基于知识提取的测试文档自动生成

孙晓韩,李 宁,于 杨,杨子仪,张 林
(北京信息科技大学 网络文化与数字传播北京市重点实验室,北京 100101)

摘 要:测试文档是产品的重要组成部分,与测试数据紧密相关,且有严格的编制要求。针对以往测试文档编写中存在的重复繁琐、灵活性不高、效率低下等问题,运用数据到文本的生成理论,该文提出了一种基于知识提取的测试文档生成方法。该方法首先对原始的测试记录数据进行分析和理解,重点进行电子表格的表头识别和单元格关联关系识别,抽取出表格数据的逻辑关系;再根据规范化的测试本体,转换成规范的知识表达,形成测试知识的结构化表示并记录于测试知识库中;最后通过局部文档模板的填充,逐步形成整体的测试文档。该方法解决了灵活多变的测试记录导致的软件重复开发的一系列问题,有利于测试结果的有效利用。此外,自底向上的文档构造方式有助于按需展现内容,提高测试文档的可读性。

关键词:文档生成;知识提取;表格识别;计算机辅助写作;软件测试
中图分类号:TP319 **文献标识码**:A **文章编号**:1673-629X(2022)12-0110-07
doi:10.3969/j.issn.1673-629X.2022.12.017

Automatic Generation of Test Documents Based on Knowledge Extraction

SUN Xiao-han,LI Ning,YU Yang,YANG Zi-yi,ZHANG Lin
(Beijing Key Laboratory of Internet Culture and Digital Dissemination Research,Beijing Information Science and Technology University,Beijing 100101,China)

Abstract:The test document is an important part of the product,which is closely related to the test data and has strict compilation requirements. In order to solve the problems of repetition and triviality,low flexibility and low efficiency in the preparation of test documents in the past,we propose a test document generation method based on knowledge extraction by using the theory of data-to-text generation. This method firstly analyzes the original test data,focusing on the spreadsheet header recognition and cell relationship recognition,extracting the logical data. Then,according to the normalized ontology,the data is transformed into the standardized knowledge expression,forming the structured representation of test knowledge and recording it in the test knowledge base. Finally,the whole test document is gradually formed by filling each basic document template. This method solves the problems of repeated software development caused by flexible and changeable test records and is beneficial to the effective reuse of test results. In addition,the bottom-up document construction helps to present content on demand and improve the readability of test documents.

Key words:document generation;knowledge extraction;form recognition;computer aided writing;software testing

0 引 言

产品测试文档是一类特殊的文档,它记录产品测试活动留下的记录,并作为交付物,供产品的相关方使用,是产品的重要组成部分。这类文档的特点是与测试项目的数据有紧密联系,种类繁多内容复杂。此外,这类文档有严格的编制要求,存在大量的测试规范和文档编制规范[1-2]。文献[3]总结了软件文档编制的四方面要求:(1)准确性:产品文档描述的内容要与实际情况保持一致,并反映产品的最新更新;(2)完整性:产品文档中能够系统、详尽地描述产品所提供的全部功能;(3)及时性:产品文档可随时按需获得;(4)可用性:文档可以指导目标用户高效地完成工作。这些要求对于其他种类的文档编制也是必要的。

文档的计算机辅助编制技术,特别是软件产品的文档编制技术研究已有很长的历史[4-5]。以往的计算机辅助软件文档编写手段主要有:(1)通过办公软件、

收稿日期:2021-11-23 **修回日期**:2022-03-23
基金项目:国家重点研发计划(2018YFB1004100);国家自然科学基金项面上项目(61672105)
作者简介:孙晓韩(1996-),女,硕士研究生,研究方向为文档信息处理;通讯作者:李 宁,博士,教授,CCF 高级会员(E20-0008144S),研究方向为文档信息处理、XML、信息技术标准化。

Dita[6]等写作工具或手工编写或填充模板;(2)通过MarkDown[7]、reStructedText[8]等工具辅助,在软件开发过程中对代码进行标记,自动化地或半自动化地生成文档;(3)使用软件工具自动记录数据并生成文档;(4)开发专用的文档编写软件。这些文档编制方法都存在各自的优缺点。

