文章编号：1005-8451（2017）10-0008-04

基于Robot框架的自动化测试平台设计与实现

乔艳茹，陈晓轩，欧阳敏
（卡斯柯信号有限公司　上海测试部，上海　200070）

摘　要：为了提高嵌入式协议软件的测试效率，提出一种基于Robot框架的自动化测试方法。通过分析功能完备并且发布频繁的软件特点和困难，给出了对应的自动化测试策略；阐述了自动化测试的主要技术和框架功能，分析了该技术的优点，并通过在实际测试中的应用效果，印证了该框架的通用性，灵活性和扩展性。

关键词：测试框架；自动化测试；Robot框架
中图分类号：TP39　**文献标识码：**A

Automated testing platform based on Robot framework

QIAO Yanru, CHEN Xiaoxuan, OUYANG Min
(Shanghai Testing Department, CASCO Signal Ltd., Shanghai　200070, China)

Abstract: In order to improve the test efficiency of embedded protocol software, an automated testing method based on Robot framework was proposed. The article analyzed the software features and difficulties of complete function and frequent release, proposed the corresponding automated testing strategy, expounded its main technical framework and function, analyzed the advantages of the technology. Through the application effect in the actual test, the generality of the framework, flexibility and scalability were confirmed.

Keywords: testing framework; automated testing; Robot framework

在软件爆发的时代，软件开发周期越来越短，需求多而变化快[1]，软件迭代周期短，这对软件短时间内高质量的发布是一个艰巨的挑战，手工测试已经不能满足时间性的要求，因此对自动化测试提出了需求，希望通过自动化测试提高测试效率[2]。在软件测试过程中发现，对于功能完备、成熟的软件，其每一轮迭代大部分功能与上一轮相似或者完全相同，而测试过程中执行的测试脚本和步骤具有一致性和可重复性；这样的测试很适合做自动化测试，其不仅能够提高测试效率和保障软件质量，而且能够解放一部分劳动力，提高公司资源的利用率[3]；因此文章提出了基于 Robot 框架的自动化测试平台。

Robot Framework是一个基于python，可扩展的关键字驱动测试框架[4]，它可以用于分布式测试，接口测试，复杂应用测试。该框架拥有丰富的生态系统，包括多种测试库和工具，因此具有良好的可扩展性、松耦合性、稳定性、开放性和灵活性[5]，用户可以根据自身的需求定义接口和功能，框架的子模块不局限特定的开发语言，在扩展组件的同时整个框架不会受太大影响[6]；在性能测试时，模拟器的数量和属性是可配的[7-8]，能够支持多设备通信，具有分布式的特点，该框架为用户提供了灵活的场景配置和描述脚本。

收稿日期：2017-04-06
作者简介：乔艳茹，工程师；陈晓轩，工程师。

1　平台设计

该平台是一种基于Robot framework关键字驱动的自动化测试框架，如图 1 所示。需要用户准备测试数据，规定数据语法规则，编写测试库函数和测试工具，植入 Robot 框架，完成接口集成，被测对象是 C 语言实现的协议，需要将被测对象生成 python 识别的 pyd 文件，从而实现跨语言编程，完成被测软件自动化测试所需的结构。

2　技术实现

实现重点是测试库的建立，测试库要完成的工作主要有模拟器和被测对象的双向通信。具体由以下几个模块协作完成：Robot framework关键字模块；Nanomsg收发消息处理模块；Protocol buffer封装解

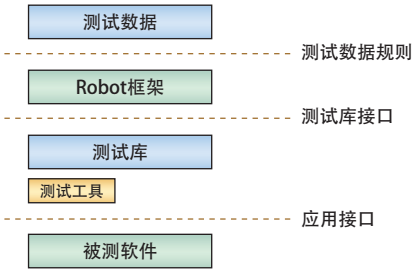


图1　平台架构

析模块，上位机解析日志模块和结果判断模块，将模拟器和被测对象结合起来完成测试工作，其中，模拟器与被测对象通过Nanomsg和Protocol Buffer完成桩信息的向下传输和被测对象 log 日志的向上传输。

2.1 Robot framework关键字模块

主要是通过用户定义的测试库函数实现模拟器的操作，包括初始化对象，向下发桩消息，发送心跳包，收集日志，日志记录以及结果检查；被测对象需要在下位机启动 Server 任务，创建协议主任务，模拟器和被测对象的交互过程，如图 2 所示。

