一种 TTCN-3 编解码器的演化方法

孙 晶 马媛媛
(北方工业大学计算机学院 北京 100144)

摘 要 软件需求常常发生变化,导致软件经常被演化。由于使用 TTCN-3 对演化后的软件进行测试时需要为大部分测试脚本重新开发编解码器,因此有必要对原来的编解码器进行演化,以提高测试效率。分析了 TTCN-3 编解码器的研究现状,提出了一种结合测试套文件对编解码器进行演化的方法。实验表明,该方法是可行的,为 TTCN-3 测试系统的使用提供了方便。

关键词 演化,编解码器,TTCN-3
中图法分类号 TP311 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.04.028

Evolution Method for TTCN-3 Codec

SUN Jing MA Yuan-yuan
(School of Computer Science and Technology, North China University of Technology, Beijing 100144, China)

Abstract Software requirements' frequent change always leads to the evolution of software. When using TTCN-3 to test the software which has been evolved, we usually need to redevelop the codec for most test suite. Therefore, it is necessary to automate the evolution of the original codec in order to improve the test efficiency. In this paper, we analyzed the state-of-the-art of TTCN-3 codec, and proposed a new method of evolving codec by combining the test suite. Experiments demonstrate that this method is feasible, and it is convenient for the TTCN-3 test system.

Keywords Evolution, Codec, TTCN-3

1 引言

软件系统在其生命周期中会面临各种各样的变化。Lehman 和 Peryy 等人通过分析软件的变化,提出当变化发生时软件的两种发展趋势:1)逐渐失去其原有价值;2)不停地改变自身以适应各种变化,持续发挥其价值[1]。软件通过一定的改变达到所希望形态的过程被称为软件演化[2]。TTCN 测试工具软件能够采用专门用于测试的 TTCN 语言来辅助测试人员自动化地完成测试工作。TTCN-3 是由 ETSI 设计的专门用于测试的标准抽象测试语言[3]。但是,TTCN-3 在使用中需要结合相应的编解码器和适配器,其中编解码器的开发至关重要,每一个测试数据都需要经过编解码才能实现测试系统与被测系统的交互。因而开发编解码器的工作量很大[4],不同的被测系统需要的编解码规则也有很大不同,即使同一个被测系统进行演化后原编解码器也可能不再适用。

Vaclav 等人把软件演化看作是为了适应变化的需求持续对程序进行调节的过程,即不断对软件功能进行添加、删除和修改,对软件结构进行完善,进而使软件能够满足新需求的过程[5]。在对演化后的软件进行测试时,原来的测试数据可能不再适用,需要加以修改。测试数据改变后会导致原来的编解码器不再适用,此时就需要为测试系统开发新的编解码器。当测试系统的规模逐渐增大或者测试人员发生变动时,测试系统管理和维护的难度也会越来越大。为了减小测试成本,提高测试效率,学者们对编解码器的开发进行了大量的研究。柳永坡等人设计了一个对 Web 应用软件进行测试的通用编解码器方案,但是他们没有给出具体的实现过程[6]。韩正伟设计了一种 TTCN-3 通用解码器模型,通过设计一个通用的解码器规则文件,进而设计出解码器的整体框架[7]。宁娇提出了一种 TTCN-3 通用编码器模型,其能够对基本类型和简单的结构类型编码[8]。尚思超设计并实现了一种用于测试手机网络应用软件的通用编解码器[9]。Li W H 等人提出了一种通过分析包含 TTCN-3 测试数据定义的 XML 文件进而生成编解码器的方法,但是其要求手动编写 XML 文件,这样在无形中增加了测试成本[10]。

本文从演化的角度研究 TTCN-3 编解码器,提出了一种根据被测系统演化之后的测试套文件对原编解码器进行演化的方法。该方法通过分析抽象测试套文件中定义的测试数据对原来的编解码器进行演化,使演化后的编解码器可以应用于演化之后的软件测试中,从而节省测试成本,提高测试效率。

到稿日期:2015-11-30 返修日期:2016-02-19 本文受北方工业大学优势学科项目资助。

孙 晶(1968-),女,硕士,副教授,主要研究方向为大数据分析与云计算、软件测试与检验,E-mail:sunjing8248@163.com;马媛媛(1989-),女,硕士生,主要研究方向为软件演化。

