程序员级别功能测试的要求。

3、软件维护中的回归测试。在软件维护的过程中，经常会引入错误，这就需要进行回归测试来避免关键的业务系统发生错误，导致系统异常，无法正常使用。TestCenter 能够帮助用户在开发的时候就建立完成的测试用例库，然后在每一次修改完成后，只需要维护很小的、被修改的需求（以及相关的测试用例），就可以实现回归测试。

4、产品升级中的回归测试。在产品升级的过程中，如果是由于技术方案的升级需要回归测试，TestCenter 可以重用几乎所有的测试用例，只需要重新编写业务组件就可以完成；如果是针对功能修改，只需要增加相关的业务组件和测试用例，就可以完成复杂的回归测试用例编写工作。

5、版本发布测试。TestCenter 能够在一组测试用例的基础上，创建不同的回归测试用例集，协助用户对版本发布测试的不同测试要求进行测试，满足版本发布测试的需求。

2.3.2. 标准化测试用例和测试用例实现过程

TestCenter 的基本功能，就是对软件进行功能测试、系统测试。

TestCenter 管理从应用系统需求出发，进行设计测试用例和管理测试用例的过程。TestCenter 定义了一个标准的过程，用来设计功能测试的测试用例，管理测试用例，复用测试用例。

在得到对应测试需求的测试用例之后，根据测试要求，通过执行测试用例集合来完成功能测试。

TestCenter 在功能测试中的最大意义，在于定义了一个标准的测试过程或者说创建测试用例的过程，就像软件工程之于软件开发一样，TestCenter 提供的方法和过程就是指导测试用例的创建，并且保证了测试用例的质量，也保证测试用例能够达到测试应用系统需求的目标。

在不使用测试管理工具的情况下，每个测试人员（或者兼职的测试人员）都会使用自己的格式和方法来定义测试用例，结果是导致测试用例无法被其他测试人员复用。

另外一个结果，是导致无法审核测试用例本身是否能够达到测试目标，无法保证测试的质量。

第三个结果，是导致用户不知道是否设计了足够的测试用例，来测试应用系统的需求。

TestCenter 不仅规范了测试用例的开发过程，也规范了测试用例本身。TestCenter 规范了测试用例的开发过程，它把测试用例本身划分成为由多个业务组件组成的过程，每个业务组件都被数据模版标准化了，用户开发测试用例的过程，就是在标准的数据模版上填写测试数据和设计测试过程。

输出的测试用例是标准的，也是可以被复用和评估的，从而在测试用例设计和实现的角度，就提高了测试的质量。

2.3.3. 自动化的回归测试

由于软件开发是面向用户需求的，而用户需求也是不断变化的。修改软件会经常性的引入错误，根据统计，每修改 3 个错误可能会引入 1 个错误。

虽然修改了很小的一部分，却存在引入巨大错误的风险。防范风险的手段就是回归测试。

手工回归测试往往需要大量的人力才能够实现，这就出现了：减少测试（降低了成本）就增加了风险；降低了风险（引入大量测试人员进行全面的回归测试）就会增加成本。

采用 TestCenter 可以很好的解决这个问题：

- **根据需求构建 TestSet**

每次回归测试的目标都不相同。例如，修改了某个子系统的一些功能，我们就需要对整个子系统作完整的回归测试，同时对与此修改功能相关的系统作一定范围内的回归测试，对其他基本没有太大关系的系统作非常简要的测试（抽取很少量的、具有代表性的测试用例或者最常使用的功能的测试用例）。

TestCenter 包含了对测试需求的管理。当发起一次回归测试的时候，根据测试要求对测试需求进行检索，得到一个测试用例的集合：一个 TestSet。

执行这个 TestSet 就可以覆盖测试要求的内容。

- **根据环境设置 TestSet 的数据场景**

TestSet 中的测试用例，很多都是具有参数的，这些参数就定义了 TestSet 的数据场景。

当数据场景发生变化，TestSet 就可以执行在不同的测试场景下。TestCenter 的这种 TestSet 定义方法很好的保持了 TestSet 对测试场景的独立性，保证一个 TestSet 可以在不同的测试场景下执行。

- **给 TestSet 分配执行资源**

在需要执行 TestSet 的时候，需要明确执行的资源，一般情况是一台设备上面只能有一个用户在使用，也就是一个虚拟用户在操作。

TestCenter 的资源分配就是给一个具体的 TestSet 分配资源：使用多少个客户端来执行，每个客户端的 IP 地址等。

只有分配了资源，才能够执行。能够分配资源的前提是，我们认为任何虚拟用户可以使用任何一台设备来执行，因此，不需要为每个测试用例来分配资源，而只需要分配一个使用多少设备来执行就可以。

- **动态资源分配，提高执行效率**

TestCenter 执行 TestSet 的时候，某个功能执行在哪台设备上，是 TestCenter 根据当前 runtime 的具体情况来调度，决定在哪台设备上执行哪个测试用例的。

动态的执行和资源分配，能够极大地提高测试效率，防止死锁出现。另外，还可以防止静态资源分配，给某个设备分配了太多的任务，而给其他设备分配的任务太少，从而导致测试效率降低。

