文章编号：2095-2163(2020)04-0001-05 中图分类号：TP311 文献标志码：A

建筑全性能仿真平台内核联合测试平台的设计与实现

郭 勇，苏小红，邱 景
（哈尔滨工业大学 计算机科学与技术学院，哈尔滨 150001）

摘 要：针对建筑全性能仿真平台的高度复杂性，且无法用已有测试工具进行测试的问题，提出了一种建筑全性能仿真平台内核功能正确性联合测试方法。首先每个模块实现者创建测试项目，根据需要，在项目中创建一个或多个测试用例，并设置精度要求；之后将被测代码从代码管理平台同步到测试平台，进行编译和语法检测；最后自动进行测试，并给出评测结果。测试者可根据给出的测试结果，调整算法，最终达到要求。实际应用表明，该系统解决了复杂的建筑全性能仿真平台内核模块的联合测试问题，能够高效快速的进行模块的联合测试，大大提高了的测试效率。

关键词：联合测试；仿真平台内核；DeST；建筑节能

Design and implementation of the kernel joint test platform for building full performance simulation platform

GUO Yong, SU Xiaohong, Qiu Jing
(School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China)

【Abstract】 Aiming at the high complexity of building full performance simulation platform and the problem that it cannot be tested with existing test tools, a joint test method for kernel functional correctness of building full performance simulation platform was proposed. First, each module implementer creates a test project, creates one or more test cases in the project according to needs, and sets the precision requirements. Then the tested code is synchronized from the code management platform to the test platform for compilation and syntax detection. Finally, the test is done automatically and the evaluation result is given. The tester can adjust the algorithm according to the test results, and finally meet the requirements. The practical application shows that the system can solve the problem of joint test of complex core modules of the full-performance simulation platform, and can carry out the joint test of modules efficiently and quickly, which greatly improves the test efficiency.

【Key words】 Joint test; Simulation platform kernel; DeST; Building energy efficiency

0 引 言

软件质量的好坏，决定了软件与需求相一致的程度，以及软件产品是否可用和生命周期的长短。需求调研不充分或开发人员的疏忽，将导致不合理的设计和软件缺陷，使软件存在潜在的危险。在一定条件下缺陷代码被执行，会导致软件运行失效，从而给软件使用者带来损失。频繁的软件故障会使软件不可用，且会带来极大危害。为了避免在开发时引入软件故障，软件测试应贯穿整个软件开发过程。通常，软件测试主要包括单元测试、集成测试和系统测试。单元是一个比较笼统的概念[1]，一个单元可以是一个方法、一个模块、一个类或是一个组件。单元测试主要是在初期，确定各个基本部分是否符合要求。单元通过测试，并不能保证单元与单元之间是否能协调工作。在通过单元测试的基础上，还要进行集成测试，以此确定相关联单元组装在一起是否能够正常工作。为了更好的实现松耦合，可采用基于构件技术的软件开发，每个构件也可看作一个单元，它把构成系统的各个部分从系统逻辑中清晰地分离出来。每个单元均采用统一的标准进行开发，然后组装在一起，构成整个系统。目前，许多软件都采用了基于构件的软件开发方法[2]。对于每个构件，用户并不需要了解内部细解，这样可以加速软件开发，但也给测试带来了困难。因此，要保证构件组成系统的正常工作，就要对基于构件开发的系统及时进行全面的回归测试。由于手工操作无法实现快速准确的回归测试，因此通常采用自动测试方法。

1 相关工作

自动测试框架通常由假设、概念以及为自动化

基金项目：本课题得到“十三五”国家重点研发计划课题“建筑全性能联合仿真平台内核开发”(2017YFC0702204)；国家自然科学基金项目(61672191)的资助。

作者简介：郭 勇(1967-)，男，博士，讲师，主要研究方向：软件测试；苏小红(1966-)，女，博士，教授，博士生导师，主要研究方向：智能软件工程、软件测试、计算机图形学；邱 景(1982-)，男，博士，讲师，主要研究方向：程序分析、软件测试。

