DOI: 10.13718/j.cnki.xdzk.2015.04.011

半实物环境下嵌入式软件通用测试平台研究①

郭　旺[1]，　丁晓明[1,2]，　唐海鹏[1]，　蔡东容[1]，　顾卫华[1]

1. 西南大学 计算机与信息科学学院，重庆 400715；2. 重庆市软件评测中心有限公司，重庆 400715

摘要：嵌入式软件规模日益增大，结构也越来越复杂. 当前嵌入式软件测试系统通常针对特定被测软件，从底层做起，没有一个通用平台提供支持. 本文提出了一个半实物环境下嵌入式软件通用测试平台框架，采用基于 Eclipse CDT/JDT 的跨语言源代码分析，使用 XML 文档作为测试数据交换介质，结合半实物环境下目标机的可替换性，实现了测试平台的通用性，自动化程度较高. 根据该框架建成的测试平台，支持对被测程序进行源代码分析、按类插桩、自动编译、测试数据分析等操作.

关　键　词：嵌入式软件；软件测试；通用；测试环境；交叉测试

中图分类号：TP311　　**文献标志码**：A　　**文章编号**：1673-9868(2015)04-0062-05

近年来，随着软件应用领域的不断扩展，嵌入式系统得到了充分的发展，并被广泛地应用于工业控制、智能家居、消费电子、物联网、无线通讯等领域. 随着嵌入式硬件的飞速发展，嵌入式系统软件也随之发展，软件的规模日益增大，结构也越来越复杂[1].

嵌入式软件通常跨越多个硬件平台，因此，对嵌入式软件测试是一项复杂耗时的工作. 嵌入式软件的测试既要考虑软件本身，也要考虑软件同硬件平台的集成、时间约束、实时及其它性能要求. 只在开发环境下进行嵌入式软件测试，难以保障其测试结果的有效性；而仅在目标机上进行测试，由于软件运行的不可视性，又使得测试者难以知晓程序当前的运行状态以及代码的覆盖情况[2].

当前大多采用全数字仿真平台对嵌入式软件进行测试. 被测软件运行在仿真模拟环境中，测试之前需要耗费大量的时间精力对目标机进行数字仿真，且随着嵌入式系统日趋复杂，仿真的代价越来越高. 同时测试系统主要针对特定被测软件，测试平台均从底层做起，缺乏一个很好的通用平台提供支持，导致测试周期长、资源利用率低[3]. 针对以上问题本文提出了一个半实物环境下嵌入式软件通用测试模型，该测试模型既具备全实物运行环境的真实性，又具备全数字仿真系统的透明性，可对基于不同硬件平台和开发语言的嵌入式软件进行测试.

1　相关技术

1.1　源代码分析

程序分析是软件测试的重要方法之一[4]. 通过对被测软件源代码进行静态分析，从代码中提取函数名称、参数类型等结构信息，将其结构信息以一种适于分析、索引和搜索的格式存储，即可得到抽象语法树

① 收稿日期：2014-11-12
基金项目：国家自然科学基金(61003203)；重庆市科技攻关计划(CSTC2012GGB00004)；教育部国家级大学生创新创业训练计划(201310635056).
作者简介：郭　旺(1990-)，男，陕西韩城人，硕士研究生，主要从事软件测试方面的研究.
通信作者：丁晓明，副教授.

AST. 这些关键信息可供后续的插桩植入探针、测试数据分析等环节使用. 例如通过程序流程图可以构造程序的基本路径集，它是进行路径覆盖测试的基础.

1.2 插桩植入探针

程序插桩技术能够按照用户的需求，记录被测软件运行过程中的各种信息，是软件测试的有效手段. 它是在保证被测软件逻辑完整性的基础上，在源代码中插入探针函数和通信函数. 探针函数在运行到插桩点时记录下运行情况，采集被测软件的运行特征数据，基于对这些特征数据的分析，可以揭示程序的内部行为和特征[5]. 通信程序可将探针函数记录的结果传回客户机.

1.3 测试数据分析与结果生成

被测程序在真实目标机环境中执行，探针程序将程序执行过程中的特征数据传输给宿主机测试平台，在宿主机上对目标机传回的特征数据和源程序进行匹配，根据不同的覆盖策略采用相应的计算方法计算出对应的覆盖率，绘制报表并生成测试报告.

1.4 测试平台的通用性实现

主要通过3种手段实现：

1) 引入半实物测试环境. 测试是基于半实物测试环境的，在半实物测试环境下，宿主机是测试模型的核心，目标机环境具有可替换性，只需更换目标机即可实现对不同硬件平台被测软件的测试.

