一种基于Web服务的嵌入式系统BSP远程测试框架

迟 剑[1] 白晓颖[2] 万国良[3] 胡林平[4]

(河北民族师范学院数学与计算机系 承德 067000)[1] (清华大学计算机科学与技术系 北京 100084)[2]
(飞思卡尔半导体(中国)有限公司北京分公司 北京 100022)[3] (中国航空工业第六三一研究所 西安 710068)[4]

摘 要 在嵌入式BSP测试过程中,为每一个测试产品和每一个开发团队都建立测试环境来进行本地测试,其代价非常昂贵。采用远程测试的方法可以降低测试代价,共享测试环境,有效地重用测试用例。Web服务是一种部署在Web上的对象,建立在以XML为主的、开放的Web规范技术基础上,具有平台无关性、互操作性、功能复用和安全通信特点。提出了一种嵌入式系统BSP远程测试框架,该框架分为测试中心服务和测试门户服务,分别采用了SOAP和REST两种风格的Web服务包装形式,用户可以通过统一的Web接口访问服务实现BSP的远程测试。

关键词 BSP测试,远程测试,Web服务,SOAP,REST

中图法分类号 TP311.56 **文献标识码** A

Web Service-based Embedded BSP Remote Testing Framework

CHI Jian[1] BAI Xiao-ying[2] WAN Guo-liang[3] HU Lin-ping[4]

(Department of Mathematics and Computer, Hebei Normal University for Nationality, Chengde 067000, China)[1]
(Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China)[2]
(Freescale Semiconductor(China) Company Limited Co Ltd the Beijing Subsidiary Company, Beijing 100022, China)[3]
(No. 631 Institute of China Aviation Industry, Xi'an 710068, China)[4]

Abstract In the embedded BSP test process, supposing we establish all test environments for each product and their development team, the testing cost is very expensive. Using the remote testing method can reduce testing cost, shared test environment, effectively reuse test cases. Web service is an object deployed in the Web on, it is based on XML, open standard technology on the Web, and has platform independence, interoperability, functional characteristics of reuse and secure communications. In the paper a framework for embedded system BSP remote test coverage which embedded BSP test in the entire process was presented, and the framework is divided into test center and test portal. Using SOAP and REST, two styles of Web services, the user can access the BSP remote test through a unified Web interface.

Keywords BSP test, Remote test, Web service, SOAP, REST

嵌入式BSP是联系嵌入式软硬件之间的桥梁,由于嵌入式系统的软硬件结合紧密,又往往具有多个软硬件配置方案,因此在测试之前必须按照被测系统设置测试环境,这样测试结果才具有实用价值。对于嵌入式BSP测试来说,为每一个产品和每一个开发团队建立一个测试环境的代价非常昂贵,特别是由不同团队之间合作开发的全球性产品。由于嵌入式系统环境的依赖性,对嵌入式BSP系统的远程自动测试提出了很大的挑战,其中亟待解决的问题是:

(1)如何能够根据测试需求自动生成测试用例以及定义测试用例之间的依赖关系从而生成测试计划,避免测试人员大量的重复劳动和降低测试过程中引入错误的概率;

(2)如何使用Web服务的方式把依赖于本地环境的测试引擎部署到Web服务器,使得用户能够在Internet方便地访问;

(3)如何设计用户界面友好的测试门户入口,并能够方便地调用测试引擎的Web服务;

(4)如何自动收集测试结果并供用户进行远程查看。

远程测试技术是网络技术与计算机技术的结合,是测试领域技术发展的必然结果。充分利用远程测试技术可以有效地利用网络资源,降低测试代价,共享测试环境,有效地重用测试用例。

Web服务技术是SOA(面向服务的体系架构)[3]的基础技术实现方式,是创建可互操作的分布式应用程序的新平台。Web服务的主要目标是实现跨平台的可互操作性,使用Internet上统一、开放的标准,如HTTP、XML、XSD、SOAP、WSDL、UDDI等,Web服务可以在任何支持这些标准的环境中使用,为异构平台的信息交互、跨平台的应用集成、B2B集成、新旧应用的集成及应用和数据重用提供了有效的解决方案。本文提出了一种基于Web服务的远程测试框架来解决嵌入式BSP系统的远程测试问题。

