PM 机制在本地化软件 BV 测试中的应用

梁惠平[1],刘 琦[2]

(1.中国科学院研究生院,北京 100039;
2.中国科技大学 电子科学与技术系,安徽 合肥 230027)

摘 要:分析了当前本地化版本验证测试 BV 中存在的问题,提出了 PM。相比当前的 BV 测试,PM 机制实现了所有测试结点、测试报告有机统一的管理。PM 机制有助于提高整个 BV 测试系统的管理能力、扩展能力和测试性能。

关键词:版本验证;策略模型;配置匹配服务器,配置匹配客户端

中图分类号:TP302.1 **文献标识码**:A

Application of PM to Localization BV testing

LIANG Hui Ping[1], LIU Qi[2]

(1.Graduate School of the Chinese Academy of Sciences, Beijing 100039, China;
2.University of Science and Technology of China Electronic Science and Technology Department, Hefei 230027, China)

Abstract: This paper analyzes the drawbacks of current localization build validation (BV) testing, proposes Policy Model(PM). Compared to current BV, PM can implement central and uniform management of all testing points and testing reports. The model can help to improve the whole BV testing system management and extent ability and testing performance.

Key words: build validation; policy model; deployment map server; deployment map agent

随着测试工具的不断完善,使得企业追求最优的质量成本成为可能。目前在软件开发过程中,更多的会采用迭代式的开发过程。在开发过程中强调在较短的时间间隔中产生多个可执行、可测试的软件版本。软件的本地化过程更是如此:会频繁地生成多个所有要支持的语言版本。一般来说,公司会根据市场需求决定语言数量。但是软件本地化是软件重要的竞争因素之一。所以为了将来占有更多的市场,公司都会更大范围地执行软件本地化。一些特殊的软件,例如:微软的 Windows 和 Office 软件,需要支持几百种语言。在多个语言版本的产生过程中,由于软件设计的不全面或者更多其他原因,例如:软件设计过程中没有考虑双字节字符的支持和字符长度处理问题或者软件周期短,都不可避免地导致出现一些坏的语言版本。特别是一些重大的错误存在于软件的安装、卸载过程中或者一些重要功能模块的基本操作中。如果坏的语言版本比较多或者某个语言坏得版本太多,都会大大延误测试进度,测试成本提高。

目前大多数公司都会在所有测试人员介入之前,对软件作版本验证 BV 测试(Build Validation),保证版本的质量,以便可以更好地控制测试进度[4]。目前的 BV 测试存在的问题:

(1)测试系统对测试结点不做任何管理。当某个测试机发生故障,无法自动察觉。

(2)测试系统对测试数据不做任何管理和分析。只简单保存,无法挖掘更有用的信息。

(3)测试系统总是选择重要的语言进行测试。某些语言长期被疏忽,一些重大问题无法及时发现。

(4)系统扩展性、自动性差。当测试结点达到一定数量,需要更多的人工干预。

下面将介绍目前本地化软件 BV 测试的结构存在的问题以及引入 PM 机制后的 BV 测试。

1 本地化软件的 BV 测试

目前,大多数公司都要求写自动化测试脚本,以便在新的版本生成后通过运行脚本达到版本检验的目的。

软件本地化过程中，对本地化版本的 BV 测试与英文的 BV 测试的区别在于需要运行的语言平台数量不同。英文的 BV 测试只在英文平台上运行，而本地化的测试在理想状态下需要在所有支持的语言平台上运行。由于产品市场的影响，对于所有要支持的语言的重要程度和优先级别是不同的。同时，受到测试周期和预算的限制，一般不会对所有的语言都进行 BV 测试，而只选择重要的语言。