2) 使用XML存储测试数据. XML具有良好的数据存储格式、可扩展性、高度的结构化、精确的数据搜索等优点[6]，提供了一种与软硬件平台无关的基于文本格式的开放共享数据的方法. 通过它提高测试数据存储的通用性.

3) 借助Eclipse CDT/JDT实现跨语言源代码分析. Eclipse CDT/JDT分别实现对C,C++语言和JAVA语言源程序的源代码静态分析，源程序中的每个语法结构对应为1个AST节点，所有AST节点按其在语法上的关系连接成1棵AST树. 通过对语法树节点的访问获得程序结构信息.

2 嵌入式软件通用测试平台设计与实现

半实物环境下嵌入式软件通用测试模型如图1所示. 测试在交叉编译环境下进行，即测试工具运行在宿主机上，而被测程序运行在目标机上，宿主机和目标机通过物理通道连接[7]. 测试工具在宿主机上对被测软件进行词法和语法分析，对源程序插桩植入探针函数和通信程序，经过链接和编译后下载到目标机上. 目标机执行被测程序并通过通信程序将探针函数采集到的数据传回宿主机，由宿主机对数据分析后得到测试结果.

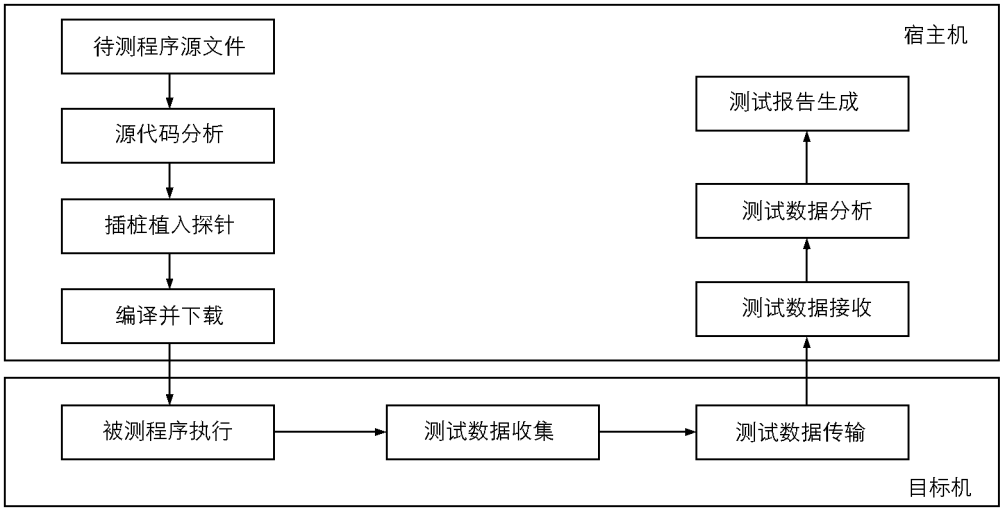

图1 半实物环境下嵌入式软件通用测试模型

根据设计框架，在 Eclipse 开发平台上实现了基于宿主机的嵌入式软件通用测试平台. 整个系统包括对被测软件源代码分析、插桩植入探针、测试数据传输、数据分析与报告生成 4 个过程.

1) 源代码分析

Eclipse CDT/JDT 开源工具包提供了获取程序 AST 的方法. 程序 AST 是在词法分析、语法分析和语义分析的过程中逐步产生的[8]. 通过修改 CDT/JDT 部分源码来定制测试所需的 AST 信息，再进行去除注释、标准化结构等操作，得到标准的被测软件程序结构信息供后续使用. Eclipse CDT 构造 AST 的过程如图 2 所示.

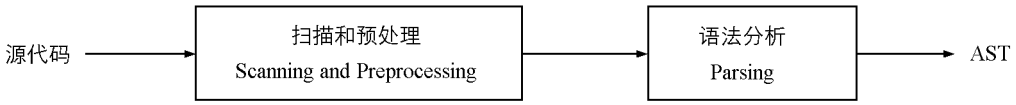

图 2　Eclipse CDT 语法树的构造过程

对 AST 的遍历采用访问者模式，访问者模式封装了一组作用于 AST 上各元素的操作. 访问者采用深度优先的原则对 AST 进行遍历，并且既可以自顶向下也可以自底向上进行[9]，使得对节点的访问方便快捷.

2) 插桩植入探针

插桩技术既要保证收集足够的测试过程信息，又要尽可能减少代码膨胀率以减小插桩对被测程序执行效率的影响. 因此在确定插桩位置时，要对程序进行划分. 此处引入块(block)[10]的概念. 块的定义是：总是在一起执行的最大的程序语句序列. 块有节点(Node)和段(Segment)这 2 类，节点包括判断、连接、程序单元的入口点和出口点；段是一段顺序执行的单入口且单出口的程序语句序列.