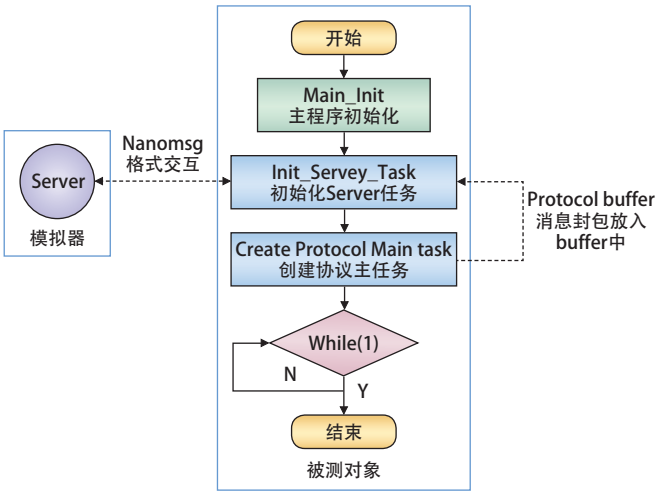


图2　数据交互过程图

2.2 Nanomsg收发消息处理模块

完成模拟器向下发桩消息，收集被测对象产生的日志信息，通信模式完成消息的收发过程，如图 3 所示。

2.3 Protocol Buffer封装解析模块

用户定义数据交互的格式，该格式是一种与通信协议、数据存储等领域的语言无关、平台无关、可扩展的序列化结构数据格式，完成平台交互信息的封装和解析，协议封包解包过程，如图 4 所示。

2.4 上位机解析日志模块

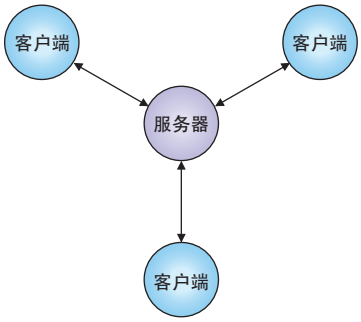


图3　Nanomsg通信模式图

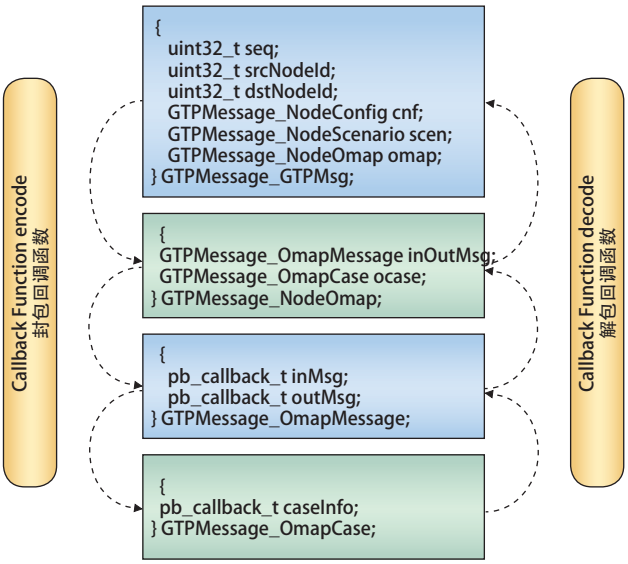


图4　数据封包解包处理流程图

上位机将下位机向上传输封装格式的日志信息进行解析并映射到字典中，将解析后的结果写入日志，如图 5 所示。

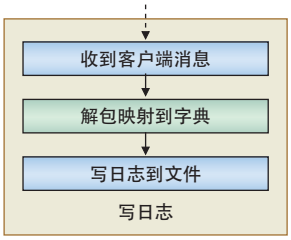


图5　日志解析记录

2.5 结果判断模块

模拟器自定义所需检查的关键字和值，通过扫描日志中所定义的关键字的值来判断是否与预期匹配，并将结果反馈给 Robot framework 接口从而将用例的执行结果可视化显示出来，如图 6 所示。

2.6 技术优点

（1）关键字封装技术：通过封装模拟器和被测

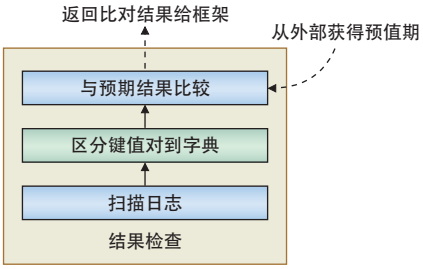


图6　结果检查

对象之间的交互接口，测试人员可以直接根据模拟器的实现接口，编写测试脚本完成用例测试，无需接触源代码，可实现测试人员快速上手和批量执行；

（2）Nanomsg 传输技术：采用可扩展协议的任务定义多个应用系统如何通信，从而组成一个大的分布式系统，测试平台采用其中的 Survey 模式；

（3）Protocol Buffer 封装技术：定义数据交互的格式，实现与通信协议、数据存储等领域的语言无关、平台无关、可扩展的序列化结构数据格式，从而实现跨平台可移植性；