本文受航空科学基金(20091958005)资助。

迟 剑(1979—),男,讲师,主要研究方向为软件工程、软件测试,E-mail:leekchj@gmail.com;白晓颖 副教授,主要研究方向为软件工程、软件测试;万国良 工程师,主要研究方向为嵌入式BSP测试;胡林平 高级工程师,主要研究方向为软件测试。

1 BSP 远程测试

嵌入式 BSP[1](Board Support Package 板级支持包)处于软硬件交界的中心位置,其结构与功能互相之间表现出较大的差异,其主要目的是为了支持嵌入式操作系统,使之能够更好地运行于嵌入式硬件主板。BSP 为上层软件提供访问硬件的手段,将嵌入式应用软件所涉及的系统硬件特性完全屏蔽,不同的嵌入式处理器和嵌入式操作系统对应于不同的 BSP。BSP 和软硬件之间的联系都非常紧密,如果硬件稍做改动,BSP 相应的部分也必须进行修改来适应这种变化,因此针对 BSP 的测试必须搭建 BSP 运行的仿真环境,包括 BSP 运行的硬件环境和软件环境。

传统的 BSP 测试通常在本地进行,测试主机通过串口发送测试指令至测试目标板,并且由接收测试目标板同样通过串口回馈的测试结果进行分析,并得出测试结果。但是对于大型的公司或者开发团队来说,一个 BSP 产品通常是由跨区域的多团队协同开发并进行测试的,如果为分布在各地的开发工程师和测试人员对每一款产品各建立一个专用的测试环境,花费在沟通和邮寄目标产品上的时间代价有时是不可接受的。解决这个问题需要设立一个测试中心,该中心包括目标产品和测试引擎,然后测试人员利用 Internet 进行测试计划、测试用例和测试脚本的编写、修改和执行,进行远程测试。基于 Web 服务的 BSP 的远程测试的主要目的为:

测试资源重用:在测试中心中为产品建立一个专用的测试环境,不同的团队可以共享测试中心中的测试环境、测试用例和测试脚本。

实现协同测试:对于需求规格说明、测试配置、测试执行和测试结果收集均提供相应的网络工具,可以协调不同角色的团队参与测试过程。

高效质量控制:通过测试过程管理和测试技术的使用,可以从需求覆盖分析、缺陷生存期管理、统计评价和预测等几方面的基础上对测试进行质量控制。

2 Web 服务技术

Web 服务[2]本质就是通过 Web 接口完成某个功能的程序段,在 Internet 上通过 HTTP 等协议可以容易地访问该功能。Web 服务是一种部署在 Web 上的对象,具有面向对象技术的所有优点,建立在以 XML 为主的、开放的 Web 规范技术基础上,具有比现有对象技术更好的开放性,是建立可互操作的分布式应用程序的新平台,具有平台无关性、互操作性、功能复用和安全通信特点。Web 服务技术是 SOA(面向服务的体系架构)[3] 的基础技术实现方式,基本分为两种风格:SOAP[4] 和 REST[5]。

SOAP(Simple Object Access Protocol)简单对象访问协议是在分散或分布式的环境中交换信息的简单的协议,是一个基于 XML 的协议,它包括 4 个部分:

SOAP 封装(envelop),封装定义了一个描述消息中的内容、发送者、接受者以及如何处理消息的框架;

SOAP 编码规则(encoding rules),用于表示应用程序需要使用的数据类型的实例;

SOAP RPC 表示(RPC representation),表示远程过程调用和应答的协定;

SOAP 绑定(binding),使用底层协议交换信息。

REST(REpresentational State Transfer)表述性状态迁移风格是一个抽象的概念,它提供了一组架构约束,当作为一个整体来应用时,强调组件交互的可伸缩性、接口的通用性、组件的独立部署,以及用来减少交互延迟、增强安全性、封装遗留系统的中间组件,而符合这一风格最著名的就是 HTTP 协议。REST 风格的原则[6]是:

网络上的所有事物都被抽象为资源;

每个资源对应一个唯一的资源标识符;

通过通用的连接器接口对资源进行操作(如 HTTP 中的 GET、POST、PUT、DELETE);

对资源的各种操作不会改变资源标识符;