2 相关概念介绍

2.1 TTCN-3 数据类型

TTCN-3 语言包含基本数据类型和结构数据类型。其中整形、浮点类型、字符型和基本串类型等都是与程序设计语言中的数据类型可以相互映射的基本类型[3]。当然，TTCN-3 也支持一些特殊的基本数据类型，如端口、地址和对象表示等。TTCN-3 中的结构数据类型可以通过基本数据类型构造而成，使用关键字 type 来定义。

2.2 编解码器

TTCN-3 测试系统对软件进行测试时，由编解码器对测试数据进行编码。编解码器使用 Encode 函数根据数据类型和相应的编码规则把数据编码成被测系统可以识别的类型；然后通过系统适配器传送编码后的数据给被测系统；数据经被测系统处理后再通过系统适配器向测试系统返回测试结果，测试结果仍需编解码器进行解码操作。编解码器中的 Decode 方法负责将数据解码成 TTCN-3 语言能识别的数据类型。编解码器在 TTCN 测试工具软件中的作用至关重要。

3 TTCN-3 编解码器的演化方案

虽然很多学者在通用编解码器的中取得了一定的研究成果，但是始终没有一个可以适用于任何测试系统的编解码器，主要原因在于 TTCN-3 测试套中表示的测试数据千变万化，并且测试数据根据不同的被测系统也具有不同的现实意义即不同的编码规则。一个软件在其生命周期中会经过很多次的测试、修改、再测试的过程。对某一个功能进行测试的编解码器在软件修改后，即使是再对同一功能进行测试，它的编解码器也不一定能继续使用，此时需要重新开发编解码器。本文希望能对原来的编解码器进行演化，使演化后的编解码器可以继续用于测试演化后的被测系统。编解码器演化方案的总体设计如图 1 所示。

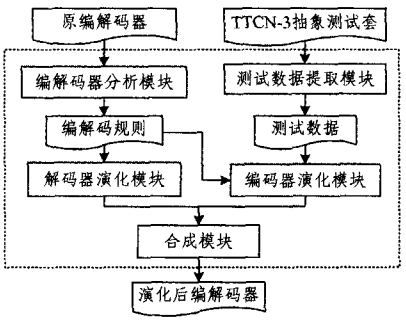

图 1 编解码器演化方案的总体设计

从图 1 可以看出，该演化方案中输入的不仅仅有原编解码器，还有 TTCN-3 抽象测试套文件。通过分析原编解码器文件，得出编解码器原来的编解码规则。软件进行演化后会为其设计新的测试用例和测试数据，但是其测试数据的编码规则一般不会改变。例如，为了确保安全，为某一网站的登录增加了验证码功能。那么对该功能进行测试时，抽象测试套中定义的测试数据类型发生了变化，原来的编解码器不再适用，但是测试数据的编码规则并不改变。利用这一特点，根据原编解码器中的编码规则并结合新的测试数据类型对原编解码器进行演化。由于数据信息都定义在 TTCN-3 语言编写的抽象测试套中，因此新的数据可以从中提取。TTCN-3 抽象测试套的语法规则与高级程序设计语言类似，数据类型以及数据模板定义时都有特定的格式，通过分析抽象测试套，获取其中的测试数据，并将其存储在 XML 文件中。编码器演化模块和解码器演化模块都需要读取存有测试数据的 XML 文件和分析通过原编解码器得到的编解码规则。编解码器演化时根据测试数据和编码规则得到新的编解码器。编解码器演化的具体流程如下：