首先对程序基于块进行划分，进而选择插桩位置，插桩的位置通常为块的入口处. 例如函数调用的开始处和结束处，特定语句(如分支语句、循环语句、选择语句、返回语句等)的开始处和结束处. 针对不同的覆盖策略，插桩的位置有所不同. 插桩位置的选择必须建立在确保插桩后的程序可以正确运行的基础上.

3) 测试文件与数据传输

在本测试平台中，宿主机与目标机通过串口或以太网进行通信. 插桩后的被测程序在宿主机上编译链接后通过物理连接通道传输到目标机上，同时宿主机开始对指定接口进行监听. 目标机执行被测程序的过程中，植入的通信函数将探针函数获得的特征数据传输回宿主机，宿主机将得到的特征数据以结构化的形式保存到 XML 文档中. 图 3 是保存条件覆盖执行数据的 XML 文件示例.

```
1〈root〉
2〈retbool condition_return_value=“1” condition_module_Id=“1”/〉
3〈retbool condition_return_value=“1” condition_module_Id=“3”/〉
4〈retbool condition_return_value=“0” condition_module_Id=“1”/〉
5〈retbool condition_return_value=“0” condition_module_Id=“2”/〉
6〈retbool condition_return_value=“1” condition_module_Id=“4”/〉
7〈retbool condition_return_value=“1” condition_module_Id=“5”/〉
8〈retbool condition_return_value=“0” condition_module_Id=“5”/〉
9〈retbool condition_return_value=“1” condition_module_Id=“6”/〉
10〈retbool condition_return_value=“0” condition_module_Id=“7”/〉
```

图 3　保存条件覆盖执行数据的 XML 文件示例

4) 数据分析与报告生成

系统目前支持语句覆盖、分支覆盖、条件覆盖和基本路径覆盖 4 种策略. 以条件覆盖为例. 对图 3 文件中的每一行进行分析，其中 condition_return_value 值为 1 代表子条件表达式执行的值为真，condition_return_value 值为 0 代表子条件表达式执行的值为假，condition_module_Id 代表子条件表达式的标识符

ID. 对以上数据进行去重处理后得到的即为程序执行后所有表达式的执行情况，条件覆盖率通过以下算式得到：

$$\text{条件覆盖率}=\frac{\text{执行的表达式数}}{\text{条件表达式总数}\times 2}\times 100\%$$

将得到的覆盖率以饼状图形式体现，结合被测软件理论数据与实际数据生成测试报告.

3 实验结果

实验中，以飞凌 OK6410 开发板作为测试目标机，以某 LCD 控制程序作为被测软件，该程序控制目标机上 LED 屏幕显示多种颜色.

实验过程中，对被测软件源程序进行分析，选择覆盖策略，此处选择语句覆盖，根据选择的覆盖策略自动插桩，通过交叉编译环境中的 arm-linux-gcc 编译器编译后生成目标机可执行文件，通过网络文件系统(NFS)传输到目标机上，同时宿主机打开 TCP 服务端开始监听. 目标机执行被测软件，在这个过程中通信函数会将程序的运行过程中探针函数记录的信息传回宿主机. 测试平台对传回的测试数据分析后生成测试报告，测试结果如图 4 所示.