所有的操作都是无状态的。

SOAP 和 REST 之间的主要区别如表 1 所列。

表 1 SOAP 和 REST 的区别

	RPC	REST
地址模型	一个 URI 可对应多个资源	URI 和资源一一对应
接口方式	每一个服务可以定义它自己的操作集	使用标准通用操作对基于 URI 标识的资源统一处理
中间媒介	不通过 URI 来操作网络资源	能够充分利用实现 HTTP 协议的 Web 中间媒介操作网络资源
安全性	对象名在方法的参数中,因此需创建新安全模型	只需匿藏 URI 就可以隐藏某个资源

SOAP 协议对于 SOAP 消息体和消息头都有定义,同时消息头的可扩展性为各种互联网的标准提供了扩展的基础;REST 具有高效以及简洁易用的特性,这种高效面源于其面向资源接口设计以及操作抽象简化了开发者的不良设计,同时也最大限度地利用了 Http 最初的应用协议设计理念。

SOAP 对于异构环境服务发布和调用,以及对厂商的支持都已经达到了较为成熟的阶段。不同平台,开发语言之间通过 SOAP 来交互的 Web 服务都能够较好地互通;由于 REST 只是一种实现资源操作的思想,因此各个 REST 实现都自有一套,在性能和可用性上会大大高于 SOAP 发布的 Web 服务,但统一通用方面远远不及 SOAP。

REST 对资源型服务接口来说很合适,同时特别适合对于效率要求很高,但是对安全要求不高的场景;而 SOAP 的成熟性可以给需要提供给多开发语言的,对于安全性要求较高的接口设计带来便利。

3 BSP 远程测试框架

BSP 远程测试框架,使用 Web 服务的方式把自动化测试框架的主要功能封装成 Web 服务,用户可以通过统一的 Web 测试门户来使用这些服务实现测试。整个远程测试框架被分成两部分:测试中心服务和测试门户服务。其中测试中心服务主要负责测试的整个过程,包括测试需求和测试用例管理、测试计划生成、测试执行和测试结果收集分析等基本功能,并且把这些功能以 Web 服务的标准发布成为 Web 接口,而测试门户服务负责调用这些接口,并设计一个网页客户端把对用户的功能发布供用户调用。该框架的框架图如图 1 所示。

在本框架中，市场人员可以为目标产品定义测试需求，并把这些需求通过网络提交给 Test Server 进行处理；测试人员通过网络来审核这些需求并且把需求映射为测试用例，这些测试用例可以是重用现有的测试用例，也可以是重新定义的测试用例；测试人员需要制定测试计划，定义测试用例之间的依赖关系，并提供测试必要的配置参数；测试计划和测试用例将会通过特定的接口传送到 Host 主机，由主机来启动测试引擎，按照测试计划通过串口在测试对象上顺序执行测试用例所对应的测试脚本，并收集测试结果信息；系统管理员可通过网络监督干预测试过程，也可以收集测试日志并进行分析。

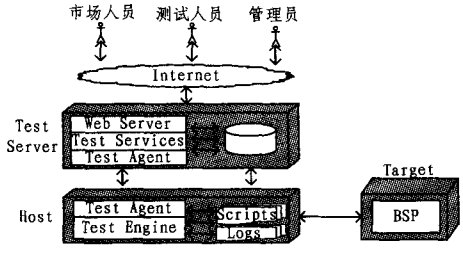

图 1　BSP 测试框架

由此可以看出，框架的核心由两部分组成，一是测试计划的建立、调整和执行，称为测试中心服务；二是远程控制，也就是使用 Web 服务发布测试需要的一系列功能，由用户通过 Internet 进行远程访问和操作，称为测试门户服务。