2.3.4. 每日构建与冒烟测试

程序员往往通过单元测试来对他（她）所负责的部分进行测试。当测试完成后，又需要进行集成测试（即几个模块组装在一起之后的测试）。

单元测试是白盒测试，往往和最后的功能测试存在一定的差异。

目前，很多先进的做法（如微软）都采用每日构建和冒烟测试的方法，就是在每天程序员都需要提交自己的代码，并且构建一个版本进行测试，第二天把测试的结果反馈给开发者。

每日构建和冒烟测试能够很大程度上提高软件的开发效率，并且对与 SQA 而言是增加了软件度量的指标。

每日构建和冒烟测试必然要建立在自动测试工具的基础上，依靠人是无法在每天晚上完成一次完整的功能测试的。

2.3.5. 版本升级测试

新软件开发完毕，即将发布的时候，用户非常关心：新的版本是否能够完成原来来版本的功能、是否和老版本功能兼容。

重新测试一边老版本的所有功能是必要的，会提前发现版本兼容的问题、数据的问题等等。

实现这个测试的基础就是自动测试功能，基于 TestCenter 的测试用例能够在很短的时间之内完成一次测试，防止问题发生。

2.4. 产品设计目标

测试需求管理

属性需求管理、需求的导入导出；

需求关联与链接；

场景；

需求正文管理；需求过滤器；

需求的团队管理；

需求的版本管理；

需求评审；

2.4.1. 测试构建

测试用例管理：树形测试用例组织；测试用例属性。

测试集：测试集的构建、查询；

测试数据字典；

数据池；

文档管理；

2.4.2. 执行测试用例

支持测试用例的自动执行与手工执行；

支持按照流程手工执行测试用例和在测试集中执行测试用例；

支持连接不同的自动测试工具，如 AutoRunner、QuickTestPro 等。

支持预约执行。

2.4.3. 测试报告

根据测试日志自动生成测试报告；

支持测试报告模版；

2.4.4. 缺陷管理

基于工作流的缺陷管理流程；

支持自定义过滤器；

支持缺陷检索；

支持缺陷合并；

支持丰富的缺陷报告；

2.4.5. 自动提示与公告管理

支持对任务的自动提示；

支持对缺陷变更的自动提示；

支持公告发布；

支持公告查看；

2.4.6. 用户自定义

自定义字段管理；

缺陷流程自定义；

2.4.7. 项目管理与用户管理

支持任务管理；

支持我的任务；

支持多项目管理；

支持用户管理和角色管理；

支持用户与项目关联；

2.4.8. 权限管理

支持角色的自定义；

支持角色的权限管理；

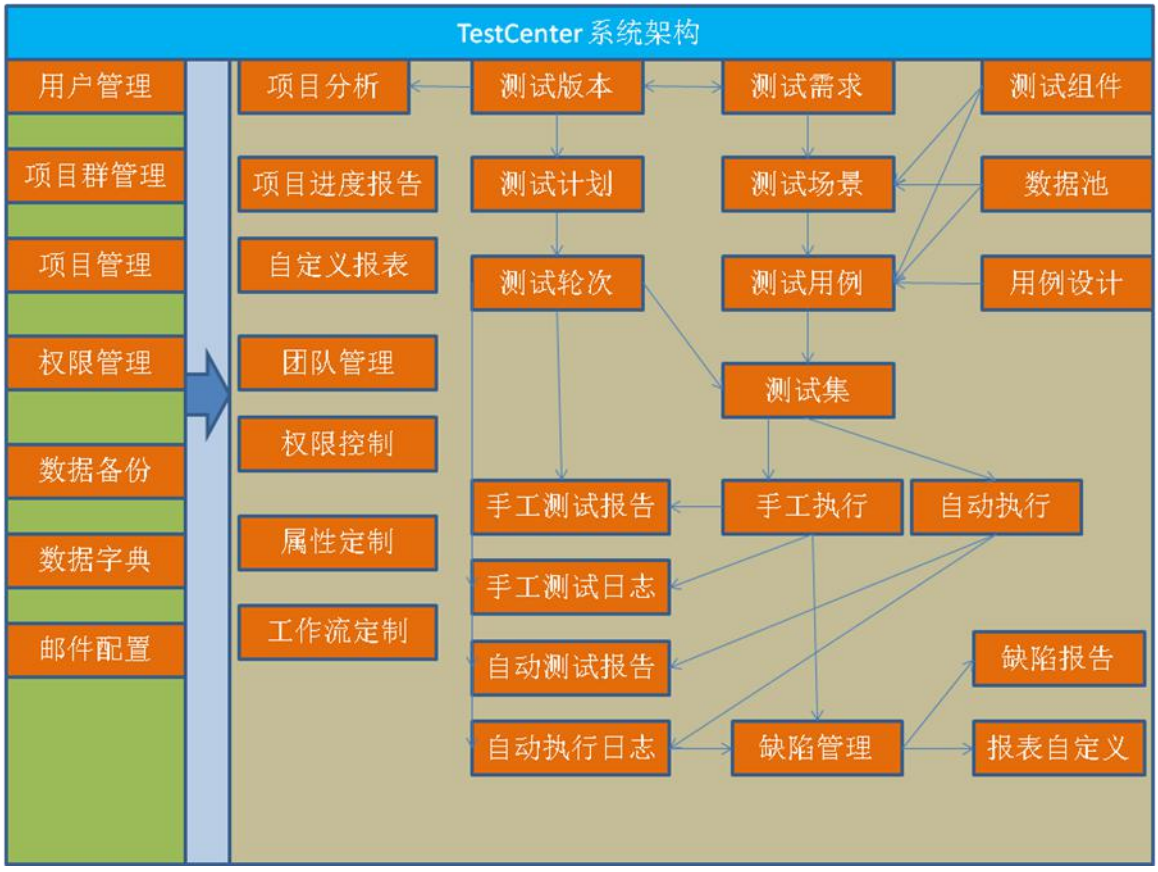
对相同对象的不同权限（读写权限）；

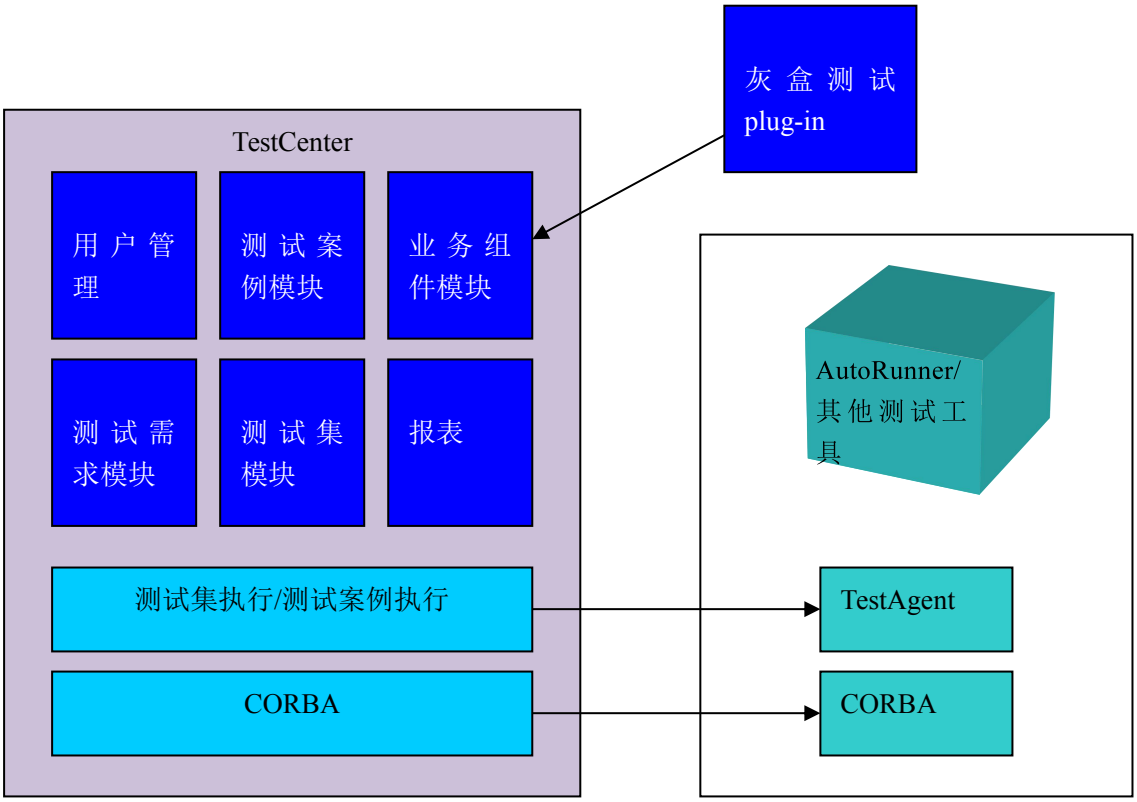
3. 系统体系结构特性要求

3.1. 系统架构

3.1.1. 系统构造

如下图所示，为 TestCenter 体系结构：



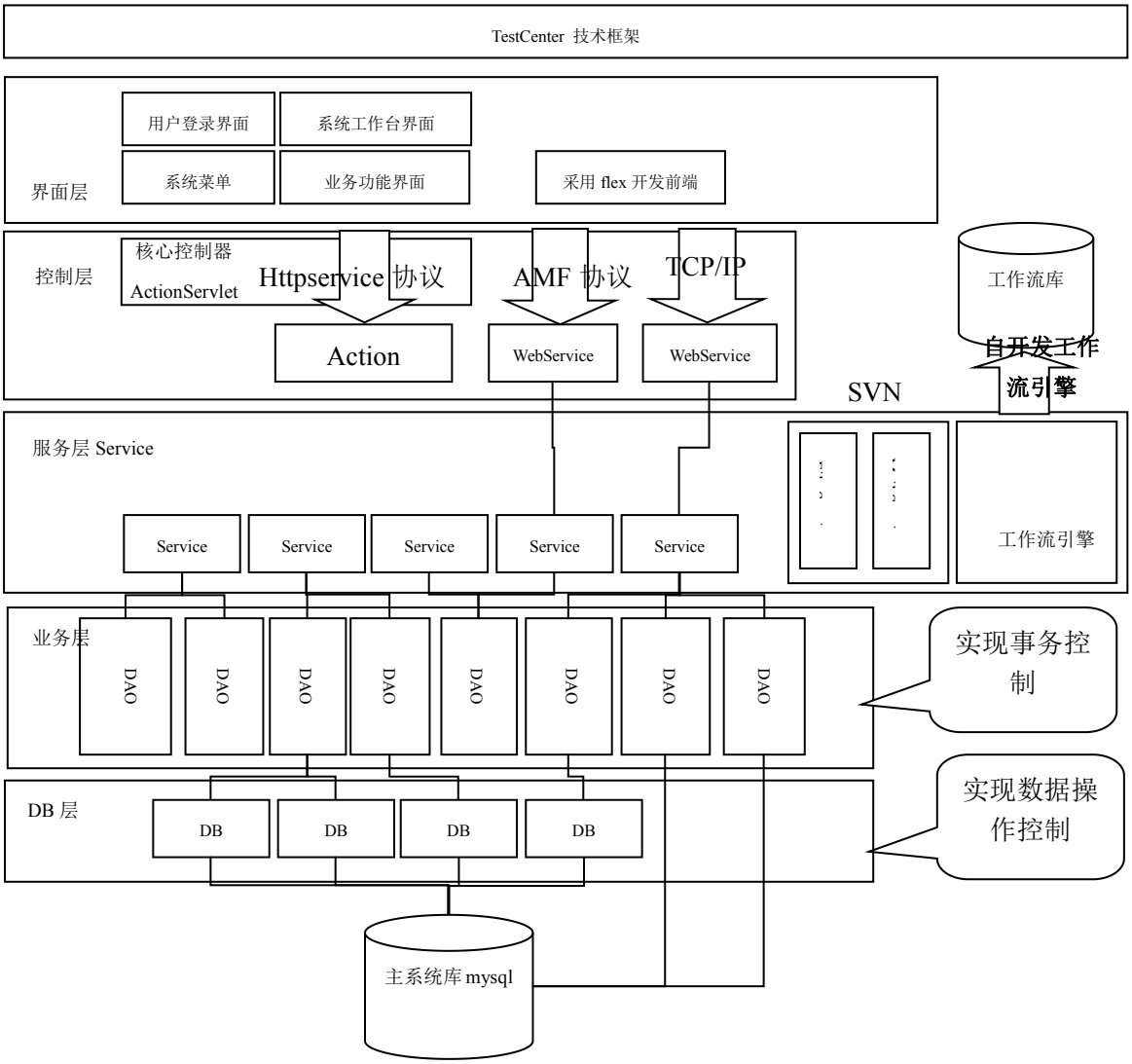


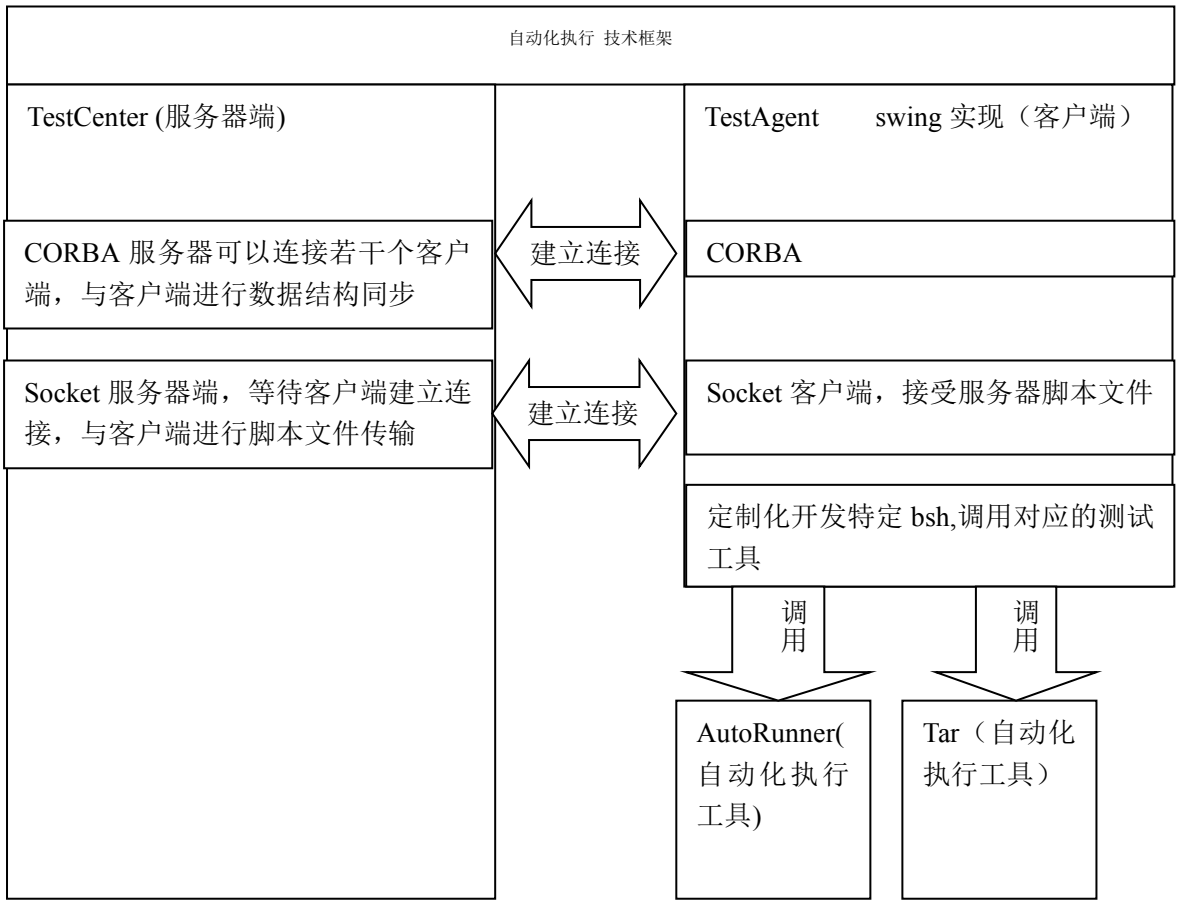
TestCenter

说明：

1. TestCenter 是基于纯 java 的软件。
2. TestCenter 采用 B/S 的体系结构，客户端只要有 JRE 就可以使用。
3. TestCenter 的执行基于 CORBA 的技术来实现，具有良好的执行效率。
4. 在执行客户端，需要安装测试工具，如 AutoRunner。如果测试工具和 TestCenter 连接，就需要安装 TestAgent。
5. 对于 TestCenter 来说，一个 TestAgent 就提供了一个可以执行测试脚本的 WorLine。
6. TestCenter 为了支持快速测试，跟上开发的节奏，支持灰盒测试插件，需要用户来编写一个 BeanShell 脚本就可以实现灰盒测试。

3.1.2. 系统技术架构


TestCenter 技术框架
界面层
用户登录界面
系统工作台界面
系统菜单
业务功能界面
采用 flex 开发前端
控制层
核心控制器
ActionServlet
Httpservice 协议
AMF 协议
TCP/IP
Action
WebService
WebService
工作流库
自开发工作流引擎
SVN
服务层 Service
Service
Service
Service
Service
Service
工作流引擎
业务层
DAO
DAO
DAO
DAO
DAO
DAO
DAO
DAO
实现事务控制
DB 层
DB
DB
DB
DB
实现数据操作控制
主系统库 mysql