在目前的 BV 测试中，多个语言平台的架构可以是多个独立的 PC，也可以使用 VMWare。如图 1 所示。

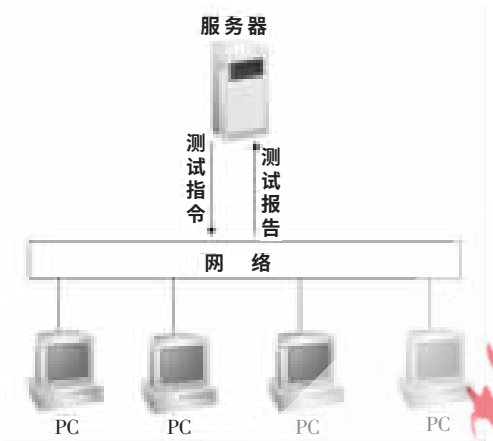

图 1 目前 BV 测试架构

当服务器接受到新版本生成信息，就向所有测试机发出“测试指令”，每个语言测试平台开始独立进行测试，测试结束后产生测试报告并发送到服务器。服务器对各个语言平台的测试机除了发送测试开始指令外，不进行任何其他管理。如果某个测试机出现故障，无法正常工作，服务器无法感知并通知管理人员。每个测试平台直到测试结束后才向服务器发送测试报告。而且每个测试报告都是独立的，不利于对测试结果进行综合分析，产生更加有意义的信息。所有这些都使得 BV 测试不够完善。

2 策略模型 PM(Policy Model)机制[1-3]

2.1 PM 机制的结构和流程

PM 机制的具体实现、采用了 PM 机制后的 BV 测试架构，集中统一管理 BV 测试结构中的所有测试结点，把 BV 测试过程中的所有数据信息集中统一管理并生成内容丰富的报表。

2.1.1 PM 机制的结构

在 PM 机制中主要包括两种对象：一个 DMS(Deployment Map Server) 和多个 DMA (Deployment Map Agent)。

DMS 是整个 PM 机制的核心，DMS 在开始测试前主动地发现并收集每个测试结点信息。并保存最新的测试结构信息和测试结构中每个测试结点的策略信息和测试结果。其中测试结构信息不仅包括了每个结点的硬件信息和软件信息，还包括网络连接状态和层次结构。

DMA 是客户端，会自动向 DMS 发送结点的最新状态，包括是否正常启动、层次结构信息、测试的结果、策略状态信息等等。当结点有任何内容更改，DMA 会自动通知 DMS 更新信息。当 DMS 无法访问时，DMA 会暂时保存信息，并不断尝试访问 DMS，直到把信息发送给 DMS。所以只要在要测试的机器上安装 DMA，此结点的所有物理信息以及所有测试过程中的动态信息会自动发送到 DMS，保证了 DMS 中信息的正确性和及时性。

在 PM 机制中，要实现对所有测试结点和数据信息的集中统一管理。

(1)实现对所有测试结点的统一管理，PM 机制主要通过由 DMS 向测试结构中的所有结点配置同一测试策略来实现。策略主要分为以下 3 种：

配置策略(Configuration Policy)：实现所有测试结点的统一系统和软件配置。

任务策略(Job Policy)：实现所有测试结点的统一操作和管理。

BV 测试策略：实现所有测试结点正确实施 BV 测试。当一个新的版本生成后，DMS 自动生成一个相应的软件测试包，并根据用户定义生成一个对应的测试策略。策略一经生成，整个测试将完全按照策略定制的信息在整个测试结构中的所有结点实施。

(2)实现数据信息的统一管理，PM 机制主要通过 DMS 的主动收集和 DMA 的自动更新实现信息的自动同步和管理。

2.1.2 PM 机制的工作流程

PM 机制的工作流程如图 2 所示：

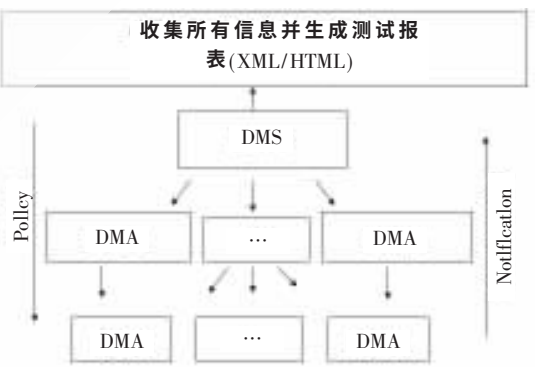

图 2 PM 机制的工作流程

通过 PM 机制，实现了对 BV 测试中所有测试结点以及所有测试结点的测试信息的集中统一管理。

2.2 PM 机制中如何实现信息的自动收集和同步[5]

PM 机制中信息的自动收集和同步是整个机制的核心功能。在 PM 机制中，整个测试结构中的所有结点都是相通的，呈树型结构。除了根结点 DMS 外都有父结点，除了叶子结点外都有子结点。这种父子关系是通过 PARENT 属性定义的，PARENT 关系网络连结了测试结