3.1　测试中心服务

测试中心覆盖 BSP 测试的全过程，目的是要实现测试过程管理、测试计划生成、测试执行和测试结果收集，其主要组成部分有：

对测试需求进行分析，建立需求的特征模型，该模型是所有需求和测试用例的层次化模型；

测试用例管理，用于管理测试用例和其对应的测试脚本；

测试计划管理，用于管理测试目标板和测试计划以及测试计划中的测试用例；

根据特征模型快速生成测试计划，尽可能重用现有的测试用例；

调用测试引擎执行测试计划；

测试引擎执行完毕后收集测试结果以供用户查看，并且进行一定的分析。

测试中心的 Web 服务最终发布成为 Web 接口，并且该接口只被特定的程序（测试门户）所调用，并不需要大量对于资源的操作；同时，测试中心可能会被扩展成为一个具有多台主机和多个测试目标的分布式框架，这些主机和目标有可能是跨多平台的，所以选择更为成熟、安全性更好的 SOAP 风格来设计测试中心服务。

SOAP 提供了一种 Web 服务的标准 RPC 调用方法，SOAP 规范定义了 SOAP 消息的格式，提供了通过 HTTP 协议来使用 SOAP 的方法。SOAP 采用 XML 的数据编码方式，也是基于 XML 和 XSD 的。

WSDL（Web Services Description Language，Web 服务描述语言）[7] 对服务的抽象接口及实现进行了描述，定义了原子服务的接口模型。WSDL 基于 XML 进行服务描述，给出了服务位置、有效操作、操作参数等信息。目前 W3C 推荐的 WSDL 版本是 WSDL 2.0，WSDL 2.0 增加了对语义信息的支持，采用 XML schema 类型定义系统进行信息类型描述。测试中心的服务通过 WSDL 描述对外发布其服务接口，该接口隐藏了服务的实现细节，允许独立于实现服务基于的硬件或软件平台和编写服务所用的编程语言使用服务，允许并支持基于 Web 服务的应用程序成为松散耦合、面向组件和跨平台的技术实现。图 2 给出了测试中心服务接口及实现的 WSDL 2.0 概念模型。

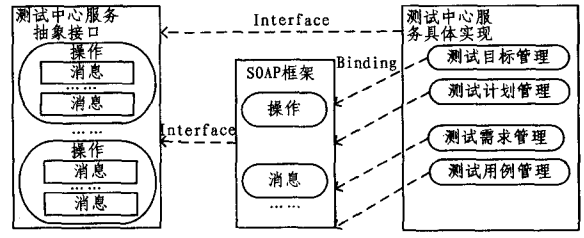

图 2　测试中心服务接口及实现

在图 2 中，消息指的是服务发送或接收的输入/输出数据的抽象类型化描述；操作指的是对服务所支持的操作的抽象描述，描述了服务提供的功能，每个操作由一系列消息组成；抽象接口是服务所支持操作的抽象集合，由一组操作组成，是服务的抽象描述；SOAP 框架使用绑定的方法将消息、操作绑定到具体的服务实现和 SOAP 网络协议上；测试中心服务实现是服务接口的具体实现描述，服务实现由服务端点描述。

3.2　测试门户服务

测试门户是一个网站服务，但是和普通的网站服务又有所不同，它的后台数据并不是直接在数据库中取得，而是通过 Web 接口由测试中心的服务得到包含有信封头的 SOAP 信息。因此，在测试门户服务中需要先处理 WSDL 中包含的 SOAP 信息才能够进行网站服务。

3.2.1　REST 风格的 Web 服务框架

REST 风格的 Web 服务架构不同于 SOAP，其本质是面向资源的架构，基于 REST 的服务架构在服务器端需要提供基于 REST 的 Web 服务。创建基于 REST 的 Web 服务有下面 3 个步骤：

首先，确定希望作为 Web 服务进行公开的所有资源，这些资源使用唯一的 URI 来标识。例如对于测试计划资源就可以使用如下 URI 来标识：/plan/{planID}，那么/plan 这个 URI 就表示所有测试计划的集合，即测试计划列表。

其次，确定操作这些资源的 API。采用 HTTP 标准的操作方法不需要另外定义 API，只需在 HTTP 包的头部标明 HTTP 的具体操作、数据格式以及操作 URI，如表 2 标明的测试计划 API。由表 2 可知，可以通过向服务器发送不同的 URI 和 HTTP 包对服务器公开的资源进行相应 CRUD 操作。

表 2　测试计划的 REST API

方法	URI	目的
GET	/plan	取得所有测试计划列表
GET	/plan/1	取得 1 号测试计划
POST	/plan/2	添加 2 号测试计划
PUT	/plan/2	更新 2 号测试计划
DELETE	/plan/2	删除 2 号测试计划