首先第一种方法编写的文档可读性较好,但是主要依靠人工,费时费力,难以做到与系统同步更新;第二种方法能够做到与系统同步更新,但是需要进行大量的标注,带来额外的负担,且生成的文档类型有限,可读性较差;第三种方法自动化程度较高,但需要完善的软件工程环境的支持,一般工具形成的文档难以做到规范和全面,且难以进行灵活定制;第四种方法需要为不同的项目开发文档系统,而今天人们更希望以"低代码"的方式适应产品的变化。此外,当前大部分文档都是静态的,难以按需展现数据内容或实现交互式的信息检索。

当前随着软硬件开发技术的快速演进,以往的文档编制过程也显得低效而难以适应需要。以测试文档为例,软件测试是软件工程的重要环节,其主要产物是测试文档。测试文档要反映测试过程的活动,包括测试计划、测试需求、测试流程、缺陷跟踪和记录,等等。特别是大型软件测试经常包含成千上万个测试用例,记录这些测试案例和测试结果,使得测试文档的编写工作量很大。另一方面,敏捷开发方法和 DevOps[9]的应用,强调持续集成(Continuous Integration,CI)和持续交付(Continuous Delivery,CD),使得传统的测试文档编制过程难以适应,一方面频繁的迭代和更新导致文档延迟交付,另一方面找不到掌握全面信息的文档编制人员,文档的开发成为了影响项目进展的瓶颈。

随着近年人工智能和自然语言处理技术的发展,计算机辅助写作技术逐渐成熟。广义的计算机辅助写作(Computer Aided Writing, CAW)以数据和知识为基础,将知识管理与认知技术相结合,辅助人工完成文档的编写[10]。在 CAW 中,数据到文本的生成(Data to Text,D2T)技术与该文关系较大。目前该领域已经取得了较多的研究进展,业界已经研制出面向不同领域和应用的多个生成系统,如基于数值数据生成天气预报文本[11]、体育新闻[12]、财经报道、医疗报告等[13]。数据到文本的生成技术一般包括以下阶段:数据的分析和理解,文本的规划以及遣词造句[14]。

测试文档的编写也可以看作是从测试记录和测试数据生成测试报告文本的过程。数据到文本的生成技术可以为之提供很好的借鉴。例如,可以基于测试规范构建测试过程的知识本体,通过对测试数据的分析,将其映射到知识本体,形成知识表达,然后借助测试文档模板和内容生成规则,最终生成测试文档。此外,可以借助 CAW 的动态模板技术和复合出版[15]的特点,在遵循测试文档编写规范的前提下,根据数据的形态个性化地展现报告内容。例如,对于重复的测试项,自动形成子章节内容,对于统计数据,根据需要展现成表格和图表,或借助智能标记和插件对测试结果进行检索。

综上,针对文档编制过程中的低效问题,该文按上述思路提出一种基于知识提取的测试文档自动生成方法,其可以实现测试文档的自动生成,有效提升了现实中软件产品测试的智能性、可复用性以及自动化程度。

1　测试文档生成方法及模型设计

该文的目标是为计算机产品的质量测试开发测试数据管理与测试文档生成系统。希望能够支持多类软件的测试,能够对测试结果进行语义检索,能够跨平台运行,能够保持测试文档的实时更新。经过调研得知,在计算机产品的质量测试中,一个被测系统一般由多部分子系统构成,每个部分均按多个测试项、多个指标和多轮次测试开展测试,有一套预先设计的电子表格文档用于记录测试数据,不同类型的被测产品对应不同的电子表格记录。一个中等规模的测试项目大约包含 50 张电子表格,500 多项测试数据,一个复杂系统的数据量会数倍于它。可见数据的规模和复杂程度是很高的。一个典型的记录测试数据的电子表格局部如图 1 所示。

时间测试			
测试项	结果记录(单位:秒)		
	第1次	第2次	第3次
开机时间	35.6	34.62	35.67
操作系统启动时间	34.513	34.797	35.021
性能测试			
测试项	结果记录		
	速度基准	速率基准	
整型性能	19.5	88.7	
浮点性能	22.1	61.6	

图 1　一个典型的记录测试数据的电子表格(局部)

一般的测试文档自动生成系统会简单建立电子表格单元格到测试文档模板中标识项的映射,通过模板填充生成测试文档[16]。但是这样处理会存在多个问题:(1)无法对测试数据进行有效的管理和后续利用,例如对测试结果进行动态查询;(2)深度依赖被测的产品的种类,如果换一种产品则需要重新编程,无法达到"低代码"的目标;(3)缺少灵活性,记录测试数据的电子表格一旦结构发生变化,将需要重建映射。