收稿日期：2020-01-07

测试提供支持的实践集合组成[3]。自动测试框架的种类较多,如:基于数据驱动的软件自动化测试框架[4]、基于关键字的自动化测试框架[5]、基于用例的自动化测试框架驱动[6-7]的等等。然而,由于软件开发的方法和采用技术的多样性,任何一个自动化测试框架都不能用于所有类型的软件测试。因此,对于无法使用通用框架进行测试的系统,则需要定制开发一套适合被测系统的的测试工具。

建筑全性能仿真平台是为解决我国绿色建筑工作中,建筑热湿环境及机电系统性能仿真,模拟这一关键基础问题而开发的。项目目标是解决建筑环境控制多因素共同耦合的复杂问题,它是基于 DeST[8]软件外部扩展功能接口开发的,内核中采用了基于功能模型接口(FMI)、功能模型单元(FMU)及联合仿真(Co-Simulation)技术,由模型建立、模拟计算、节能评估等模块构成[9]。实现了建筑负荷与独立人行为功能模型单元的联合仿真,并判断出建筑节能设计方法的适用性。在建筑全性能仿真平台内核上,可以扩展多个功能模块。比如人行为、遮光采阳、系统及冷热源模拟等。由于采用了 FMI 标准,每个功能模块可以单独开发,独立功能模块开发完成后完成组装,再进行集成测试。由于整个平台内核由多个单位联合开发,给内核的联合测试带来不便。另外,每个模块的测试数据量都非常庞大,生成的结果数据可能会有几十万条,根本无法通过人工方式对结果数据进行误差分析[10]。

本文提出了一种建筑全性能仿真平台内核功能正确性联合在线测试方法,该方法可以直接对源代码进行测试,代码从管理平台(比如:Github)同步到测试端,并自动编译生成相应模块;测试人员可自行创建测试用例、自动保存测试用例、随时修改测试用例、指定测试通过标准;该工具提供相应的管理功能,测试人员及审核人员可通过软件进行通信。审核人员对测试结果审核评估,并及时反馈给测试人员。

2 系统需求

建筑全性能仿真平台内核联合测试平台是一个分布式测试平台,平台主要包括:用户管理、代码管理、测试项目管理、测试用例管理、代码编译及语法检测、模块组装、被测程序自动执行、结果判定及结果审核等功能。

用户管理:主要包括用户的新建、管理范围设置、删除用户、修改用户、查询用户及权限分配。

代码管理:用户可以通过该功能自动从代码开发管理平台同步代码到测试平台,并可随时更新被测代码。

测试项目管理:用户以项目为单位创建测试项目,每个项目中可以创建一个或多个测试用例。

测试用例管理:主要是对测试用例参数进行设置、测试标准文件进行上传更新、重命名。

代码编译及语法检测:用于对被测代码进行编译进行语法检查,生成统一标准的测试模块。完成测试前的模块与内核组装,建立通信联系。

被测程序自动执行:主要是自动执行被测程序,收集计算结果,自动实现计算结果的比对,并存入数据库中。

结果判定及审核:主要用于检查不符合计算精度的项,反馈给开发人员做为模块改进的依据。若计算结果符合要求,则审核通过,锁定该版本。

根据主述需求,系统结构如图 1 所示。

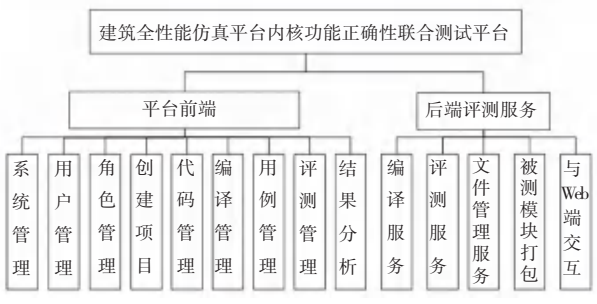

图 1 系统功能结构

Fig. 1 Functional structure of the system

3 系统设计及实现

软件设计结构有许多种,但分层结构是最为流行的软件设计方式。分层结构有效的降低了单个问题的规模和整个系统设计的复杂度。系统采用分层结构,具有良好的可扩展性、易于维护,上层使用下层的各种服务,每一层都对自己的上层隐藏其下层的细节,这样可更好的保证系统设计上的高内聚、松耦合。因此,本平台在设计上采用分层结构。