1)数据提取模块负责从抽象测试套中提取数据；

2)编解码器分析模块读取原编解码器，分析其中的相关规则；

3)编码器演化模块读取编码规则和测试数据文件，从测试数据文件中提取抽象测试套中向被测系统发送的数据，根据数据信息和相应规则得到演化后的编码器；

4)解码器演化模块读取编码规则和测试数据文件，从测试数据文件中提取抽象测试套中要接收的被测系统返回的数据，根据数据信息和相应规则得到演化后的解码器；

5)合成模块负责组合前面得到的编码器和解码器；

6)输出演化后的编解码器。

4 TTCN-3 编解码器的演化实现

4.1 测试数据的提取

从以上演化方案可知，本算法需要从抽象测试套中提取测试数据。根据 TTCN-3 语言的语法规则，分析抽象测试套文件，找出其中定义的所有数据并保存到 XML 文件中。将定义好的每个数据分别作为 XML 中的节点保存起来。XML 文档结构的设计主要分为 3 个部分：1) TemplateList，用来保存抽象测试套中所有定义好的 Template。对于每个 Template 节点中的子节点，若其取名为 T1Child，则代表它是 Template 节点的第一层子节点，如果 T1Child 还有子节点，则命名为 T2Child，以此类推，Template 的第 n 层孩子命名为 TnChild。2) SendList，用来保存要发送给被测系统的测试数据，其结构与 Template 节点的一致。3) ReceiveList，用来保存期望从被测系统接收的测试数据。提取数据的方法并不复杂，首先读取 TTCN-3 抽象测试套文件，然后对其进行逐行扫描，当发现“template”关键字时，说明有一个 Template 的定义，截取这个 Template，将其作为 XML 文件中的 TemplateList 中的 Template 节点保存起来，并根据截取内容为每个 Template 节点设置“name”，“type”和“isBasicType”属性。当发现“send”或者“call”关键字时，说明有要发送的数据，将其截取出来作为 XML 文件的 SendList 的节点保存起来。当发现有“receive”关键字时，说明有要接收的数据，将其截取出来作为 XML 文件中的 ReceiveList 中的节点保存起来。提取测试数据算法的伪代码如算法 1 所示。

算法 1 Extracting Test Data Algorithm

输入：(AstFile)//抽象测试套文件

输出：XmlFile//存储测试数据的 XML 文件

1. 新建 XML 文件的内容文档变量 Doc，并设置其格式；
2. 按行读取 AstFile，并把每行内容赋给 TempString；
3. while AstFile 文件没有读完
4. If TempString 包含“template” then do //提取 template
5. 在 Doc 中创建节点 Parent 并截取 TempString 中 template 的

类型和数据名称等信息,将其设置为 Parent 的属性;
6. If TempString 包含"{" then do
7. "{"入栈;
8. end If
9. while 栈非空
10. 读取 AstFile 的下一行,并赋给 TempString;
11. If TempString 包含"{" then do //提取子域
12. "{"入栈;
13. 创建 Parent 的子节点 Child,并把 Child 赋给 Parent;
14. Else TempString 中包含字符"}" then do
15. "{"出栈;
16. 将 Parent 的上层节点赋值给 Parent;
17. Else then do
18. 创建 Parent 的子节点 Child 并截取 TempString 中 template 的相关信息,设置为 Parent 的属性;
19. end If
20. end while
21. Else TempString 包含". send" then do //提取发送数据
22. 在 Doc 中创建代表接收数据的节点 Send 并截取 TempString 中的类型和数据名称信息,将其设置为 Send 的属性;
23. Else TempString 包含". receive" then do//提取接收数据
24. 在 Doc 中创建代表接收数据的节点 Receive 并截取 TempString 中的类型和数据名称信息,将其设置为 Receive 的属性;
25. end If
26. end while
27. 创建 XmlFile 文件,并将 Doc 写入文件,输出;

4.2 编解码器的演化

由于软件在开发或者演化阶段需要不断进行修改,这些修改常常导致测试数据发生变化,进而导致用于测试的编解码器不再可用,这样的编解码器称为不可用编解码器;而仍可以继续使用的编解码器则称为可用编解码器。在编解码器演化时先判断原编解码器是否可用,然后做不同的处理。编解码器演化主要是修改不可用编解码器,使其在修改后可以继续应用于演化后软件测试工作。软件的每个功能的修改都可能导致编解码器不可用,故编解码器的修改需要花费测试人员很多的时间和精力。软件演化后所使用的编解码器和演化前其编解码规则发生变化的概率很小,根据被测系统输入数据的改变进行修改比重新编写编解码器更容易、方便。因此编解码器的演化对于演化软件的测试来说具有重大意义。这里把不可用编解码器分为可以演化的编解码器和不可演化的编解码器。不可演化是指若要演化,则需要付出过高的代价,比如编解码规则已经改变等。这种编解码器直接返回演化失败。编解码器的演化流程如图 2 所示。