笔者认为,要改进上述问题,需要一种有效的方式来管理测试数据。由于测试数据记录与具体的被测产

品相关,灵活多变,要将其体现在标准的测试文档内容中,构建的流程如下:

第一步:对不同类型产品的测试数据进行分析和知识提取,得到测试结果的结构化语义表示;第二步:将提取后的测试结果结构化表示映射到通用的测试本体,形成稳定、通用的知识表示并加以存储;第三步:构造知识本体到测试文档的映射关系,将测试结果通过模板填充或格式转换形成测试文档。其中,第一步对应 D2T 的数据理解过程;第三步对应 D2T 的文本规划和遣词造句过程;其中第二步是该文特有的。在自然语言处理中,D2T 生成的是自由的文本,不需要特定的结构和格式约束,而测试文档需要符合相关的编制规范,对于不同的产品测试类型,测试文档的撰写方式是基本一致的,因此需要对测试结果进行规范化的表示并保存到测试知识库中。

为方便后文表述,p_c 为类型 c 的软件测试项目,S_p 为项目 p 的电子表格形式的测试记录,K_c^p 为从 p_c 提取出的知识,按测试本体 O 规范化后表示为 K_p。测试文档生成过程模型如图 2 和图 3 所示。

时间测试			
测试项	结果记录(单位:秒)		
	第1次	第2次	第3次
开机时间	35.6	34.62	35.67
操作系统启动时间	34.513	34.797	35.021
性能测试			
测试项	结果记录		
	速度基准	速率基准	
整型性能	19.5	88.7	
浮点性能	22.1	61.6	

电子表格形式的测试记录S_P

测试数据理解

```
<Tables>
  <Table 列数="4" 行数="4">
    <Cell Paragraph="测试项" Type="Table">
      <Cell Paragraph="结果记录(单位:秒)" Type="indication">
        <Cell Paragraph="第 1 次" Type="indication">
          <Cell Paragraph="35.6" Type="entry"/>
          <Cell Paragraph="34.513" Type="entry"/>
        </Cell>
        <Cell Paragraph="第 2 次" Type="indication">
          <Cell Paragraph="34.62" Type="entry"/>
          <Cell Paragraph="34.797" Type="entry"/>
        </Cell>
        <Cell Paragraph="第 3 次" Type="indication">
          <Cell Paragraph="35.67" Type="entry"/>
          <Cell Paragraph="35.021" Type="entry"/>
        </Cell>
      </Cell>
      <Cell Paragraph="开机时间" Type="indication">
        <Cell Paragraph="35.6" Type="entry"/>
        <Cell Paragraph="34.62" Type="entry"/>
        <Cell Paragraph="35.67" Type="entry"/>
      </Cell>
      <Cell Paragraph="操作系统启动时间" Type="indication">
        <Cell Paragraph="34.513" Type="entry"/>
        <Cell Paragraph="34.797" Type="entry"/>
        <Cell Paragraph="35.021" Type="entry"/>
      </Cell>
    </Cell>
  </Table>
</Tables>
```

提取的知识K_P^C

图 2 测试文档生成过程(1)