3.1.3. 表现层（Flex+JSP+AS3.0）

表现层可采用 Flex 实现富客户端的展现方式，同时使用 Flex 异步提交方式实现请求无刷新，优化前端客户端体验，并且结合 CSS 实现 PORTAL 风格的主应用画面，使得测试用户可以在主应用画面上集中了解测试项目概况以及处理测试任务，而无需每次都进入到具体功能页面去工作。通过 flex 可以很方便的实现可定制的集中作业平台（DESKTOP），每个测试用户可以根据自己的角色以及不同的工作侧重点配置自己的 DESKTOP，这样系统登录成功后立刻就可以展开工作。

3.1.4. 控制层（STRUTS1.x）

采用 STRUTS1.x 实现业务请求分发和控制。STRUTS1.x 是非常成熟的控制层开源框架，它通过 ActionServlet 主控制器和各模块 Action 组件相互配合实现请求转发，并且通过 XML 配置文件来定义和集成控制组件，成功实现控制层的解

耦合，提高系统的灵活度和可扩展性，系统添加新业务时只需在架构中配置新的Action。

3.1.5. 业务层（WebService）

采用 Service 层统一对外提供业务接口，内部接口直接通过 ServiceFactory 获取 XXXServiceImpl 实现类的方法实现；外部接口可将 XXXServiceImpl 包装成 WebService 接口，并通过 WSDL 发布；HTTP 接口通过 Action 调用 XXXServiceImpl 实现。然后通过 Service 层调用 DAO 层完成完整的业务需求。这样的好处是很容易实现接口的复用，有利于接口的维护，开发人员只需修改一处即能实现对多个调用方的修改。在实现 CNAS 库文件管理时，可在 Service 层添加对 SVNKIT 的调用来实现。

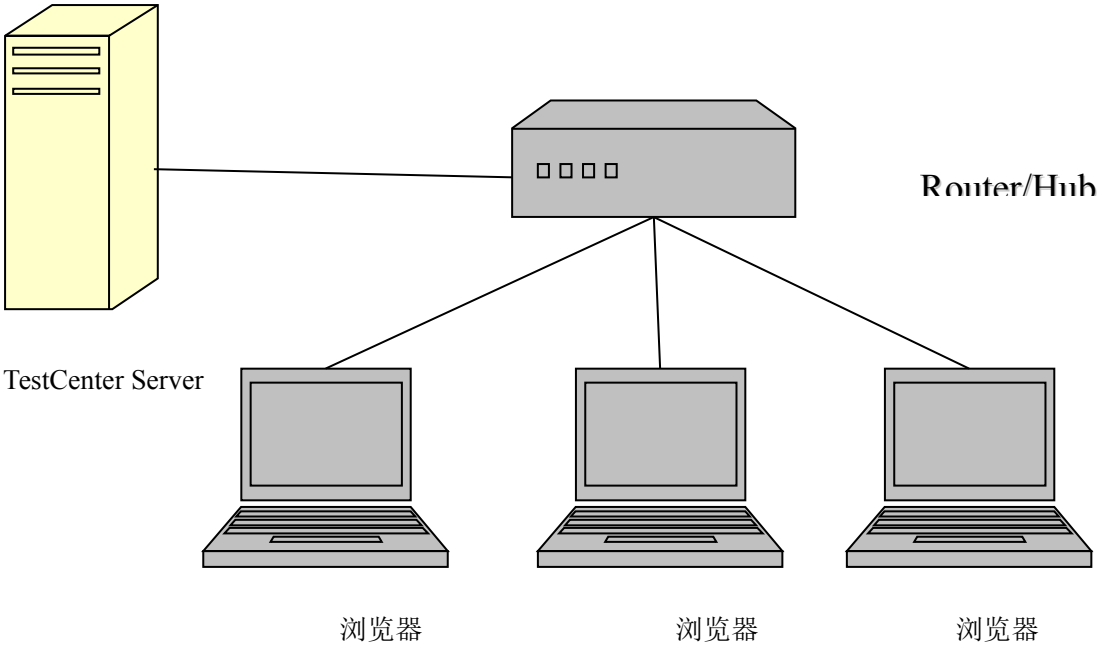
3.1.6. 数据访问对象（DAO）层（Hibernate）

DAO 层(Data Access Object)用来封装具体的业务逻辑，这一层的实现比较具有灵活性，可以采用 JDBC，或者采用 Hibernate 等 ORM 框架。

3.1.7. 数据库系统（mysql）

Mysql 是经过实践，能够承受较大数据压力的关系数据库，能够满足测试数据的快捷，安全，完整性读取。

3.1.8. 拓扑结构和运行环境



说明：

1.TestCenter Server 安装在 windows 系统上，使用 MYSQL 数据库。

2.TestCenter 支持 B/S 结构，客户端使用浏览器来访问服务器，就可以使用。

3.TestCenter 的数据存放在 MYSQL 上。

4.TestCenter 执行的时候，需要客户端起动 TestAgent，才能够允许一个执行的 Workline 运行在本地的系统上。

系统要求：

JDK1.6 以上

3.1.9. 技术架构的优点

1） 此架构已成功在测试管理系统二期项目中使用，实践证明该架构具有易于理解、层次分明、接口统一性能稳定等优点；

2） 架构中主要采取成熟的开源框架和协议（Flex、Struts1.x、Hibernate 等）实现 MVC 分层架构可以在保证系统稳定的同时保证系统的灵活性和扩展性，该架构通过 Service 层统一实现内部接口和外部接口，尽量多的实现了接口的复用，这样的设计有利于实现调用的统一性，易于维护；

3） 使用框架结合先进的 JAVA IDE 工具可以有效缩短开发周期，并且有助于保证模块的标准化；

4） 采用单一入口 BaseAction 和 Filter(过滤器)处理权限可以保证系统的安全性；

5） 采用 Flex 和 JQUERY 实现 PORTAL 界面提高系统可配置性和易操作性，也紧跟目前 B/S 的开发风向，具有技术的先进性。

3.2. 系统性能

TestCenter 具有非常高的性能，并且对硬件系统没有很高的要求：

第一，TestCenter 的客户端基于浏览器，并且核心采用 APPLET 来实现，完成界面操作，对系统的性能没有很高的要求；

第二，TestCenter 服务器，自动安装 MySQL，对系统的内存和 CPU 有一定的要求。在用户不使用大量的 WorkLine 来并发执行测试用例的时候，TestCenter 对系统的开销要求很小——TestCenter 的客户端会自己来访问数据库，而不需要消耗 Server 的资源；在执行的时候，Server 也会把大量的工作交给在客户端上执行的 TestAgent 和 Workline 来运行，自己的压力不会很大。