最后，在服务器端实现具体的 Web 服务。REST 风格的 Web 服务现在已有很多开源的框架可用，特别是应用在

Web2.0 上，如 Ajax[8]、Ruby on Rails、Django[9]、Apache CXF、AXIS2 等，这些框架都是符合 REST 风格的，其主要优点是：

可以利用缓存 Cache 来提高响应速度；

通讯本身的无状态性可以让不同的服务器处理一系列请求中的不同请求，提高服务器的扩展性；

浏览器即可作为客户端，简化软件需求；

相对与其他叠加在 HTTP 协议之上的机制，REST 的软件依赖性更小；

不需要额外的资源发现机制；

在软件技术演进中的长期兼容性更好。

REST 风格的 Web 服务框架一般都是符合 MTV(Model-Template-View)架构的，但是随着 MVC(Model-View-Control)架构的逐渐兴起，越来越多的 REST 风格的 Web 服务框架开始加入了控制器，转为 MVC 架构。

3.2.2 测试门户的服务框架

测试门户的框架也是一种符合 REST 风格的 Web 服务框架，同时也遵循 MVC 架构。该框架的数据来源于测试中心的服务发布，最后以网页模板的形式发布出来，其框架结构图如图 3 所示。

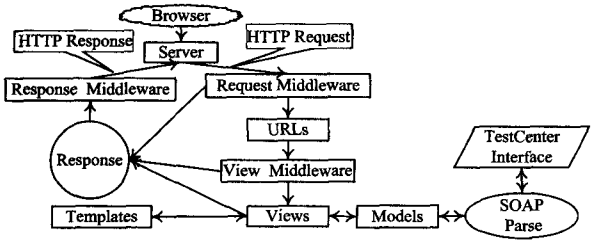

图 3 测试门户框架

在图 3 的测试门户框架中，除了 MTV 架构必备的 Models、Views、Templates 以外，主要组成部分有：

Server：浏览访问服务，接收用户的 HTTP 请求并返回给用户 HTTP 响应。

request_middleware：是一个保存请求方法的列表，它们是真正的方法，可以直接呼叫；同理，view_middleware 是一个保存视图方法的列表，response_middleware 是一个保存响应方法的列表。

URLs：由请求方法分发得到的 URL 列表，该列表与视图方法是松耦合的。

SOAP_Parse：SOAP 消息解释，可以从包含信封头的 SOAP 消息中得到所需要的数据。

测试门户的服务框架通过测试中心的服务框架得到数据，以模板的方式规范数据，并且使用标准的 HTTP 方法和用户进行交互。

3.2.3 测试门户和测试中心的交互

测试门户本身并不具备数据存储，所需的数据均由测试中心服务得来。测试中心服务接口所提供的数据是一个 SOAP 格式的信封，真正需要处理的数据封存在信封中，这样就需要对 SOAP 信封进行解析。SOAP 成功地解决了异构网络环境中的软件组件和应用程序之间进行可靠信息交换的难题，一般开发环境都隐藏了 SOAP 消息处理的过程，这部分功能封装于由开发环境自动生成的客户端和服务器端的 SOAP 代理模块中，其中都使用了编译好的高层类库。

SOAP 消息的生成和解析工作与 WSDL 文档密切相关，只有遵照 Web Service 对应的 WSDL 文档中描述的信息格式，SOAP 消息才能得到正确的、自动化的处理。本框架中选用了一个 Python SOAP 类库 ZSI[10] 来解决这个问题，只需给出 WSDL 的地址，ZSI 可以将其解析成一个 Python 类，WSDL 里面描述的接口解析成该类的方法，用户可以随时通过 URL 调用该类的方法。

在测试门户中，需要有一个 URL 和视图函数之间的对应关系。这样，用户在网页模板中进行的操作，通过视图函数绑定到 URL，再由 URL 调用 WSDL 中的接口，从而实现和测试中心之间的交互。