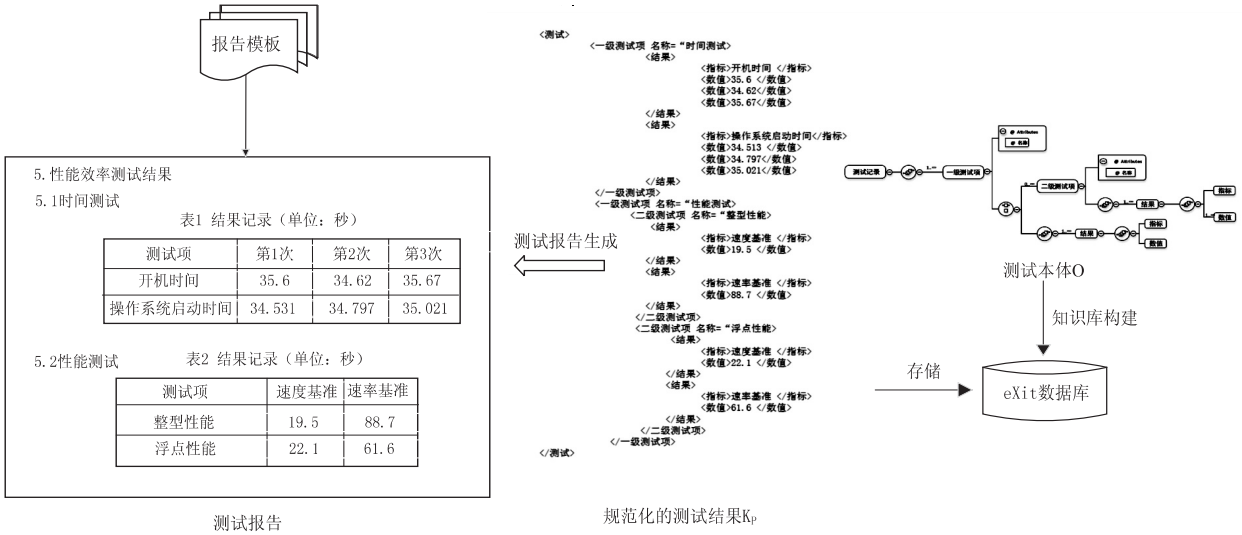


图 3 软件文档生成过程(2)

2 测试数据理解与知识提取

前文提到,简单建立记录测试数据的电子表格单元格到软件测试文档模板中标识项的映射会有很多弊端,需要对测试数据进行分析、理解。按第 1 节中的定义,需要从 S_p 得到 K_c^p。

大量的电子表格中的测试数据是以非关系型的方式记录的,例如,图 1 中有多个表格嵌套,不符合关系型范式。因此需要与之相适合的层次模型表示数据及其关系,XML 数据显然是最好的选择。从自由设计的电子表格中提取出 XML 并非易事。例如,图 1 的电子表格并没有明确的与 XML 数据的对应关系。为此需要进行表格逻辑结构的识别,以获得测试数据的知识表示。

设计了以下的表格逻辑结构识别算法。

从逻辑结构来看,表格模板是由单元格集合 C 和

单元格间关联关系集合 R 组成的一种数据存储结构。其中单元格集合 C 为:

$$C = \{c \mid c \in T_1 \cup T_2\} \tag{1}$$

式中, $T_1 = \{\text{Header}\}$, $T_2 = \{\text{Data}, \text{HeaderData}\}$。Data 表示数据型单元格,在表格模板中, Data 不含内容。Header 表示表头型单元格,它是一种只含表头信息而不含数据信息的单元格。HeaderData 表示表头数据型单元格,它既含表头信息,又能被填入数据。

单元格间关联关系分为两种:层次关系(表示为 Hie)和指示关系(表示为 Ind)。层次关系集合 H 为:

$$H = \{(c_\alpha, c_\beta) \mid (c_\alpha, c_\beta) \in T_1, c_\alpha \xrightarrow{\text{Hie}} c_\beta\} \tag{2}$$

层次关系是 Header 与 Header 间的关系,它表明了表头的级别。例如,图 4 中存在 $c_A \xrightarrow{\text{Hie}} c_B$,它表示单元格 c_A 的表头级别比单元格 c_B 高。指示关系集合 I 为:

$$I = \{(c_\alpha, c_\beta) \mid c_\alpha \in T_1, c_\beta \in T_2, c_\alpha \xrightarrow{\text{Ind}} c_\beta\} \tag{3}$$

指示关系是 Header 与 Data(或 HeaderData)间的关系。例如,图 4 中存在 $c_B \xrightarrow{Ind} c_C$,它表明单元格 c_B 是单元格 c_C 的表头。关联关系集合 R 为:

$$R = H \cup I \tag{4}$$

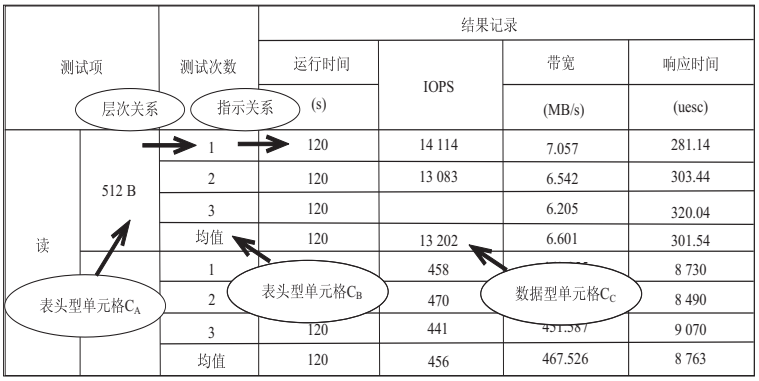


图 4　表格模板示例

值得注意的是,关联关系集合 R 中的元素均表示单元格间直接关联。例如存在 $c_A \xrightarrow{\text{Hie}} c_B$, $c_B \xrightarrow{\text{Ind}} c_C$,则产生的关联关系为 $c_A \xrightarrow{\text{Hie}} c_B \xrightarrow{\text{Ind}} c_C$, (c_A, c_C) 并不存在于集合 R 中。None 表示单元格间没有关系,表示为:

$$N = \{(c_\alpha, c_\beta) \mid c_\alpha, c_\beta \in T_1 \cup T_2, c_\alpha \xrightarrow{\text{None}} c_\beta\} \tag{5}$$

如果所有单元格两两不重复组合形成的集合为 P,则:

$$P = R \cup N \tag{6}$$

由此可知,若要提取表格的逻辑结构,必须要确定表格的单元格集合 C 和单元格间关联关系集合 R。其中, C 的确立依赖于表头型单元格集合 T_1、其他单元格集合 T_2 的确立,即表头型单元格的识别。并且,关联关系集合 R 是在已知 T_1、T_2 集合的基础上建立的,故表头识别是逻辑结构识别的首要任务。

该文构造了一种全连接神经网络分类器来区分 Header 和 HeaderData。神经网络由输入层、隐含层和输出层组成,它依靠神经元内的激活函数构造输入与输出的非线性关系。全连接神经网络结构如图 5 所示。经验证,在表头识别任务中,隐含层采用 ReLU 激活函数比其他激活函数更能使模型快速地趋于收敛,所以在隐含层该文方法采用 ReLU 激活函数。在输出层,由于 Softmax 函数可以将多分类转化为概率,所以该文方法采用 Softmax 激活函数。此外,针对分类问题,交叉熵函数便于梯度下降反向传播,因此,采用交叉熵函数作为网络的损失函数。

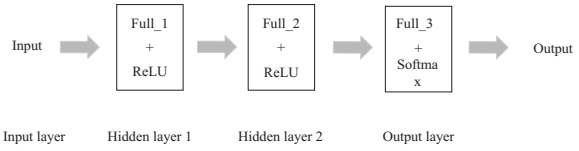


图 5　全连接神经网络的整体结构

为了在反向传播中降低交叉熵,需要以一定的力度调整网络参数。交叉熵与参数的偏导数指明了以多大力度调整该参数,以 W_{11} 为例,经过一次反向传播后,它的新值 W_{new11} 更新情况如公式(7)所示:

$$W_{new11} = W_{11} - \eta \times \frac{\partial E_{Total}}{\partial W_{11}} \tag{7}$$

式中, E_{Total} 为交叉熵, η 为学习速率。经过多次反向传播,网络参数可以更新为最优值。

网络的配置参数如表 1 所示。其中 Full 为隐含层, η 为学习速率。

表 1　全连接神经网络的网络参数

名称	输入维度	输出维度	数值
Input	14	14	–
Full_1	14	10	–
Full_2	10	6	–
Full_3	6	2	–
η	–	–	0.01

在关联关系识别中,首先剔除绝对无关联的单元格对,然后基于规则将表格分割成较小的逻辑块,再应用 Siamese-BiLSTM 模型对其进行关联关系识别,并

根据识别结果中相关联的单元格对,构建表格逻辑结构树。对应图 1,识别出的逻辑结构树见图 6。

```
<Tables>
    <Table column="4" row="4">
        <Cell Paragraph="Test Items" Type="Table">
            <Cell Paragraph="Result Record（unit: second）" Type="indicatio
                <Cell Paragraph="1st" Type="indication">
                    <Cell Paragraph="35.6" Type="entry"/>
                    <Cell Paragraph="34.513" Type="entry"/>
                </Cell>
                <Cell Paragraph="2nd" Type="indication">
                    <Cell Paragraph="34.62" Type="entry"/>
                    <Cell Paragraph="34.797" Type="entry"/>
                </Cell>
                <Cell Paragraph="3rd" Type="indication">
                    <Cell Paragraph="35.67" Type="entry"/>
                    <Cell Paragraph="35.021" Type="entry"/>
                </Cell>
            </Cell>
            <Cell Paragraph="Boot Time" Type="indication">
                <Cell Paragraph="35.6" Type="entry"/>
                <Cell Paragraph="34.62" Type="entry"/>
                <Cell Paragraph="35.67" Type="entry"/>
            </Cell>
            <Cell Paragraph="OS Startup Time" Type="indication">
                <Cell Paragraph="34.513" Type="entry"/>
                <Cell Paragraph="34.797" Type="entry"/>
                <Cell Paragraph="35.021" Type="entry"/>
            </Cell>
        </Cell>
    </Table>
</Tables>
```