3.3. 扩展能力

TestCenter 是测试管理工具，本身提供了完善的测试管理的功能。

测试是整个开发的一部分，因此，扩展功能应该包括与开发相关系统的接口。

- **配置管理工具支持**

测试用例、需求和代码一样，也具有版本。

TestCenter 对配置管理工具提供了接口，需要开发一组 plug-in 来和配置管理工具交互。

TestCenter 的配置项如下：

1、测试需求；

2、测试用例；

3、业务组件；

4、测试集；

TestCenter 的后期版本，将会在这些配置项下增加一个 check in 和 check out 来从配置库中更新。用户只需要编写 check in 的脚本和 check out 的脚本就可以完成此工作。

目前此本版本不支持配置管理工具。

- **需求导入**

测试需求被 TestCenter 来管理。

需求是开发项目来定义的，某些情况下需要把需求导入到 TestCenter 中。

因为 TestCenter 本身管理的是测试需求，不是需求，所以此功能要求不多。

- **第三方测试工具支持**

TestCenter 采用 TestAgent 来和测试工具连接。

TestCenter 提供了一个 BeanShell 的接口脚本，用户只要来实现这个 BeanShell 脚本，并且通过这个脚本来驱动第三方的测试工具，就可以完成对他们的接口。

- **测试非 Windows 应用**

由于 TestCenter 采用纯粹的 java 来编写，可以执行在任何平台上，包括 linux 和 unix 等。

因此，TestCenter 可以和第三方测试工具整合，测试运行在其他系统上的应用系统。

3.4. 可靠性和可用性

1. 基于 B/S 结构

2. 基于 CORBA 和 JAVA

3.5. 国际支持

支持多种语言 Unicode 编码形式。

用户可以选择中英文界面的版本。

所有的界面提示使用了参数文件的方式，如果需要使用不同的文件，只需要修改参数文件就可以完成。

系统对语言编码的识别是由系统自动完成，用户不必考虑选码的问题。

4. 系统功能

4.1. 测试项目管理

TestCenter 支持同时对多个应用进行测试。

每个应用系统测试，都可以建立一个项目。不同的项目之间，不允许测试用例、业务组件、执行用户等对象在项目之间共享。

测试项目提供了对用户的关联，测试项目具备一个访问列表，进入项目用户访问列表的用户才能够“看到”这个项目、访问这个项目的资源。

4.2. 用户管理

TestCenter 提供了用户管理，只有有权限的用户才能够登录。

TestCenter 的用户种类有：

第一，测试用例设计人员。一般是业务人员，简称为 BA。使用 TestCenter 设计测试用例、业务组件。

第二，自动化测试工程师。使用 TestCenter 来创建业务组件、维护测试用例、创建测试集、执行测试集、查看测试日志。

第三，测试经理。具有所有的权限。并且能够创建报表、维护测试主题等。

第四，系统管理员，用来创建用户、删除用户、修改用户密码等。

第五，访问者。只能够查看，但是不允许修改任何内容。

4.3. 用户权限管理

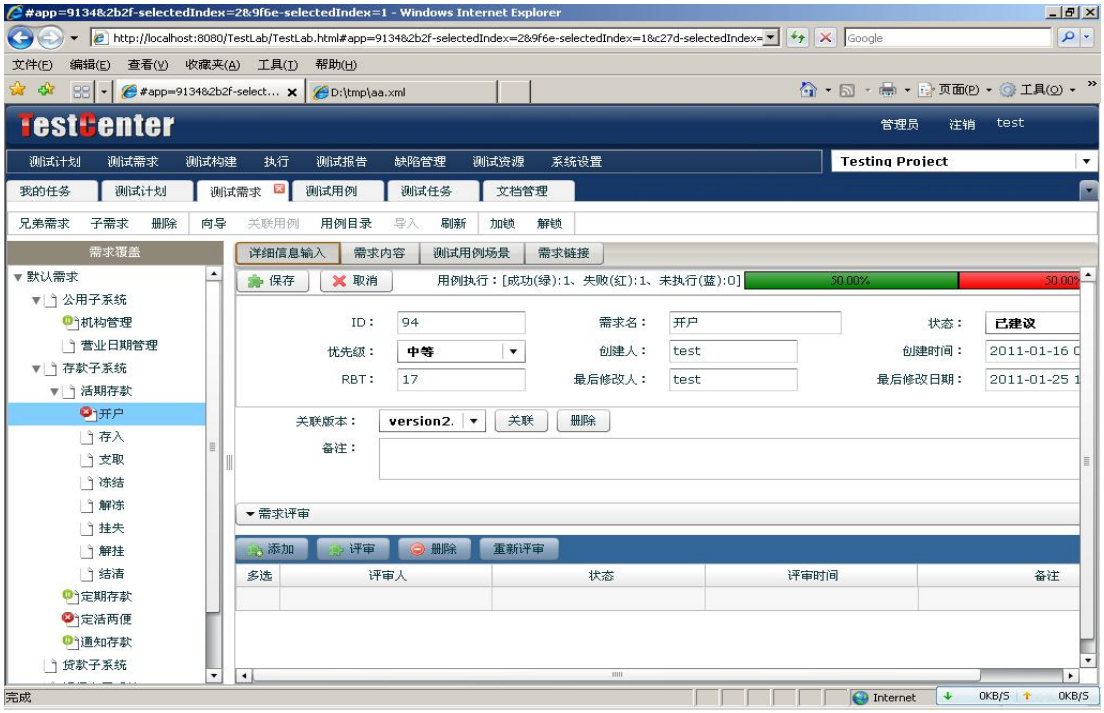
TestCenter 的用户权限包括：1）完全控制；2）访问（不可以修改）。

TestCenter 的 admin 用户作为管理员，能够对每个角色进行权限配置，使得各个角色使用不同的权限。

权限包括：测试管理、测试用例审核、缺陷、报表等。

4.4. 测试需求管理

测试需求包括了需求描述、测试主题、包括的测试用例、测试用例执行情况。



如上图，为 TestCenter 的需求模块：

测试需求是一棵树，可以建立多层次的关系。

可以增加、修改、删除、查询测试需求。

用户可以从需求树来创建测试用例目录，以便于管理。

用户可以从需求树的某个节点出发来创建测试集合，对应于具体的需求。

TestCenter 可以显示需求被创建、更改的历史记录。

如下图，为测试用例的基本属性：

4.5. 业务组件

业务组件，包括业务组件脚本、业务组件数据模版、业务组件资源等信息。

业务组件反映了一个用户交互流程。TestCenter 把业务组件作为测试系统的一个输入来看待。

由于业务组件对应的测试脚本和资源文件，和测试工具本身紧密相关，因此对不同的测试工具，都需要给这个测试项目定义一套筛选器。筛选器在用户使用 TestCenter 的时候自动被执行。