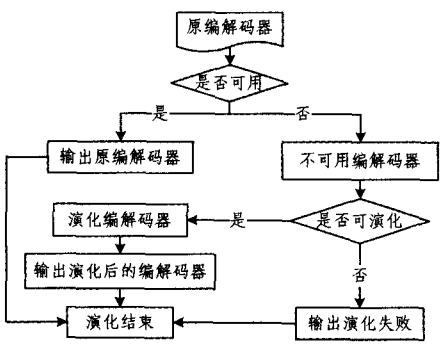

图 2　编解码器的演化流程图

由图 2 可知,对原编解码器进行演化之前需先判断它是否可用,原编解码器是否可用主要取决于测试脚本中定义的测试数据类型是否发生变化,故要判断原编解码器是否可用,只需要判断测试数据类型是否改变即可。不可用编解码器是否可演化主要取决于编码规则是否改变,而编码规则是由被测系统的数据传输协议决定的。所以要判断原编解码器是否可演化,只需要判断使用的协议是否改变即可。这些判断非常简单,这里不再给出具体的算法,而把重点放在编解码器的演化上。

编解码器演化时,从原编解码器中截取编解码器的头部信息;然后从 XML 测试数据文件中取出要编码或解码的数据,判断它是否为基本类型,如果是基本类型,则直接进行编码或解码,如果不是基本类型,则运用递归的思想为其每一个子域进行编码或解码;最后再根据原编解码器中分析得到的编解码规则将编码或解码的数据进行组合,得到演化后的编解码器。

关于编码部分,从原编解码器中取得头部信息和编码后数据的合成规则,将其运用到新的编码器中,基本类型直接转换为字节流,结构数据类型先将各个部分合成,再将合成后的数据转换成字节流,从而完成编码部分。编码器的演化算法如算法 2 所示。

算法 2　Evolution of the encoder

输入:(Codec,XMLFile)//编解码器文件,测试数据文件
输出:Encode//演化后的编码器
1. 读入 XMLFile 文件,查找要编码的数据节点并将它赋值给 Data;
2. If Date 为基本类型 then do//编码基本类型数据
3. 根据 Data 的类型信息为其编码,并追加到 Encoder 尾部;
4. 从 Codec 中获取数据之间的分隔符,并追加到 Encoder 尾部;
5. Else then do
6. 获取 Data 的所有子节点并将它赋给 Children;
7. while Children 还有子节点
8. 获取 Children 的子节点赋给 C;
9. 为 C 进行递归编码;
10. end while
11. end If
12. output Encoder

解码器的演化思想和编码器的基本一致,解码器的演化主要是对 XML 中存储的待解码数据进行处理,这里不再赘述。

5 演化实例

本文对适配器库管理系统[11]测试时使用的编解码器进行演化。该系统主要用于组织管理适配器,用户开发的适配器可以通过该系统上传到适配器库中,需要时也可以从适配器库中下载,管理系统提供了用户登录、注册及适配器的分类、检索等功能。完成该系统后对其进行了一定的测试,保存了测试时的测试脚本和编解码器。由于该系统还存在很多问题(比如登录时安全性低、适配器描述的信息不完整等),需要进行演化。系统演化后再进行测试时很多编解码器都需要重新设计。对其原来的编解码器进行演化,然后将其用于演化后适配器库管理系统的测试。由于该适配器库管理系统在演化前后使用的协议是没有改变的,因此按照本文的思想,该系

统中所有的编解码器都是可以演化的。对于数据类型没有改变的演化，都是直接输出原编解码器，非常简单。故这里只给出数据类型发生改变的演化，并将演化结果在测试系统中进行应用实验。

下面以该适配器管理系统中的添加用户功能为例对实验进行说明。对于添加用户功能，原来只是添加用户的用户名和密码，系统改进之后，把用户信息进行细化，在原来的基础上增加了真实姓名、电话和用户权限信息等，故其编解码器需要被演化。编解码器演化及演化后的编解码器在测试系统中应用的实验流程如图 3 所示。

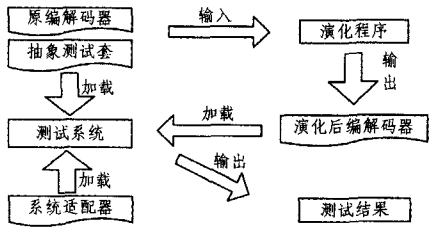

图 3 实验流程图

系统演化后使用的抽象测试套中的测试数据定义部分的设计如下。

```
type record Request{//请求数据类型定义
    RequestLine requestLine,
    RequestHeader requestHeader optional,
    GeneralHeader generalHeader optional,
    EntityHeader entityHeader optional,
    MessageBody messageBody optional
}
type record Response{//响应数据类型定义
    StatusLine statusLine,
    ResponseHeader responseHeader optional,
    GeneralHeader generalHeader optional,
    EntityHeader entityHeader optional,
    EntityBody entityBody optional
}
```