（4）模拟器可配置：可实现灵活增删通信节点。

3　项目应用

该测试平台在 GM 软件中的实际应用场景如下描述。

3.1　配置

桩信息配置，如图 7 所示。

```
simulator_nodemsg.xml
<?xml version="1.0" encoding="utf-8"?>
<!--设置模拟器发给下位机的控制桩脚本 L_min最小周期 L_max最大周期 CaseNo用例号 Enable是否启用 1 or 0-->
<!--NanoMsg RefName="nanoNode1" 配置NanoNode的引用名称-->
<NodeMsg>
  <Scenario>
    <NanoNode RefName="nanoNode"/>
    <Item L_min="30" L_max="40" CaseNo="1" Enable="1"/>
  </Scenario>
</NodeMsg>
```

图7　桩配置文件

服务器端配置，如图 8 所示。

```
simulator_nanomsg.xml
<?xml version="1.0" encoding="utf-8"?>
<!--nanomsg survey,Type is "server" or "client", this configuration only has one node-->
<!--server id is 9999, client id is form 1 to 8999-->
<Survey>
  <Node Id="9999" Url="tcp://127.0.0.1:5555" Type="server"/>
</Survey>
```

图8　服务器端配置文件

模拟器使用的组件配置，如图 9 所示。

3.2　测试脚本

脚本文件，如图 10 所示。

```
simulator_feature.xml
<?xml version="1.0" encoding="utf-8"?>
<!-- 模拟器特性配置 -->
<!-- confile路径相对于simulator.py -->
<!-- ClassName className也可以全局函数名称 -->
<!-- 扩展OnMethod的方法时，必须包含一个参数，即将simulator的对象传进去 -->
<!-- ModuleFile 指定了模块的路径，若为空，则是simulator内置模块不用指定路径 -->
<FEATURE>
 <!--variable-->
 <Item Ref="variant" ClassName="VaribleDic" confile="./setting/simulator/simulator_variant.xml" OnMethod="" OffMethod="" ModuleFile=""/>
 <!--simulator node msg-->
 <Item Ref="scenario" ClassName="NodeMsg" confile="./setting/protocol/simulator_nodemsg.xml" OnMethod="startNodeMsg" OffMethod="" ModuleFile
 =""/>
 <!--nanomsg node-->
 <Item Ref="nanoNode" ClassName="createSurveyNode" confile="./setting/protocol/simulator_nanomsg.xml" OnMethod="nodeRun" OffMethod="
nodeStop" ModuleFile=""/>
 <!-- 以上为内置模块 -->
 <Item Ref="sacem_test" ClassName="PY_SACEM_TST" confile="./setting/extends/extends_sacem.xml" OnMethod="startSacem" OffMethod="stopSacem"
 ModuleFile="./extends/sacemtest.py"/>
</FEATURE>
```

图9　组件配置文件

```
sacem.robot
*** Settings ***
Documentation     Example test cases using the keyword-driven testing approach.
                  Example some by QYR
Library           sacemlibrary.SacemNode    sacem_test    1    0.5

*** Test Cases ***
01
    caseConfiguration    ${CURDIR}
    deviceInit           ./setting/simulator/simulator_feature.xml
    runSacemExe
    sleepSeconds         5
    deviceRun
    sleepSeconds         15
    pushCaseInfoFromXml
    HeartBeat            60
    deviceEnd
    sleepSeconds
    resultShallBe        ${CURDIR}
```

图10　脚本文件示例

脚本中所使用的关键字是测试人员自己定义的 key word 函数，测试人员只需要知道测试平台开放的接口及需要的参数，就可以实现编写脚本，而不需要关心底层是怎么实现的，从而能够实现封装作用。测试人员也可以编写多个脚本，放置在指定目录下，批量执行，但是这些驱动函数是需要测试人员自己创建所需要的 library。