4 框架实现

本框架有两个组成部分，即测试中心 TestCenter 和测试门户 TestPortal。其中 TestCenter 使用 J2EE+Xfire 平台实现，Xfire[11] 是一个轻量级的 SOAP 引擎，代码轻便，结构清晰，使用该引擎可以方便快捷地把 TestCenter 的功能接口发布成 Web 服务。TestPortal 使用 Django+HTML+JavaScript+Ajax 平台实现，其中 Django 是核心框架，负责 URL 的分发和测试中心接口的调用，还需要完成数据和模板之间的交换；网页模板使用标准的 HTML 语言编写；使用 JavaScript 处理页面控件动态效果或者控件之间的关系，并处理数据以及输入数据时或提交数据前的数据验证；Ajax 向服务器发送并取回必需的数据，它使用 SOAP 或其它一些基于 XML 的 Web 服务接口，并在客户端采用 JavaScript 处理来自服务器的响应。该原型系统的体系结构如图 4 所示。

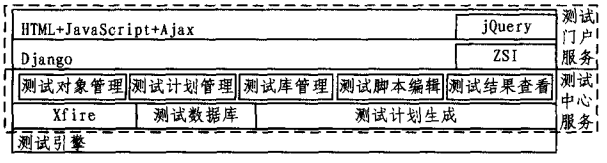

图 4 原型系统体系结构

从图 4 中可以看出，该系统的主要功能有：

系统服务发布，可以将测试中心各个功能以 Web 服务的形式发布，发布的接口以 WSDL 的形式进行规范；

SOAP 解析，可以将 WSDL 中的接口解析成为 URL；

URL 分发，可以将 URL 绑定到 TestPortal 中的视图函数；

视图函数，可以将用户的 HTTP 请求转化为对 URL 的 GET、POST 操作；

用户交互，网页模板负责和用户之间实现交互，能够接收用户浏览器的输入请求以及将服务器反馈返回给用户浏览器。

底层调用，可以在服务器端调用测试引擎实现和服务器直接通过串口相连的嵌入式目标板的自动测试。

4.1 测试中心实现

测试中心将测试需求、测试用例、测试目标、测试参数和测试计划分别建模后统一分类保存在数据库中，使用 JDBC 连接数据库进行操作，以完成测试库和测试计划的管理工作；在后台依据测试用例的依赖模型；使用拓扑排序技术将测试

计划中的测试用例排序，并将顺序传递给测试执行引擎；测试执行引擎根据测试计划中的测试用例执行顺序，取得以文件形式存放的对应测试脚本，并通过串口连接测试目标板，在测试目标板上执行该脚本，随后收集此脚本的执行结果并返回测试主机，把测试结果以文件的形式存放于主机存储。

测试中心的主要功能有：

1)测试需求管理，主要用于管理市场人员可能提出的测试需求，建立通用需求特征模型；

2)测试用例管理，主要用于管理用来测试需求的测试用例，建立通用测试用例特征模型；

3)测试对象管理，主要用于管理测试的对象，即嵌入式系统开发板，建立测试对象的特征模型；

4)测试计划管理，主要用于提供用户制定测试计划，配置测试环境参数；

5)测试脚本管理，主要用于管理测试脚本；

6)测试执行，通过串口连接测试目标，然后在目标板上顺序执行测试计划中的测试脚本，该功能面向用户是透明的；

7)测试日志的收集和分析。

另外，测试中心还提供一系列管理员管理的功能，如测试数据库的选择维护等。测试中心将这些功能全部包装成Web服务的接口，服务部署于Apache服务器，以Webapp的形式发布服务，最终的服务接口以WSDL的形式描述出来。

4.2 测试门户实现

系统是一个网站服务，基于Django＋HTML实现，其中使用到了Ajax和Javascript技术。TestPortal的基本思想是首先使用WSDL描述初始化ZSI中的Serviceproxy，这时ZSI就能够解析WSDL中携带的SOAP消息为正常的数据；ZSI解析后的数据将由Django框架中的URL分发器分发到对应的处理过程，在Django中的视图函数将会把数据进行处理后反馈给网页模板，由网页模板进行相应的更新。由于REST式的Web服务把一切资源均抽象为URL，因此视图函数也抽象为URL。为了保证整个系统的安全性，在开发过程中把对URL的操作限定在POST和GET，这样可以使得数据资源不会因为数据服务的错误而造成数据安全性和完整性受到影响。