3.1 总体设计

系统采用分层结构,共分为五层:UI(User Interface)层、业务层、持久层、服务层和数据层。系统架构如图 2 所示。

UI 层是系统和用户之间进行交互的接口,实现信息的内部形式与用户接受形式之间的转换。本系统以 Web 方式显示相应的操作界面,为用户提供相应的操作,具体功能由业务层提供。

业务层是系统架构中体现核心价值的部分。业务规则的制定、流程的实现等,与业务需求有关的处理都在该层实现。业务逻辑层在体系架构中的位置

很关键，它处于持久层、服务层中间，起到了数据交换中承上启下的作用。

图 2　建筑全性能仿真平台内核联合测试平台架构

Fig. 2　Architecture of federated testing platform for building full performance simulation

持久层主要负责对用例数据、日志数据、帐户数据、结果数据的访问操作。

为了方便分布式处理，服务层将一部分功能以服务的形式提供给业务层。其中包括：编译服务、文件上传服务、文件解压缩服务和测试服务等。

数据层主要包括用来存贮信息的数据库。

3.2　核心流程设计

平台在设计上支持多用户同时在线使用，需要同时处理多个用户的并发请求。为此，系统采用了多任务面向服务的多进程处理方法。进程分为服务端进程队列和客户端进程。为了保证在线用户的服务质量，系统采用了限制进程数量的方法。为了提高处理效率系统起动时需创建一个进程队列，用于多任务处理，每个任务由一个进程处理。初始进程队列中的进程数是计算机内核的整数倍。当检测到用户请求时，从进程队列中取出一个进程，处理客户的请求，完成客户请求后，将进程重新放入队列中。客户-服务器进程的流程如图 3 所示。

当创建测试进程后，测试进程首先获取测试用例路径，然后读取测试程序，创建相应测试目录，启动测试。程序流程如图 4 所示。

3.3　代码同步设计

由于模块开发者分布在全国各地，通常是将代码放在托管平台进行管理，模块开发完成后，需要进行联合测试，这样就需要将代码放在联合测试平台。手动将代码拷贝到测试平台非常麻烦且不方便，而一般的工具又无法直接用到联合测试平台。为解决代码自动同步的问题。本文将这一环节设计为 ，由联合测试平台自动完成同步操作。

图 3　客户-服务器进程

Fig. 3　Client-server process

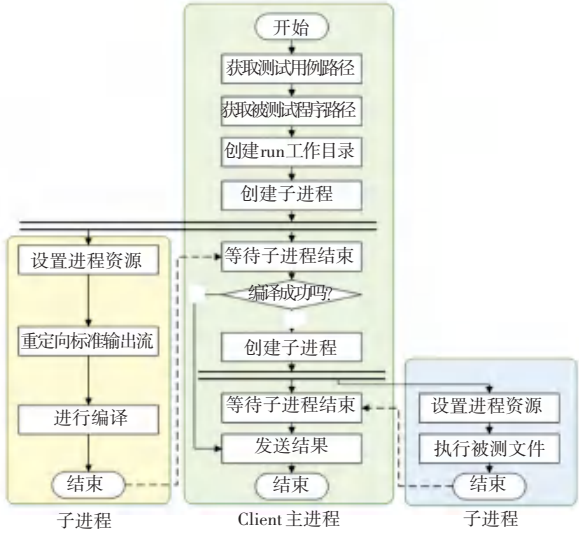

图 4　测试进程流程

Fig. 4　Test process flow

在测试人员创建测试项目时，通过参数设置的方式 ，将代码管理平台的信息保存在测试项目配置信息中(即：在参数设置中，可指定被测代码所在的服务器)。在代码管理或执行测试前，测试人员在进行代码同步时，系统会根据设置的代码服务器地址，从服务器端将代码同步到被测平台。为了代码安全，被测代码被保存在联合测试平台服务器端，测试人员不能接触到后端代码，这样确保了代码的安全。代码管理平台与测试平台的通信关系如图 5 所示。