请求与响应数据类型定义中的每个子域也有详细的定义，限于篇幅，这里省去具体的细节定义，直接给出具体的 template 的定义。

```
template Request req_1:= {//发送数据模板
  requestLine:={
    method:=GET,
    requestUri:={
      protocol:="http://",
      hostName:="localhost",
      portNumber:=80,
      path:="testadapterlib/cn.edu.ncut.login.action"
    },
      version:="HTTP/1.1"
  }
  messageBody:={
    username:="zhangsan",
    password:="123",
    confpassword:="123",
    realname:="zhangsan",
    phone:="13261516666",
    privi:="administrator"
  }
  template Response res_1:={//接收数据模板
  statusLine:={
    version:="HTTP/1.1",
    statusCode:=200,
    reasonPhrase:="OK"
  }
  entityBody:="success.html"
}
```

经过测试数据提取模块的处理，测试数据被保存成 XML 文件中节点的形式。下面以 template Request req_1 为例来展示 XML 文件中数据的形式。

```
〈Template name="req_1" type="Request" isBasicType="N"〉
 〈T1Child name="requestLine" isBasicType="N" type="Request-
 Line"〉
  〈T2Child name="method" isBasicType="Y" type="charstring"〉
  GET〈/T2Child〉〈T2Child name="requestUri" isBasicType="N"
  type="URL"〉
   〈T3Child name="protocol" isBasicType="Y" type="char-
   string"〉http://〈/T3Child〉
   〈T3Child name="hostName" isBasicType="Y" type="char-
   string"〉www.ncut.edu.cn〈/T3Child〉
   〈T3Child name="portNumber" isBasicType="Y" type="inte-
   ger"〉80〈/T3Child〉
   〈T3Child name="path" isBasicType="Y" type="charstring"〉/
   index.html〈/T3Child〉
  〈/T2Child〉
  〈T2Child name="version" isBasicType="Y" type="charstring"〉
  HTTP/1.1〈/T2Child〉
 〈/T1Child〉
〈/Template〉
```

此编解码器的演化结果如表 1 所列。

表 1 编解码器的演化结果

数据类型是否变化	协议及编解码规则是否变化	输入	输出	输出文件位置	备注
Y	N	/test.ttcn3 /codec.java	/test.xml /newcodec.java	D:\Decoder	演化成功

在 TTCN 测试工具软件中使用演化得到的编解码器对演化后的被测系统进行测试，得到的测试结果如表 2 所列。

表 2 测试结果

测试用例	输入	期望输出	实际输出	结果	备注
T1	req_1	res_1	res_1	pass	添加用户成功
T2	null	res_1	No Content	fail	添加用户失败

表 2 证明了演化后的编解码器的有效性。使用本文提出

（下转第 139 页）

[12] Virusshare[OL.]http://virusshare.com.

[13] Virustotal[OL]. https://www.virustotal.com.

[14] ZHANG R. Research on Malware Detecting based on Static Analysis under Android Environment [D]. Chongqing: Chongqing University, 2014. (in Chinese)
张锐. Android 环境下恶意软件静态检测方法研究[J]. 重庆：重庆大学，2014.

[15] ZHOU Y J, JIANG X X. Dissecting android malware: Characterization and evolution[C]// Proceedings of the IEEE Symposium on Security and Privacy. 2012: 1063-1069.

[16] XING L Y, PAN X R, WANG R, et al. Upgrading your android, elevating my malware: privilege escalation through mobile os updating[C]// Proceedings of the 35th IEEE Symposium on Security and Privacy. 2014: 393-408.

[17] PANDITA R, XIAO X S, YANG W, et al. Whyper: towards automating risk assessment of mobile applications[C]// Proceedings of the 22nd USENIX Conference on Security. 2013: 527-542.

[18] WERTHMANN T, HUND R, DAVI L, et al. Psios: bring your own privacy and security to ios devices[C]// Proceedings of the 8th ACM SIGSAC Symposium on Information, Computer and Communications Security. 2013: 13-24.