3.3　建立Survey通信模型

如服务器端配置所示，在模拟器端创建一个 Server 服务器，在被测对象中创建 Client 端，从而实现模拟器与被测对象之间 Survey 模型通信。由于 Client 端是接收到 Server 端发送的消息才会回复消息，Server 端需要发送心跳包，以收集下位机产生的日志，也防护下位机因为队列满而导致存储日志消息失败。

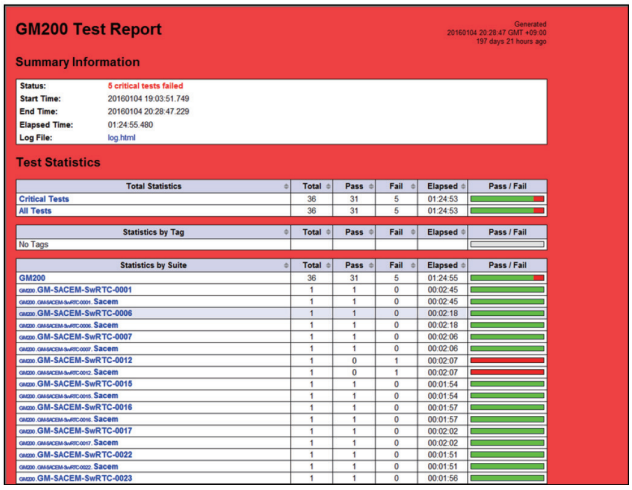
3.4　Protocol Buffer封装数据

采用 Google Protocol Buffer 数据存储格式，自定义数据结构，编译生成 pb 文件，按照 Protocol Buffer 封装数据的规则调用回调函数进行封装，并放入每个 case 对应的日志队列里，等待发给 Server 端，之后由 Server 端解析。

4　效果展示

运行脚本命令，测试平台就开始依次执行该目

录下的所有用例，并产生 report 和 log，如图 11 和图 12 中所示。



GM200 Test Report

Generated 20160104 20:28:47 GMT +09:00 197 days 21 hours ago

Summary Information

Status:	5 critical tests failed
Start Time:	20160104 19:03:51.749
End Time:	20160104 20:28:47.229
Elapsed Time:	01:24:55.480
Log File:	log.html

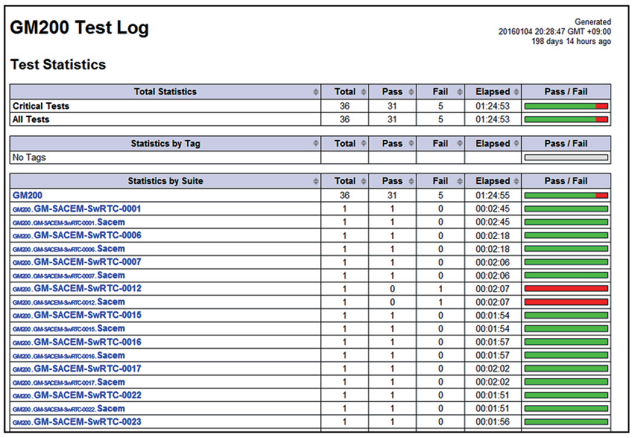
Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	36	31	5	01:24:53	
All Tests	36	31	5	01:24:53	

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
GM200	36	31	5	01:24:55	
GM200.GM-SACEM-SwRTC-0001	1	1	0	00:02:45	
GM200.GM-SACEM-SwRTC-0001.Sacem	1	1	0	00:02:45	
GM200.GM-SACEM-SwRTC-0006	1	1	0	00:02:18	
GM200.GM-SACEM-SwRTC-0006.Sacem	1	1	0	00:02:18	
GM200.GM-SACEM-SwRTC-0007	1	1	0	00:02:06	
GM200.GM-SACEM-SwRTC-0007.Sacem	1	1	0	00:02:06	
GM200.GM-SACEM-SwRTC-0012	1	0	1	00:02:07	
GM200.GM-SACEM-SwRTC-0012.Sacem	1	0	1	00:02:07	
GM200.GM-SACEM-SwRTC-0015	1	1	0	00:01:54	
GM200.GM-SACEM-SwRTC-0015.Sacem	1	1	0	00:01:54	
GM200.GM-SACEM-SwRTC-0016	1	1	0	00:01:57	
GM200.GM-SACEM-SwRTC-0016.Sacem	1	1	0	00:01:57	
GM200.GM-SACEM-SwRTC-0017	1	1	0	00:02:02	
GM200.GM-SACEM-SwRTC-0017.Sacem	1	1	0	00:02:02	
GM200.GM-SACEM-SwRTC-0022	1	1	0	00:01:51	
GM200.GM-SACEM-SwRTC-0022.Sacem	1	1	0	00:01:51	
GM200.GM-SACEM-SwRTC-0023	1	1	0	00:01:56	