构中所有的结点,形成一个测试整体。

PM 机制中任何一个 DMS、DMA 都有一个相应的对象是 PMDB(Policy Model Database)。PMDB 的定义类似数据库包括可以访问的用户、组、对象以及访问策略。此外,还包含了一个 SUBSCRIBER 列表。SUBSCRIBER 可以定义为任何一个 PMDB。能够更新某个 SUBSCRIBER 的 PMDB 称作该 SUBSCRIBER 的父亲。子 SUBSCRIBER 的任何更新会自动上传到父 SUBSCRIBER 中。其中主机(Host)作为一个特殊对象,可以定义任何一个 PMDB 为其父 SUBSCRIBER。

一般来说,创建 PMDB 时需指定授权可访问的用户/组、父结点,还要指定可访问的终端对象,即 SUBSCRIBER。整个测试结构结点间的 SUBSCRIBER 关系实际上是结点中 PMDB 之间的 SUBSCRIBER 关系。

图 3 所示为 PMDB 的层次结构一般定义。

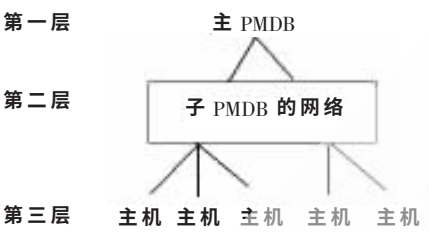

图 3 PMDB 的层次结构

整个 BV 测试层次结构的定义过程也就是 PARENT 关系网络的定义过程,同时也是 PMDB 层次结构的定义过程。而 SUBSCRIBER 关系网络定义却不同,SUBSCRIBER 关系网络可以和 PARENT 关系网络保持一致,但有时为了提高可访问性,每个 PMDB 要建立多个 SUBSCRIBER 关系。主 PMDB 通过比较更新信息的时间戳来确定是否接受新的更新。

在图 4 所示 PM 机制的工作流程图中可以看到 2 种信息流。一种是策略(POLICY);另一种是通知(NOTIFICATION)。"策略"是沿着测试结构中结点的 PARENT 网络流动。而"通知"是沿着测试结构中 SUBSCRIBER 网络流动。

图 4 PM 机制中的信息流

在 PM 机制中,通过定义结点间的 PARENT 和 SUBSCRIBER 两种关系,使测试结构中所有结点成为一个整体。通过 POLICY 和 NOTIFICATION 两种信息流,实现了所有结点和结点信息的集中统一管理。

2.3 信息存储及测试报告

2.3.1 信息存储

采用 PM 机制后,DMS 结点存储了大量的测试信息,占用很大的存储空间。如何有效地存储并管理这些数据是必须考虑的问题。在 PM 机制中采用了按照时间合并数据的方法。PM 机制会保留最近的 5 个工作日的单个测试详细信息。用户可以查看当天或者过去 4 天详细的所有语言平台测试信息。当测试信息早于 5 天前,PM 机制定期的合并信息形成每天、每星期、每月的数据。这个过程类似数据库 Rollup。图 5 所示为用户可以根据具体项目测试信息定义数据合并过程的时间表,保留多少天的详细数据和合并数据的规则。

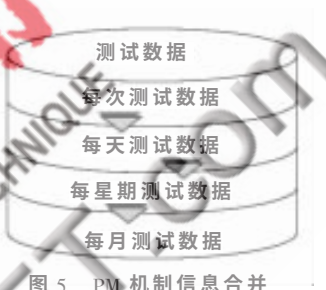

图 5 PM 机制信息合并

2.3.2 测试报告

DMS 结点存储了所有的测试信息。从信息存储策略可以看出 PM 机制可以很容易地形成各种测试报告,并且可以通过对不同语言平台测试信息的对比,挖掘出更加有意义的信息。例如:如果某个语言平台测试结果总是正常,说明本地化过程对此语言比较规范,就可以考虑更换其他语言。因为对于某些重要的语言,设计人员和测试人员都会重视并提前考虑很多。相对来说,对于比较偏的语言会考虑的少一些。虽然首先需要保证重要语言平台的正确性,但事实上比较偏的语言由于缺少重视出错的机率更大。