在 TestCenter 中，筛选器是一组 BeanShell 脚本。

4.6. 测试用例

测试用例是整个 TestCenter 的核心。

众多的测试用例组成一棵测试用例树。测试用例树支持增加、修改、删除、查询等操作，并且用户可以从测试需求直接链接到测试用例。

测试用例包括：输入参数、输出参数、数值传递、组成测试用例的业务组件实例以及实例之间的依赖关系（先决条件等）。

TestCenter 的测试用例支持手工测试用例和自动测试用例。手工测试用例包括了测试用例的前置条件、测试用例的步骤和预期结果等信息，用户用户的手工

执行；自动测试用例配置的方式，编写测试用例不需要编程的背景和知识，通过组件化的配置和数据修改，就可以轻松的创建测试用例。

测试用例支持复制和粘贴功能，能够很容易的创建大量测试用例。

TestCenter 的测试用例支持“审核”状态，通过审核之后的测试用例不能被修改、不能删除；如果需要修改，只有解除了“审核”状态，才能修改、删除。

4.7. 测试集

测试集是根据测试要求定义，并且执行大批量的测试用例的集合。

测试集内部的测试用例之间允许定义依赖关系。

测试集也可以把其他的测试集作为自己的一个子集。

对于测试集中所有的测试用例需要的输入参数，会自动生成一个.csv 文件作为输入参数的读入参数模版。

在测试集执行的时候，需要根据这个模版来输入数据——使用一个.csv 文件作为输入，并且在执行过程中自动对这些案例的输入参数赋值。

4.8. 启动自动测试执行

测试执行的对象是测试集。

启动测试集，TestCenter 会要求用户选择一个 数据场景文件，作为测试集执行时刻各个不同测试用例的输入参数。

在测试集执行过程中，TestCenter 会显示各个测试集和测试集内部的业务组件实例执行的状态：等待执行、执行失败、执行通过、暂停。

TestCenter 的自动执行，在启动的时候，支持用户选择可以执行在哪些设备上，也就是说用户可以选择几台机器来作为执行环境。

TestCenter 的自动执行，是支持可伸缩的方式来执行的，能够大大缩短执行的时间。

4.9. 手工执行

启动手工执行

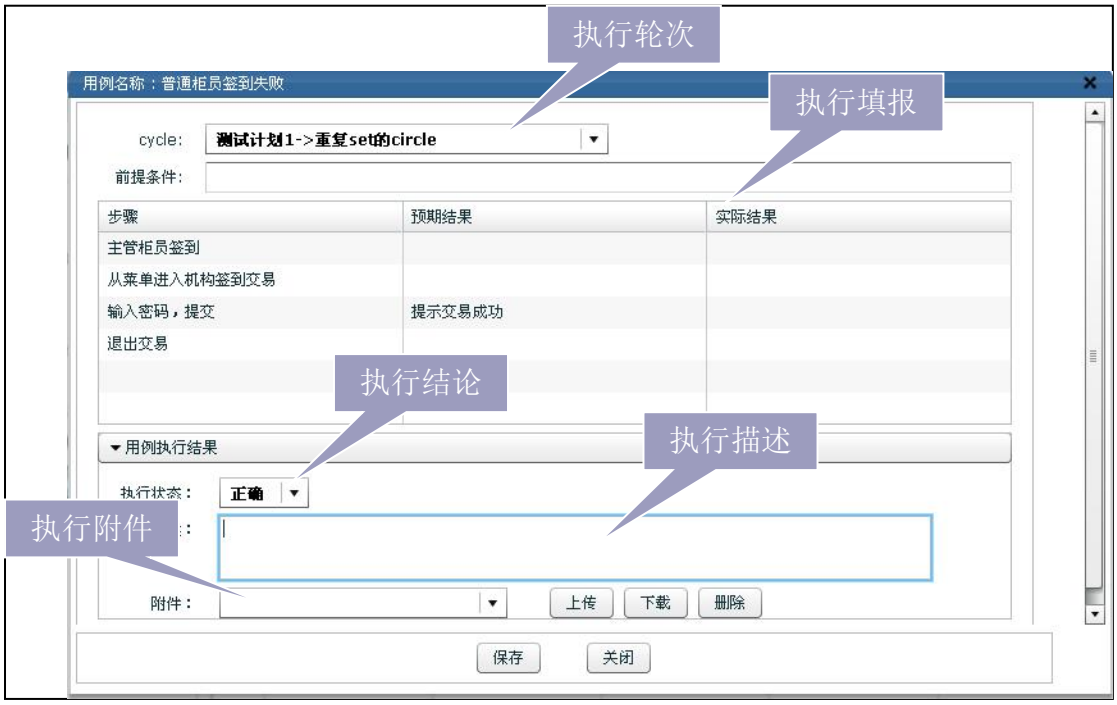
TestCenter 支持从测试计划来启动一个手工执行，并且要求用户在启动的时候输入一个测试执行编号，这个编号作为统计分析、测试日志的索引。

测试执行任务管理

在启动之后，TestCenter 的“手工执行”模块就可以把这次执行中的每个测试用例作为一个任务来进行管理：第一，需要进行任务分配，由测试经理把任务分配给具体的测试工程师；第二，测试工程师查询分配到自己名下的测试任务，然后来执行；第三，测试工程师来填报测试结果，如果是失败就可以直接来提交缺陷。

如下图，为在测试集中执行测试用例。

执行用例：



实时显示用例执行情况（需求模块）：

4.10. 测试执行监控和管理

在测试集执行过程中，TestCenter 允许测试人员“暂停”测试执行，这时候，已经分配、并且正在执行的业务组件会继续执行，并且在执行完毕后暂停，其他没有分配的 WorkLine 会暂停。

暂停的操作便于用户停止来临时查看问题。

4.11. 报表和测试报告

测试执行报告

测试集运行之后，会得到一个执行报告。包括：

第一， 测试日志执行的时间，执行人等；

第二， 测试对需求的覆盖率；

第三， 测试对组件，案例的覆盖率。

通过查看测试执行的日志，测试人员就可以明确测试用例执行的情况，如果出现问题就可以明确问题的现场情况。

测试报表

测试集执行完成之后，测试人员最关心的是：1）测试对需求的覆盖率；2）测试对组件，案例的覆盖率；3）测试用例执行结果的百分比；4）测试用例成功的个数与历史数据的比对关系。

TestCenter 将会提供一个报表，主要包括：

1、测试执行报告。这个报表上会显示测试用例执行的结果：测试通过、失败、未执行的百分比；还包括测试对需求的覆盖率。

2、测试执行日志。也就是整个测试集包括的业务组件，最终执行的比例。业务组件执行的比例，是衡量业务流程是否覆盖到一定比例的关键数据之一；

3、对比报告。测试用例执行结果与历史数据比对关系。根据比对关系，会发现测试用例执行的趋势。

自定义报表支持

TestCenter 的测试报表整合了 ireport 的功能，用户可以通过使用 ireport 的报表设计工具来自己设计测试报表，然后导入到 TestCenter 中，这样用户就可以实现自定义报表。