图11　测试报告文件



GM200 Test Log

Generated 20160104 20:28:47 GMT +09:00 198 days 14 hours ago

Test Statistics

Total Statistics	Total	Pass	Fail	Elapsed	Pass / Fail
Critical Tests	36	31	5	01:24:53	
All Tests	36	31	5	01:24:53	

Statistics by Tag	Total	Pass	Fail	Elapsed	Pass / Fail
No Tags					

Statistics by Suite	Total	Pass	Fail	Elapsed	Pass / Fail
GM200	36	31	5	01:24:55	
GM200.GM-SACEM-SwRTC-0001	1	1	0	00:02:45	
GM200.GM-SACEM-SwRTC-0001.Sacem	1	1	0	00:02:45	
GM200.GM-SACEM-SwRTC-0006	1	1	0	00:02:18	
GM200.GM-SACEM-SwRTC-0006.Sacem	1	1	0	00:02:18	
GM200.GM-SACEM-SwRTC-0007	1	1	0	00:02:06	
GM200.GM-SACEM-SwRTC-0007.Sacem	1	1	0	00:02:06	
GM200.GM-SACEM-SwRTC-0012	1	0	1	00:02:07	
GM200.GM-SACEM-SwRTC-0012.Sacem	1	0	1	00:02:07	
GM200.GM-SACEM-SwRTC-0015	1	1	0	00:01:54	
GM200.GM-SACEM-SwRTC-0015.Sacem	1	1	0	00:01:54	
GM200.GM-SACEM-SwRTC-0016	1	1	0	00:01:57	
GM200.GM-SACEM-SwRTC-0016.Sacem	1	1	0	00:01:57	
GM200.GM-SACEM-SwRTC-0017	1	1	0	00:02:02	
GM200.GM-SACEM-SwRTC-0017.Sacem	1	1	0	00:02:02	
GM200.GM-SACEM-SwRTC-0022	1	1	0	00:01:51	
GM200.GM-SACEM-SwRTC-0022.Sacem	1	1	0	00:01:51	
GM200.GM-SACEM-SwRTC-0023	1	1	0	00:01:56	

图12　测试日志文件

图 11 所示的 report 是本次批量执行用例的整体情况，其记录了测试结果，执行用例总数，Pass 和 Fail 用例数以及全部执行完用例耗时，每个用例的执行结果和耗时。此外，背景显示为红色，说明有用例 Fail，背景若为绿色，则说明全部 Pass。

图 12 所示本次批量执行用例的每个用例执行情况，包含该用例的执行结果、耗时以及每个用例执行的步骤。

5　结束语

在基于 Robot 框架的软件测试开发中，使用 python 和 C 作为开发语言，实现了跨语言编程。实际使用表明，该框架易于二次开发，配置和部署灵活，具有良好的扩展性和稳定性，有效地提高了测试平台的开发效率以及产品测试效率。

参考文献：

[1] 王金波，张　涛 . 基于故障注入的嵌入式软件安全性测试框架及实现 [J]. 计算机应用研究，2012，29（8）：2991-2995.

[2] 张　雪，杨春林 . 基于 python 的自动化测试框架在 Scrum 开发模式中的应用 [J]. 福建电脑，2010（8）：150-152.

[3] 刘振玉，任　军 . RSSP-II 安全通信协议软件的自动测试设计及实现 [J]. 铁路通信信号工程技术，2015，12（6）：44-47.

[4] 刘锦峰，欧阳敏 . 城轨嵌入式软件自动化测试框架的设计与实现 [J]. 铁路计算机应用，2015，24（7）：49-52.

[5] 张志敏 , 周四望 . 针对软件系统功能的自动化测试工具设计 [J]. 计算机系统应用，2010，19（6）：123-127.

[6] 杨清玉 , 李金丽 , 陈吉兰 , 等 . HTTP 接口自动化测试方法研究 [J]. 微型机与应用，2016，35（18）：22-25.

[7] 刘慕涛，张　磊，王　艳，等 . 基于 XML 的 API 自动化测试工具设计与实现 [J]. 计算机工程，2007，33（13）：96-98.

[8] 高　静，兰雨晴，金茂忠，等 . 一个基于 XML 的自动化类测试框架 [J]. 微型机与应用，2007，7（增刊）：201-205.

责任编辑　徐侃春