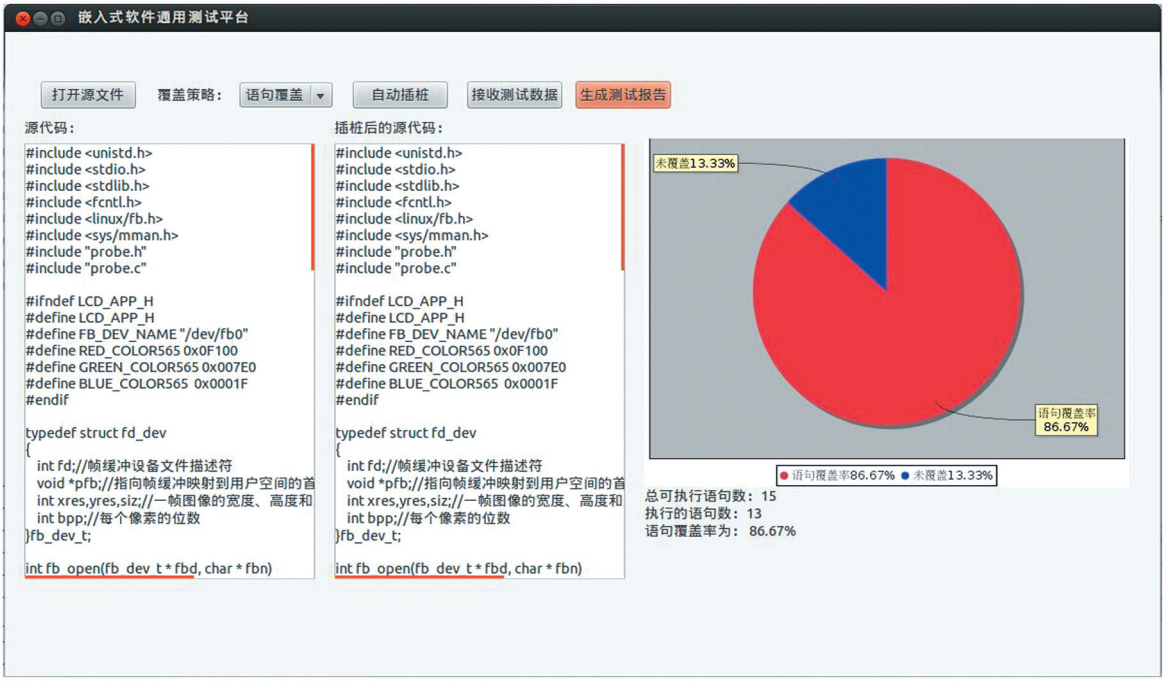

图 4 测试平台实验结果图

4 总 结

本文针对目前嵌入式软件测试通用性差、重复利用率低等问题，提出了嵌入式软件通用测试平台框架，研究了嵌入式软件测试关键技术，并最终开发实现了测试平台，该平台能够对基于 C,C++,JAVA 语言开发的嵌入式软件进行覆盖测试，并通过实验验证了平台的有效性. 进一步的工作包括改进宿主机与目标机之间的通信方式、评估插桩对被测程序的影响、支持更为严格的覆盖策略等.

参考文献：

[1] EBERT C, JONES C. Embedded Software: Facts, Figures, and Future [J]. Computer, 2009, 42(4): 42－52.

[2] 杨 俊，张 倩，林依刚. 一种嵌入式软件覆盖测试方法 [J]. 指挥信息系统与技术，2010，1(6)：24－26.

[3] 蔡建平. 嵌入式软件测试实用技术 [M]. 北京：清华大学出版社，2010.

[4] 蔡　虹，沈　雷，李永红. 基于覆盖测试的嵌入式软件自动裁剪 [J]. 计算机工程，2010，36(1)：73－75.
[5] HUANG J C. Program Instrumentation and Software Testing [J]. Computer，1978，11 (4)：25－32.
[6] 汪　洋，徐建芬，王海平. 基于 XML 的自动测试信息交换标准研究综述 [J]. 电子测量与仪器学报，2008，22 (5)：1－7.
[7] 罗克露. 嵌入式软件调试技术 [M]. 北京：电子工业出版社，2009.
[8] 高传平，谈利群，宫云战. 基于抽象语法树的代码静态自动测试方法研究 [J]. 北京化工大学学报：自然科学版，2007，34(Z1)：25－29.
[9] KUCERA M. Overview of Parsing [EB/OL] (2011－5－20) [2014－10－29]. http：//wiki. eclipse. org/CDT/designs/Overview_of_Parsing.
[10] 陈卫东，杨建军，叶澄清，等. 基于块的流图模型及其控制流图 [J]. 浙江大学学报：工学版，2003，37(2)：144－150.

Research on General Testing Platform for Embedded Software Under the Hardware-in-the-Loop Environment

GUO　Wang[1]，　DING Xiao-ming[1,2]，
TANG Hai-peng[1]，　CAI Dong-rong[1]，　GU Wei-hua[1]

1. *School of Computer and Information Science，Southwest University，Chongqing 400715，China*；
2. *Chongqing Software Testing Center Co. Ltd.，Chongqing 400715，China*

Abstract：As the scale of embedded software becomes bigger day by day，its structure gets more and more complex. Currently，the testing system is generally developed for aspecific embedded software and its development has tostart from scratch，for no general platform is available which can support the test. In this paper，a general testing platform frame under the hardware-in-the-loop environmentis proposed. Cross-language source code analysis is made based on the application of Eclipse CDT/JDT. XML document is used as the exchange platform of the testing data. With the replace ability of the target machine，a general platform with high-degree automation is realized. The functions of the testing platform include source code analysis，probe insertion by test type，automatic compilation and testing data analysis.

Key words：embedded software；software testing；general；testing environment；cross-test

责任编辑　崔玉洁