4.12. 缺陷管理

TestCenter 包括了完善的缺陷管理功能，并且和测试管理整合在一起：

✓ 简洁方便的操作界面

TestCenter 的缺陷管理使用非常简便，能够非常容易的让用户查看到自己

关注的缺陷、需要自己处理的缺陷、团队的缺陷。

TestCenter 提供了缺陷管理的主页面，能够实现这些功能，并且提供二次查询的功能，便于用户使用最少的时间发现自己所需要的缺陷。

✓ 完善的功能

缺陷管理提供了对筛选器，通过自定义筛选器，能够实现进一步的缺陷过滤。

提交缺陷模块非常容易使用，并且功能强大，支持通过 title 描述缺陷、备注缺陷等功能。

提供了通过缺陷编号快速定位缺陷的功能。

支持通过改变当前所选择的项目来查看不同项目中的缺陷。

✓ 自定义缺陷管理流程

TestCenter 的缺陷管理使用了工作流的技术，通过修改和订制工作流配置文件，能够非常容易的自己来定义缺陷管理流程，支持复杂流程和简单流程。

✓ 备注

为了跟踪缺陷管理的过程，查看缺陷处理的状态、人员、处理情况，TestCenter 支持缺陷管理备注。

每个参与缺陷处理的人员，都能够编写自己的额外备注，描述处理的情况，同时系统自动显示处理的时间。通过备注能够非常容易的查看缺陷流转的情况，找出瓶颈，发现问题。

✓ 与测试用例关联

缺陷能够直接通过链接到测试用例，并且显示测试用例的情况。

缺陷与测试用例紧密关联在一起。

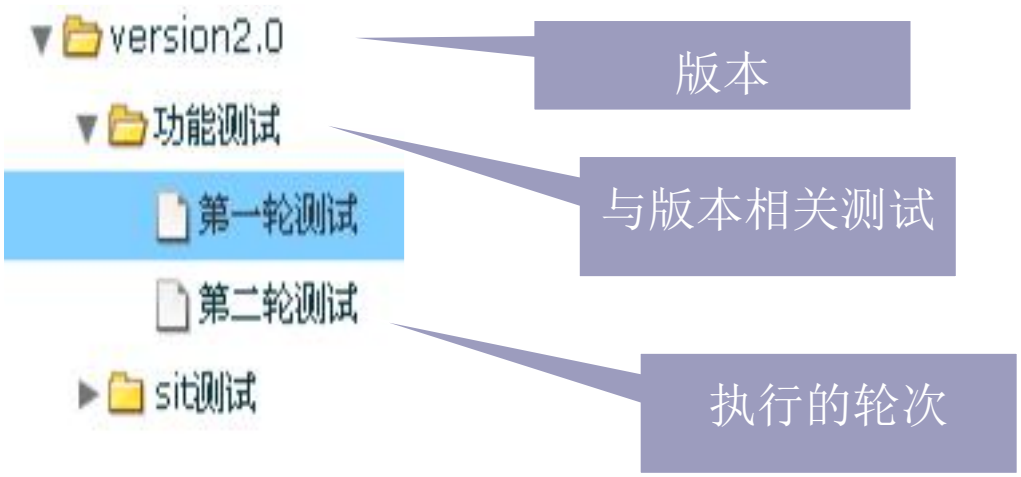
如下图，支持自定义的缺陷报告：

5. TestCenter 的特点

相对于 TestCenter Version4.x 版，TestCenter Version5 提供了更多新功能：

5.1. 测试计划模块

对测试计划提供了强大的支持：



如上图：

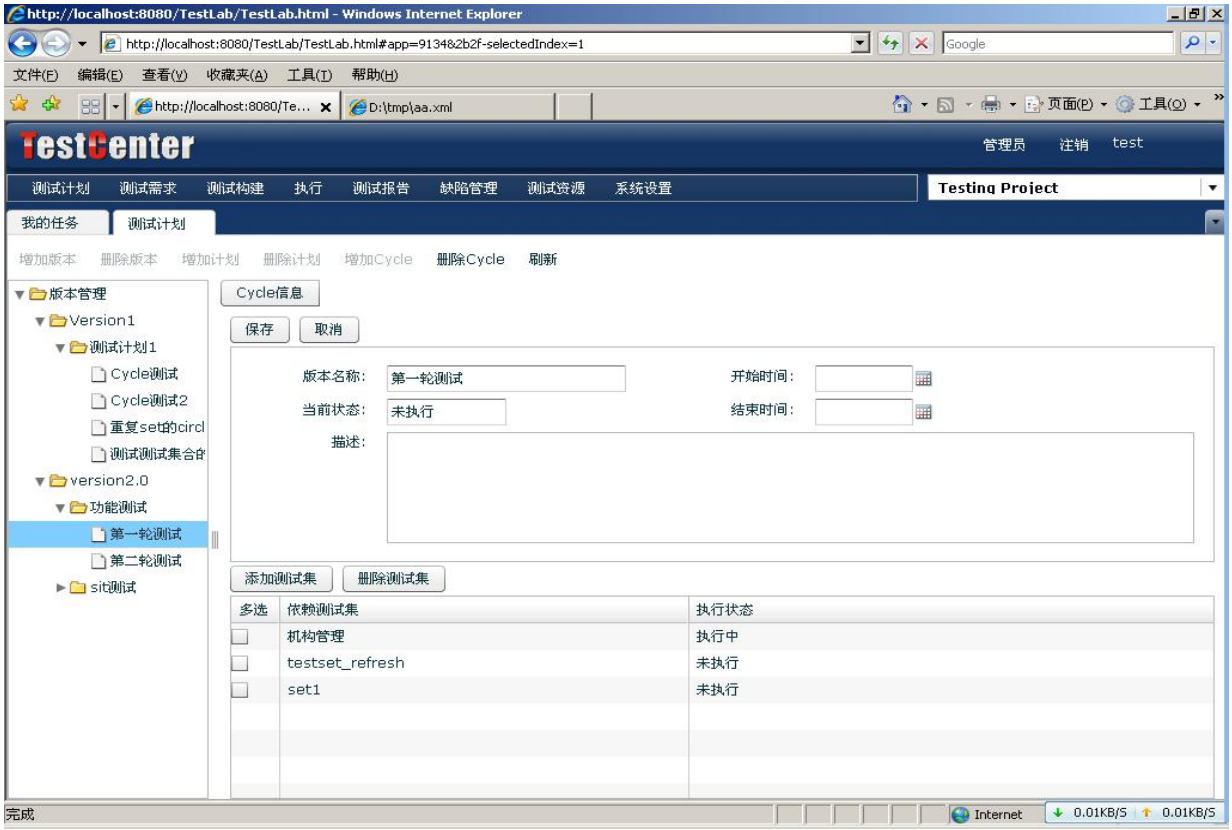
1、测试管理中纳入了版本的信息，是面向每个版本进行测试；

2、每个版本具有多个测试计划；

3、每个测试计划可以包括多个轮次；

5.2. 测试轮次

如下图，增加了对测试轮次的管理：



测试轮次包括测试集（一个或者多个），测试如果要执行，必须被包含在测试轮次中。

测试轮次：

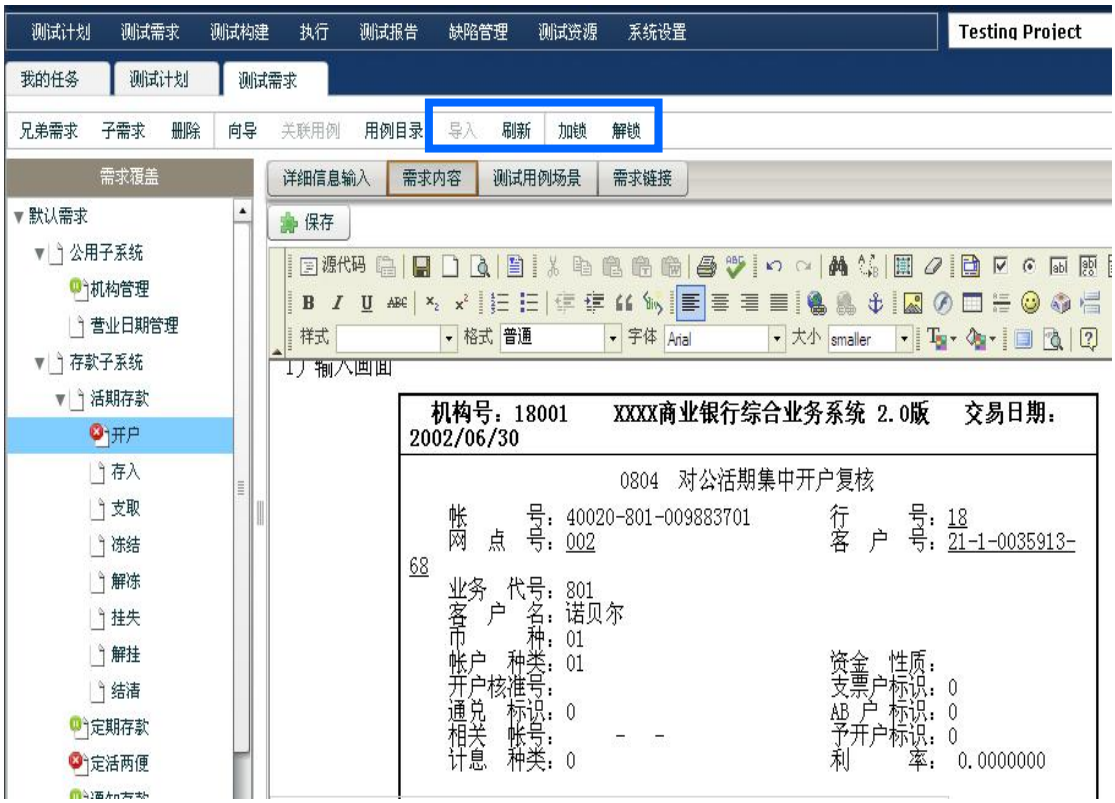
如上图，测试轮次包括了描述和详细信息。

5.3. 面向需求集合的版本

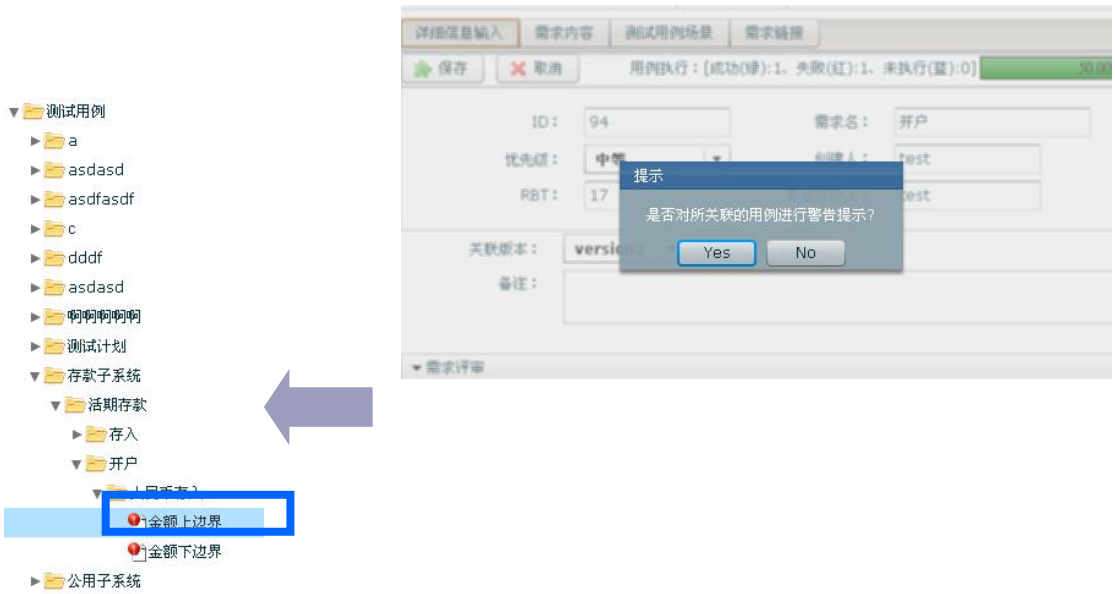
5.4. 面向需求项的需求管理

需求条目化：所有的需求变成需求项；需求项以需求树的形式体现；

需求支持团队管理模式（团队互斥）：



5.5. 需求跟踪矩阵



如上图：当一个需求被修改，会自动提示与它相关联的测试用例，并且对测试用例进行提示。

测试工程师根据提示来维护测试用例。

5.6. 测试任务管理

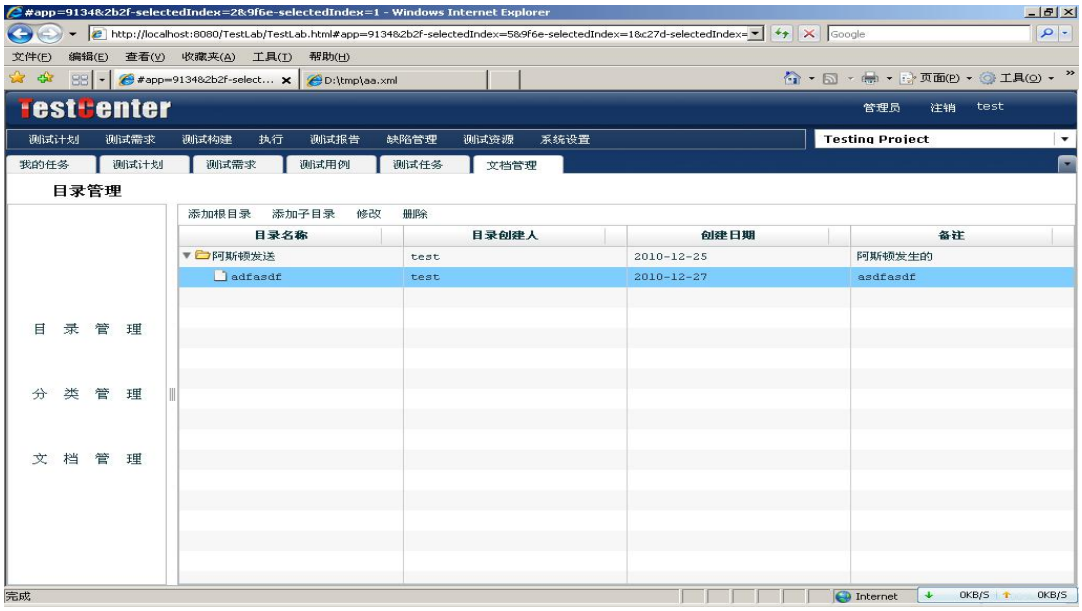
如下图：

项目经理可以在项目中创建任务，并且分配给各个相关人员；

测试工程师登录系统之后，在“我的任务”模块可以查看与自己相关的任务，并且可以根据实际完成情况标注。

5.7. 文档管理

如下图：



支持对测试中的文档进行管理。

文档支持版本管理。

6. 系统创新

6.1. 可伸缩的自动化测试执行框架