(上接第 134 页)

的演化方法将原来的编解码器演化成新编解码器的整个过程用时 23s，主要用时在于手动输入内容。如果重新开发编解码器，一个技能娴熟并且对测试数据非常了解的测试人员也需要至少 5min 的时间，而且开发出来的编解码器一般都会存在一些问题，需要经过多次调试后才能正常使用。编写和调试的耗时一般在 10～30min。当测试数据比较复杂时，还会花费更多的时间。假设一个软件系统中有 20 个功能因为演化导致原来的编解码器不可用，若使用本文方法，只需要 10min 的时间即可得到所有的可用编解码器，但是手动开发至少需要 200～300min，本文方法将速率提高了 20～30 倍。被测系统的规模越大，尤其是中途更换测试人员时，这种从原编解码器进行演化以替代重新开发的速度优势也越明显。

结束语　软件演化表明了软件适应各种变化的能力。本文提出了一种对 TTCN-3 编解码器进行演化的方法，并且对其中的演化算法进行了实现，最后对实现结果进行了验证。该方法将不能继续使用的编解码器自动演化成了可用的编解码器，为 TTCN 测试工具软件的使用提供了便利，节省了测试成本。但是本文的演化方法仍然存在不足之处，即演化后的编解码器程序不存在循环结构，对于结构相同且可以用循环语句表示的编解码程序，这里也需逐条写出，所以当测试套中定义的数据类型比较庞大时生成的编解码器会相对臃肿。调整编解码器的程序结构是下一步的研究目标。

参 考 文 献

[1] LEHMAN M M, PERYY D E, RAMIL J F. Metrics and Laws of Software Evolution-the Nineties Views[C]// Proceeding of 4th International Symposium on Software Metries, 2000. IEEE Computer Society Press, 2000: 20-32.

[2] MENS T, BUCKLEY J, ZENGER M, et al. Towards a Taxonomy of Software Evolution[C]// International Workshop on Unanticipated Software Evolution, 2003. Warsaw, Poland, 2003: 34-45.

[3] ETSI ES 201 873-1 v4.5.1. Methods for Testing and Specification (MTS); The Testing and Test Control Notation version 3; Part 1: TTCN-3 Core Language[S]. 2013.

[4] SUN M, ZOU F Y, MA J, et al. A method to automatically generate codec based TTCN-3 to software testing[C]// IEEE 2nd International Conference on Communication Software and Networks, 2010. Singapore: IEEE Press, 2010: 200-204.

[5] RAJLICH V, SILVA J H. Evolution and reuse of orthogonal architecture[J]. IEEE Transactions on Software Engineering, 1996, 22(2): 153-157.

[6] LIU Y P, WU J, LIU S M. Research of Generic Codec for Web Application Testing[J]. Computer Science, 2013, 40(8): 157-160. (in Chinese)
柳永坡，吴际，刘霜梅. Web 应用测试的通用编解码器研究[J]. 计算机科学，2013，40(8)：157-160.

[7] HAN Z W. The Research and Implementation of Universal Decoder Based on TTCN-3[D]. Beijing: Beijing University of Posts and Telecommunications, 2010. (in Chinese)
韩正伟. 基于 TTCN-3 通用解码器的研究与实现[D]. 北京：北京邮电大学，2010.

[8] NING J. The Research and Implementation of Universal Encoder Based on TTCN-3[D]. Beijing: Beijing University of Posts and Telecommunications, 2010. (in Chinese)
宁娇. 基于 TTCN-3 的通用编码器的研究与实现[D]. 北京：北京邮电大学，2010.

[9] SHANG S C. Study on Implementation Technology of Mobile Network Application Software Emulation Test Environment [D]. Beijing: North China University of Technology, 2008. (in Chinese)
尚思超. 手机网络应用软件仿真测试环境开发技术研究[D]. 北京：北方工业大学，2008.

[10] LI W H, MA Y, HUANG X H. Common TTCN-3 Codec to reduce test engineering cost[C]// IET 6th Proceedings of Advanced Intelligence and Awareness Internet, 2010. Ayia Napa: IEEE Press, 2010: 333-336.

[11] ZHANG B. Design and Implementation of TTCN-3 Adapter Library Management System[D]. Beijing: North China University of Technology, 2013. (in Chinese)
张暴. TTCN-3 适配器库管理系统的设计与实现[D]. 北京：北方工业大学，2013.