对大量历史数据分析的测试报告可以对此产品的开发设计有一定指导性的意义。

2.4 实验数据和性能比较

采用 PM 模型前和采用之后的对比如表 1 所示。随着结点的增加,测试系统性能对比会非常明显。

3 PM 机制的优化

目前,在 PM 机制的设计中 DMA 的层次结果可以给出很灵活的定义,但是整个 PM 机制只能有一个 DMS 结点。目前的 PM 机制不适合测试结点分布比较广的 BV 测试。并且整个测试架构中只有一个 DMS 结点对测试的高访问性具有一定的风险,今后的设计中需要加强 DMS 层的管理。

BV 测试中采用 PM 机制,实现了对所有测试结点、测试过程中各个动态测试信息的集中统一管理,通过对

表 1 采用 PM 机制前后的对比

BV 测试 / 比较项	采用 PM 机制之前	采用 PM 机制后
是否对测试结点进行管理	否 在服务器端有结点操作列表，但是服务器对每个结点不作任何管理操作	是 通过定义结点间的关系，使得所有测试结点是一个有机的整体，当某个测试机出现故障时，可以提前察觉并通知管理人员
是否对测试报告管理	否 仅仅简单保存每次的报告	是 对测试数据进行保存、合并和分析，提供灵活丰富的测试报告
系统初始化	比较简单	比较复杂 需要设计 PM 的框架，并定义每个结点的关系网络。当结点达到一定数量时，初始化过程会比较复杂，需要一定时间
增加新结点的复杂性	比较简单 只需把结点添加到服务器要操作的结点列表中	简单 设置新结点与其他已有结点的属性关系即可
当结点达到一定数量时(>10)	测试会变得杂乱无章，需要更多的人工干预，当结点多于 20 个，整个测试无法正常进行	层次结构非常清晰，容易管理，节省人力，不需要人工干预。测试结点可以无限增加（机器性能允许的情况下）
当报表达到一定数量时(>10 天的数据量，10 个测试结点)	仅仅简单存储，并只能提供一次测试结果数据。测试数据占用很大空间，难以查找	报表管理井然有序，通过对报表数据合并整合和分析，产生丰富的报表并挖掘出更有价值的信息
测试时间(实验数据)	测试时间= 测试运行时间+数据传输时间 当测试结束后传输测试报告，多个结点同一时间发送数据，容易导致网络拥塞，以至整个测试时间延长。而测试运行时间只与测试内容有关 测试内容不变的情况下： 当结点=3，测试时间=135 min 当结点=5，测试时间=237 min 当结点=10，测试时间=497 min	测试时间=测试运行时间 边测试边传输，不会造成网络拥塞现象 测试内容不变的情况下： 当结点=3，测试时间=120 min 当结点=5，测试时间=198 min 当结点=10，测试时间=396 min

测试数据的合并和分析，可以做到更有效地管理测试数据并挖掘出更有价值的信息，使得 BV 测试更加完善。

参考文献

[1] 张茂林.软件自动测试的研究与程序实现[J].北京航空航天大学学报，1997，23(1)：74-80.

[2] 赵晖，王刚.软件自动测试方法浅谈[J].雷达与对抗，1997(3)：68-72.

[3] 凌永发，张云生，郭秀萍.软件测试自动化的脚本技术[J].云南民族学院学报(自然科学版)，2002，11(1)：544-548.

[4] 王华伟，崔启亮.软件本地化——本地化行业透视与实务指南[M].北京：电子工业出版社，2005.

[5] 候俊杰.深入浅出 MFC(第 2 版)[M].武汉：华中科技大学出版社，2005.

(收稿日期：2009-06-09)