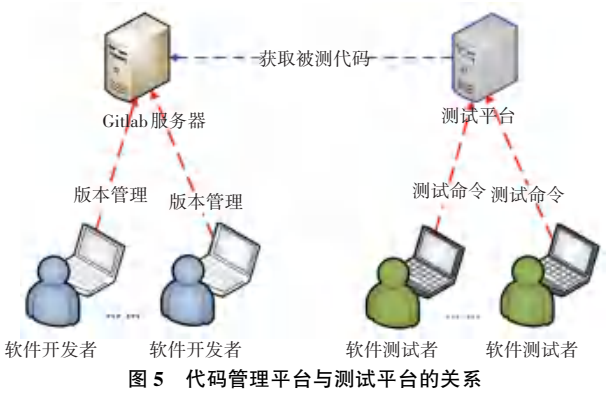

图 5 代码管理平台与测试平台的关系

Fig. 5 The relationship between code management platform and test platform

3.4 主要功能模块的设计及实现

3.4.1 系统管理模块的设计

系统管理主要功能是用户管理和角色管理,这些功能主要是管理人员使用。用户管理主要是用户的新建、管理范围设置、删除用户、修改用户、查询用户及权限分配。其中权限管理使用了角色及具体权限相结合的方法。权限分配界面如图 6 所示。

图 6 权限管理

Fig. 6 Privilege Management

系统通过角色和具体权限设定给每个用户分配权限。其中,超级用户初始设置时使用,其它用户要通过具有管理用户权限的人员创建,不支持自己注册。用户名、密码输入正确进入系统后的界面会根据登录用户拥有的权限,显示相应的信息和可操作的功能项。

3.4.2 项目管理模块的设计

项目管理主要是创建项目和进入项目执行代码同步、测试用例输入、编译程序、评测、设置项目信息等针对该项目的其它功能。创建项目时需要输入项目名称、简介、GITURL、GIT 上的被测程序分支名。使用联合测试平台时,要求将被测试的程序保存在 GIT 服务器上。需要指定被测程序在 GIT 上的地址及 GIT 分支,同时还要提供服务器公钥并将该公钥加到 GIT 服务器。

3.4.3 代码管理模块的设计

代码管理实现了从 GIT 服务器下载或更新被测程序到本测试平台。这里采用的多进程方式,在进行同步时创建一个子进程,单独处理代码同步操作。若已执行从服务器同步代码的操作,界面中会显示同步的文件及目录,供操作人员选择相应操作。文件列表采用分页方式显示,页面最下面一行有向前、向后翻页的按钮。界面设计如图 7 所示。

图 7 代码管理界面设计

Fig. 7 Code management interface design

3.4.4 测试用例管理模块的设计

操作人员可创建测试用例、输入测试命令、删除测试用例,也可执行评测。可以列出已经创建好的测试用例,包括“编号、命令、允许误差、创建时间、创建人”。其中,“允许误差”是指当预期结果与实际结果偏差小于等于该值是,则认为通过了测试。“创建”或“修改”测试用例时,可以输入此值。

选择某一测试用例后点击“评测”,即可进行自动执行该测试用例的评测工作;或者点击左侧选择框,选中某些测试用例进行评测。也可以点击“评测所有测试用例”对列表中的所有用例进行评测。每一条后面的“修改”项,可用来修改该条测试用例的测试命令;“测试用例”项可用来上传被测文件和预期结果文件。