TestCenter 本身包含一个可伸缩的自动化测试执行框架，在服务器端存储了自动化执行的脚本及测试数据。由客户端发起请求需要执行的脚本及数据进行自动化执行。

主要有以下特点：

1. 支持脚本共享，所有执行脚本数据都存储在服务器端。

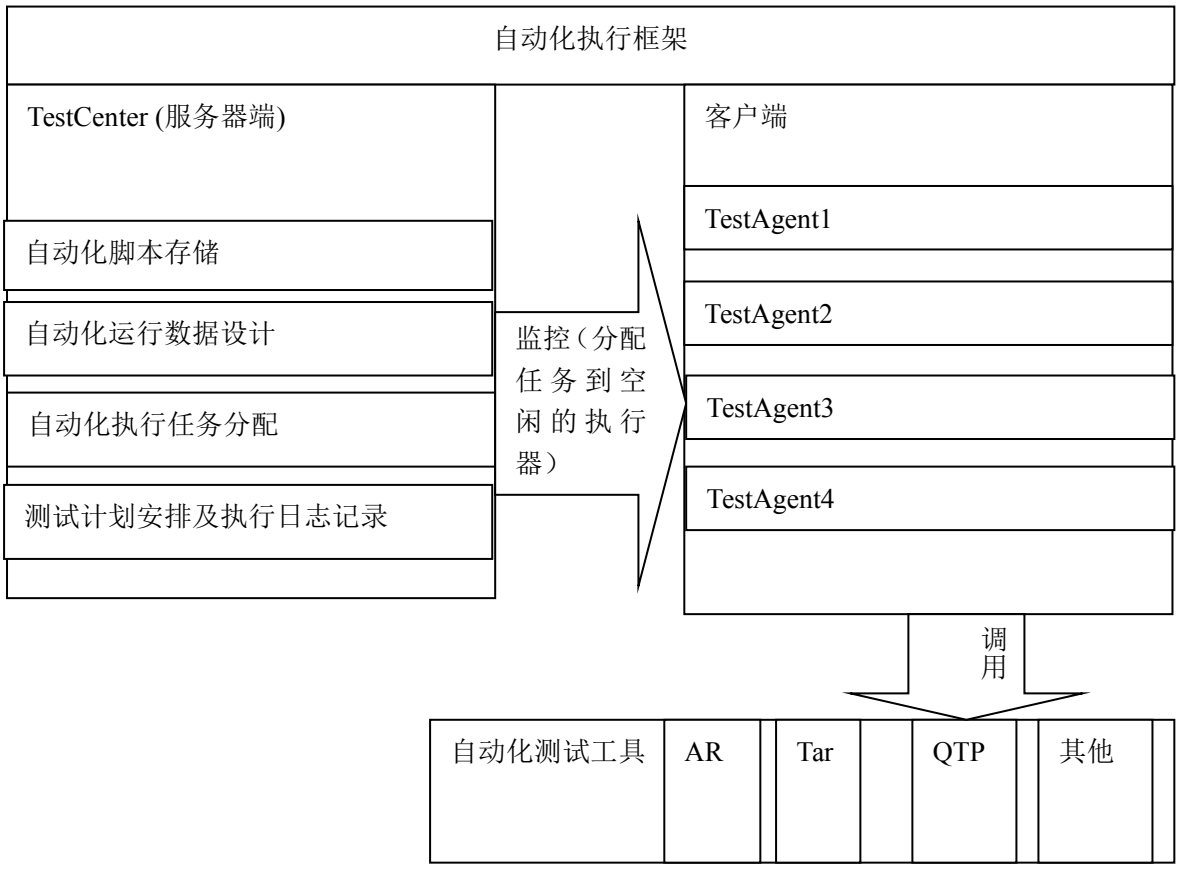
2. 支持自动化数据设计，可集合用例设计工具，进行自动化执行数据的设计，增强自动化执行的覆盖率。

3. 支持任务组成的灵活性，可通过不同的组件组合去执行不同的流程

4. 支持回归的并发性，可通过一个客户端，控制多个客户端去同时进行回归测试。服务器会自动监测哪个客户端空闲，给安排执行任务。

5. 支持第三方自动化执行工具，通过配置可集成 qtp 等工具

6. 支持对回归过程的监控（录屏，日志）。并对回归进行统计，生成测试报告。



6.2. 模型驱动自动用例设计

1. 概述

模型驱动自动用例设计是 TestCenter 独特的功能特性，可用于测试设计阶段，为测试设计人员提供需求到用例端到端的自动化解决方案。

模型驱动是指使用 UML 模型作为描述需求的工具，将**活动图**当做需求的载体，通过若干步骤生成有效和覆盖度高的测试用例。

通过此插件将测试设计的概念和方法论体系融入自动的设计过程，从而可以获得以下手工过程无法比拟的好处：

第一， 过程规范化，统一规范的入口和出口，适合团队化测试设计；

第二， 使用场景分析法自动寻找每一条可能的业务路径，设计出全路径覆盖的测试用例，避免测试遗漏；

第三， 使用正交分析法生成用例，列举出各种可能的数据组合，通过设置不同级别的数据（一般分为 Master 数据和 Slave 数据）确定一个正交范围，从

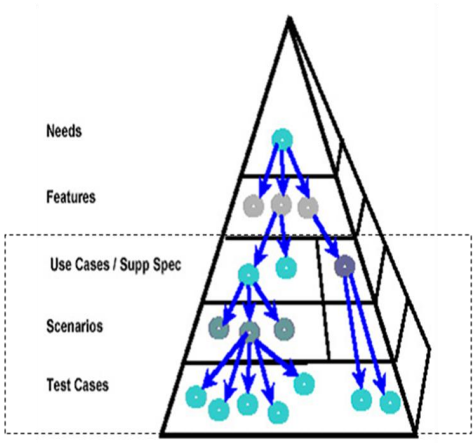
而避免用例集和过大变得难以执行；

第四， 可以通过配置实现等价类、边界值，通过分组概念实现业务规则的引入，从而提高用例的覆盖，并保证用例的有效性；

第五， 通过引入 RBT（Requirement-based testing 基于需求的测试）概念，使用 RBT 权重标识用例的重要性，使用例的评估更科学；

第六， 使设计过程自动化，生成用例迅速，用例覆盖广。设计过程可保存，可反复生成，优化了用例的维护过程，有助于实现 Daily building（每日构建）和自动化测试迁移。

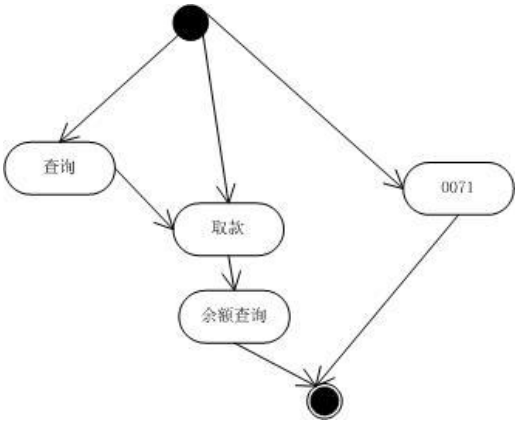
2.用例设计架构



如图，TestCenter 的用例设计模型是一个金字塔形的自顶向下逐级分解的过程。处于金字塔尖的是需求，最下层的是用例。产生的中间结果包括特性、功能规格、场景等。最后自动形成需求到用例的关联，使得设计可追溯。

3.UML 建模技术

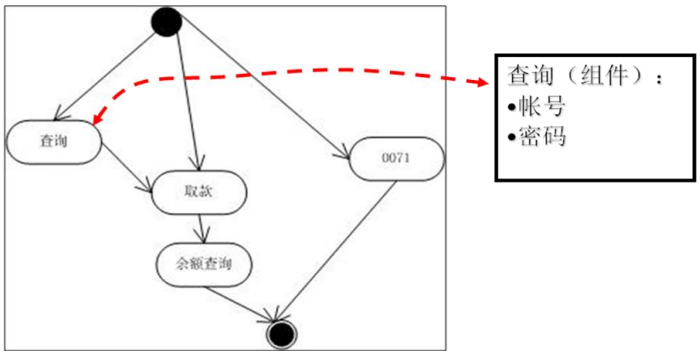
本模块在需求建模时采用 UML 中的活动图（Activity diagram）来描述需求说明书中的业务流程，因为活动图是 UML 中的一种动态模型，所以可以用来描述业务的流转方向，另外，活动图中也具有决策点可以用来支持分支流程和循环等基本业务逻辑。我们对活动图做了一步扩展，在活动图中增加了泳道来表示活动发生的范围，这样可以支持由众多子系统构成的复杂系统的描述。活动图必须从开始结点处开始，在结束结点处结束，所以使用有向图路径算法很容易得到从开始结点到结束结点之间有多少条路径。如下图所示是用活动图描述需求的一个例子：