Django中的URL和视图函数之间是松耦合的，决定URL返回哪个视图函数和实现这个视图函数是在两个不同的地方，这样可以有效地避免逻辑错误；同时，一个函数也可以被两个甚至更多URL使用，利用Django支持的正则表达式匹配也可以实现动态URL绑定，进一步提高了系统的可重用性。

需要进行管理的数据较多，为了提升网页的效率，在网页端使用了JavaScript＋Ajax的方式来进行数据的处理。在HTML代码中使用了变量和模板标签来规范文档的显示格式和提高代码的可重用性；Ajax的引入可以使得在服务器和浏览器之间交换的数据大量减少，减少了带宽占用；Ajax引擎在客户端运行，承担了一部分本来由服务器承担的工作，从而减少了大用户量下的服务器负载，结果会使得网页的应用更加快速。

测试门户服务也部署于Apache服务器，以网页模板的形式发布服务，其测试计划管理的运行界面如图5所示。

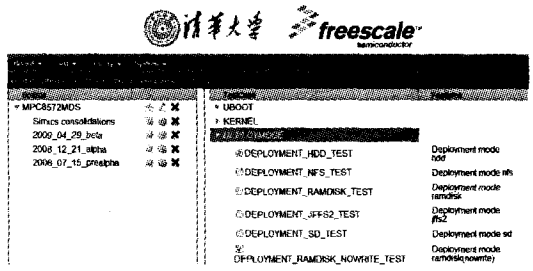

图5 原型系统的界面

结束语 本文提出了一种基于Web服务的嵌入式BSP测试框架并实现了框架的原型，实现了远程嵌入式BSP的自动测试。在本框架中，运用了SOAP和REST两种风格的Web服务，既保证了在测试过程中有良好的效率，也保证了友好的用户接口和安全性。

本文基于Web服务的思想在远程测试方面做出了初步尝试，还有许多需要去解决和改进的地方。在下一步研究中，可以考虑构建测试对象集合BoardFarm，使用多台主机和多台服务器进行分布式测试，其框架图如图6所示。

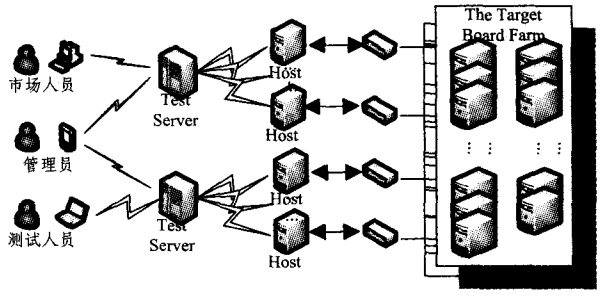

图6 BoardFarm的框架图

进一步的研究目标包括：

客户端支持不同的用户角色以不同的方式接入网络中；

客户端通过互联网可以接入不同的测试服务器；

每一个测试服务器可以控制多个测试主机进行测试；

每个主机可以通过选择器连接多个测试目标，可以动态选择测试目标进行测试。

参 考 文 献

[1] Wind River Company, VxWorks 5.5 BSP Developer's Guide [OL]. http://www.windriver.com/licensing/documents

[2] The Web Services Protocol Stack[OL]. http://roadmap.cbdiforum.com/reports/protocols/, 2005

[3] Web Services Architecture[S]. W3C Working Draft[OL]. http://www.w3.org/TR/ws-arch/, 2004

[4] W3C. SOAP Version 1.2 Part 0:Primer(Second Edition)[OL]. http://www.w3.org/TR/soap12-part0/. W3C Recommendation, 27 April 2007

[5] REST[OL]. http://java.sun.com/developer/technicalArticles/WebServices/restful/

[6] Fielding R T. Architectural Styles and the Design of Network-based Software Architectures[D]. University of California, Irvine, 2000

[7] Web Services Description Language(WSDL) Version 2.0 Part 1:Core Language[OL]. http://www.w3.org/TR/wsdl20/

[8] Ajax[OL]. http://www.asp.net/ajax/

[9] Django[OL]. http://docs.djangoproject.com/en/1.1/

[10] Python ZSI[OL]. http://pyWebsvcs.sourceforge.net/zsi.html

[11] XFire[OL]. http://xfire.codehaus.org/