3.4.5 编译文件的管理

对从 GIT 服务器同步到本系统的代码需进行编译,编译成功会生成执行文件,否则需要通知编程人

员修改被测程序，并重新上传至 GIT 服务器。编译后的文件管理包括：压缩文件、将文件复制到某个测试用例中、将文件复制到指定位置、文件重命名、创建文件夹。

3.4.6　代码联合测试

测试前要保证各项信息设置正确，评测才能正常进行。若已有完成的评测任务，界面中应显示评测任务列表，其中包括该任务的“编号、类型、创建时间、结束时间，状态、通过、未通过、用例总数”等信息。“通过”表示该任务通过测试的用例数；“未通过”则表示测试结果未达到误差要求；“出错”表示测试时出错的用例数。“未通过”的测试任务界面如图 8 所示。

图 8　评测结果未通过时

Fig. 8　When the evaluation result fails

3.4.7　审核功能模块的设计及实现

经过上述测试，如果测试结果符合要求则发出审核请求，该项目进入被审核状态。审核时该项目被锁定，许多操作将被限制，直到审核完毕。审核通过后的项目状态如图 9 所示。

图 9　审核界面

Fig. 9　Audit interface

4　结束语

针对由大量独立模块组成的建筑全性能仿真平台内核，为了验证其功能正确性及其能耗计算是否达到精度要求，提出了建筑全性能仿真平台内核联合测试平台的设计与实现方案。平台在设计上采用了面向服务的分层设计结构，验证方法采用了国际建筑能耗对比标准。通过程序间对比法和实验验证法，对建筑全性能仿真平台内核功能正确性及计算准确性进行了测试，实现了对全性能仿真平台计算内核的全面检验。测试平台从代码管理平台直接同步代码到测试平台，简化了测试流程，被测模块源代码直接在测试平台被编译并组装运行，避免了不同平台模块间的兼容问题。经过实际使用表明，所设计实现的平台达到了预期要求，完全满足实际需要。

参考文献

[1] JORGENSEN P C, ERICKSON C. Object－oriented integration testing[J]. Communications of the ACM, 1994, 37(9): 30－38.

[2] 毛澄映, 卢炎生. 构件软件测试技术研究进展[J]. 计算机研究与发展, 2006, 43(8): 1375－1382.

[3] MARINOV D, KHURSHID S. TestEra: A novel framework for automated testing of Javaprograms[C]//Proceedings 16th Annual International Conference on Automated Software Engineering (ASE 2001). IEEE, 2001: 22－31.

[4] 朱菊, 王志坚, 杨雪. 基于数据驱动的软件自动化测试框架[J].. 计算机技术与发展, 2006, 016(005):68－70.

[5] 接卉, 兰雨晴, 骆沛. 一种关键字驱动的自动化测试框架[J]. 計算機應用研究, 2009, 26(3): 927－929.

[6] NOLLER J A, MASON JR R S. Automated software testing framework: U.S. Patent 7,694,181[P]. 2010－4－6.

[7] YU L, TSAI W T, CHEN X, et al. Testing as a Service over Cloud[C]//2010 Fifth IEEE International Symposium on Service Oriented System Engineering. Ieee, 2010: 181－188.

[8] NOUIDUI T, WETTER M, ZOU W. Functional mock－up unit for co－simulation import in EnergyPlus [J]. Journal of Building Performance Simulation, 2013, 7(3): 192－202.

[9] 刘攀, 缪淮扣, 曾红卫, 等. 基于 FSM 的测试理论, 方法及评估[J]. 计算机学报, 2011, 34(6): 965－984.

[10] 孙红三, 燕达, 吴如宏. 基于 DeST 平台的联合仿真系统开发[J]. 建筑科学, 2018, 34(10): 2－8.

[11] 孙丽. DeST 内核代码评测及搜索系统的设计与实现[D]. 哈尔滨: 哈尔滨工业大学, 2018.