4. 组件分离技术

在本模块中组件是业务流的组成单位，可以实现组件和流程的分离，即就是我们可以分别实现业务流程的建模和组件的设计。

组件可以是手工的也可以是自动的。手工组件主要用以描述一组业务字段，而自动组件还要加上可执行的自动测试脚本。

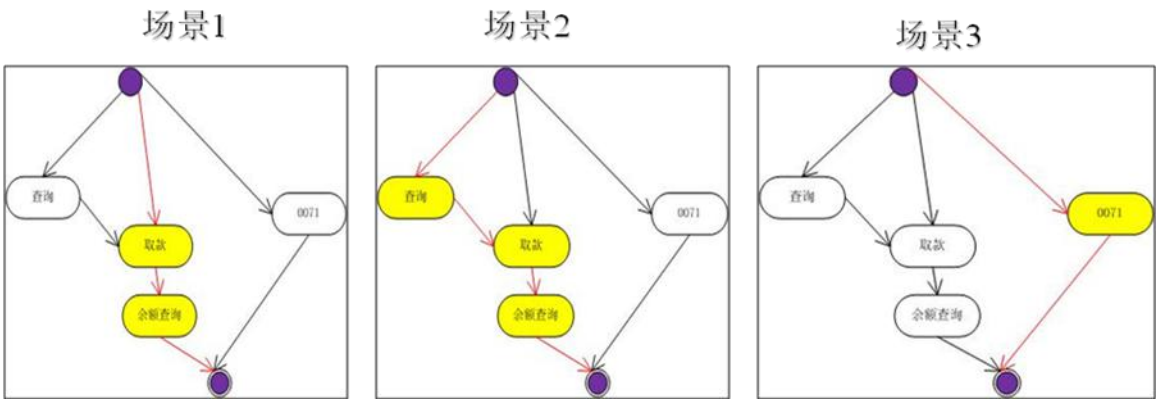


组件可以和活动图中的结点进行绑定，当活动图中所有活动结点都已绑定组件后，建模完成。一个完整的模型包括每个分支的操作步骤以及各个分支所包含的业务字段的集合。

本地组件：在本模块中本地组件式用户在建模过程中随即建立的组件，通常是一些比较特殊复用率不高的组件；

远程组件：远程组件是保存在 TestCenter 组件库中的组件，通常是一些公共组件，并且复用率较高，可以在 TestCenter 中预定义好这些组件然后在本模块建模阶段引用即可。

5. 业务场景分析



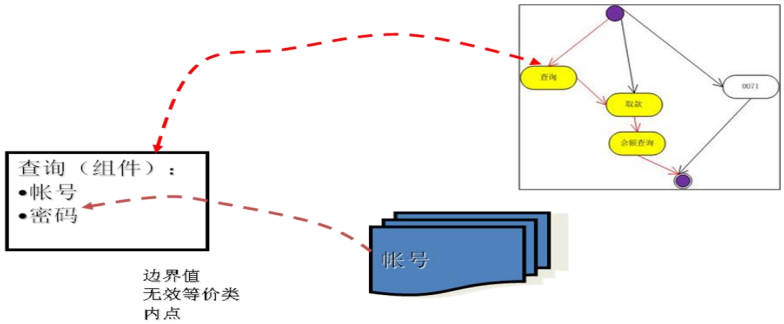
场景拆分的过程是自动的，上图展示了某金融产品的业务流程图，本模块自动拆分并找出了该业务中的三条分支路径（图中黄色节点）。

6. 测试数据管理

测试场景生成后需要准备测试数据。本模块采用两种方法实现测试数据的管理：直接录入数据；数据字典。

直接录入数据就是按照场景业务字段的顺序分别录入数据，录入完成后生成一个二维表，其中列是业务字段，行是测试数据。

数据字典是用来管理测试数据的有效方法，每个数据字典用来表示一个业务系统中的业务字段，并且包含了和业务字段关联的取值，使用数据字典来管理测试数据具有非常好的灵活性，便于查找和引用，缩短了设计的周期。



上图是管理测试数据的示意图，对于某个场景中的查询结点来说，具有账号和密码两个业务字段，需要测试设计人员在账号密码字段下录入多个账号数据，在录入数据时可以将数据标识为边界值、无效等价类和内点（即对应于边界值、无效值和有效值），这样就可以运用等价类划分、边界值、正反例等测试方法。

7. 笛卡尔积和 M-S 法则

笛卡尔（Descartes）乘积又叫直积。假设集合 A={a,b}，集合 B={0,1,2}，则两个集合的笛卡尔积为{(a,0),(a,1),(a,2),(b,0),(b,1),(b,2)}。可以扩展

到多个集合的情况。

将测试数据加工为测试用例时首先使用笛卡尔积获取一个集合 S，集合 S 是各个业务字段取值之间的所有组合。例如：

仍然讨论以上的那个例子，查询组件包含两个业务数据：账号、密码。经过了数据管理录入测试数据，得到了下面的二维表：

账号	密码
活期	正确
定期	错误
一账通卡	

经过笛卡尔积处理后得到的 S 有 6 个元素：

账号	密码
活期	正确
活期	错误
定期	正确
定期	错误
一账通卡	正确
一账通卡	错误

这只是一个非常简单的情形，业务组件只含有两个业务字段，并且测试数据量也不大。如果遇到几十个字段，并且测试数据很大的情况就会导致笛卡尔积计算时间很长，生成的集合很大等问题，这时候需要使用 M-S 法则进行处理。

M 是指 Master，Master 是所有业务字段中的关键字段，关键字段通常有以下特征：1）有特定的值域，或易于做等价类划分；2）不同的取值或等价类代表了不同的产品或将业务流程引到特定的分支上或者影响最终的结果；S 是指 Slave，Slave 是从属字段，指除了关键字段之外的不对业务流程和规则产生影响的字段。

标记为 M 的字段需要运用笛卡尔积以确保测试业务的完整性；另一方面，可以认为 S 字段是一个纬度，采用一组数据，还有一种方法是对 S 字段采用轮询的方式，保证 S 字段的值也可以被测一遍。

M-S 方法的核心在于在保证覆盖不受较大影响的前提下降低了正交法的数据纬度。

8.分组、合并

分组与合并是用来使生成的测试用例符合需求描述的业务规则的技术。

分组指的是测试数据的分组，通过分组将不同业务字段的测试数据之间建立

约束关系，可用以实现业务字段之间的依赖，例如实现：当甲字段取值为 A 时乙字段只能取值 2、3。

分组的过程就是将不同业务字段的值进行组合的过程，组合完成后形成一个不规则的矩阵，之所以说是不规则矩阵那是因为分组时参与分组的字段数是不确定的。下面举一个例子：

业务类别	期限
整存整取	3 个月
整存整取	6 个月
整存整取	1 年
活期	不输

上表定义了 4 个分组。

合并是指业务字段的合并，经过合并的业务字段在生成测试用例时取值相同，可以通过合并实现录入和上面某业务字段相同值的需求。

9.RBT

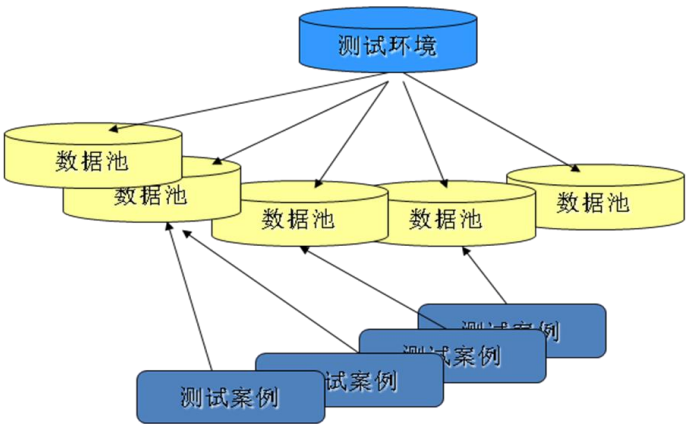
RBT（Requirement Based Test）是基于需求的测试，本模块是基于 RBT 实现的。并且本模块会为每一个生成的测试用例赋予一个业务风险指标——RBT 值，这个 RBT 值可以看做是对这个测试用例在业务需求中重要程度的衡量，本模块计算 RBT 的依据是根据这个案例的一些特征，例如关键业务字段（Master）的取值（边界值、内点、无效等价类）。

通过 RBT 能够更加客观的分析用例的覆盖率。

6.3. datapool 数据池

1.概述

TestCenter 数据池（Data pool）功能可以实现测试数据的全面管理，可以满足客户对各种测试数据的需求。本功能的基础架构如下：



2.功能特性

数据池功能主要具有以下特性：

第一， 用户可以在 TestCenter 中自由地创建数据池，以数据池为单位对数据进行管理；

第二， 用户可以根据业务需要定义数据池的格式，主要是业务字段的设置；

第三， 用户可以将测试用例（案例）的参数绑定到数据池的列，通过这个动作可以实现测试用例从数据池里取数据；

第四， 对数据池的引用分为两种情况：1）循环使用 2）一次性使用（此情况仅对资源数据，用完则需再造）；

第五， 用户可以选择不同的数据收集方式：1）手工录入 2）数据库抽取 3）Excel 外部文件导入；

第六， 用户可以将场景参数绑定数据池的列，一旦绑定则可以自动将场景下创建的用例的同一列与数据池的列做绑定，从而起到模板复用的作用；

第七， 数据池可同时用于手工和自动化测试过程，在自动化过程中数据池会进行数据检查，查看是否数据已经取完并及时通知用户。

7. 厂商支持能力

TestCenter 测试管理软件，我们通过在线 QQ、微信、电话、电子邮件为您提供支持与服务，您也可访问我们的网站 http://www.spasvo.com/寻求帮助；为保证服务质量，确保有效地解决用户的问题，保障用户的项目实施进度，技术支持仅向授权用户和授权试用用户提供。请您在联系泽众技术支持时，告知您的单位名称和服务代码。

技术支持

电话：021-60725088-8007

传真：021-60725088-8017

电子邮件：support@spasvo.com

QQ：1404189128

泽众微信公众号

产品服务

有关培训、产品购买及试用授权方法的问题，请与销售代表联系，或联系泽众咨询热线.

电话：021-60725088-8006

传真：021-60725088-8017

电子邮件: sales@spasvo.com

提供完备的用户手册，管理员使用手册，系统技术手册并再系统升级后及时修改更新服务。

厂商能够根据在实际应用中的问题，迅速给予解答（2 小时内），并给出解决方案(48 小时内)。