图 6 图 1 的表格逻辑结构识别结果(简化表示)

电子表格理解的具体算法详见文献[17]。

3 测试知识库构建

如前文所述,在获得测试数据的知识表示之后,为了形成规范的测试文档,需要构建一个通用、稳定的测试知识本体。由于测试的概念体系是一个典型树状结构,并不适合用关系型数据来描述,因此采用 XML Schema 来描述测试数据的本体结构,对应图 1,测试知识本体的局部结构如图 7 所示。

于是,需要将第 2 节中表格逻辑结构识别的结果转换成符合该本体的测试知识表达,即把 K_p^c 转换为符合本体 O 的规范化的知识表示 K_p(参见第 1 节)。由于 K_p^c 和 K_p 都是 XML 形式的数据,可以简单地设计一个 XSLT 转换程序进行转换,称之为测试项目 p 的 XSLT 式样单 T_p。

为了将来有效利用测试数据,并为测试文档生成做准备,需要有效管理各个测试项目的 K_p,为此采用 XML 数据库管理系统 eXist 来进行测试知识的管理。

eXist 是一个开源的原生 XML 数据库管理系统[18],能够很好地支持数据索引,支持 XSLT,XQuery 和 Java 开发,特别是能够支持无预定义的存储结构,这样能够很好地适应测试数据结构的变化。此外 eXist 可以通过建立共享目录和文件拷贝简便地实现数据入库,对于非专业的测试人员非常方便。

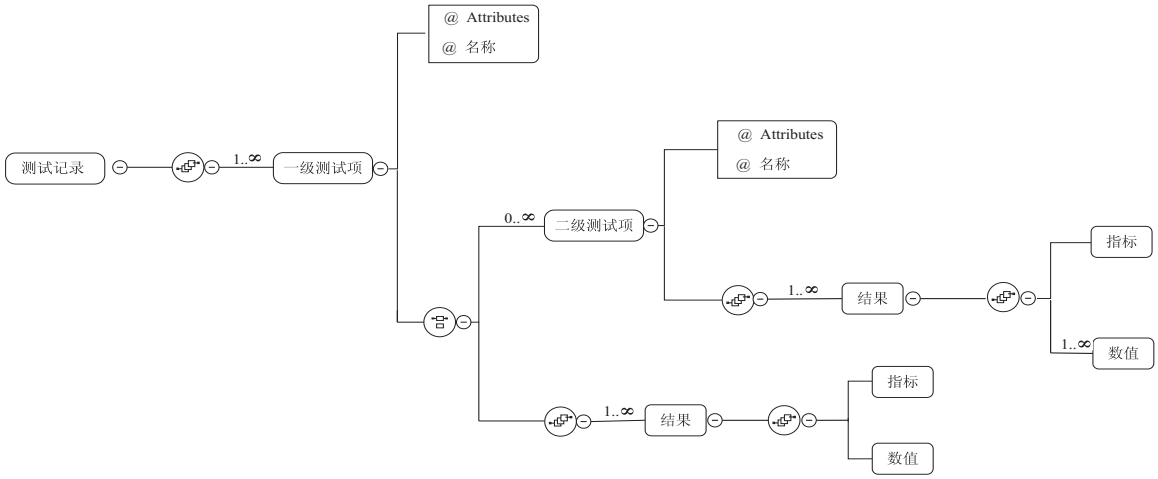


图 7 测试知识本体的局部结构

eXist 数据库的基本概念包括文档集(collection)、资源(resource)和文档(document)。eXist 通过 collection 将一系列数据文件聚集起来,存储所有文件的元数据以及多个 resource。resource 中以命名空间为单位存储多条 XML 数据(document)。在该文的系统中,每一类测试项目 c 对应一个 collection,其中包含各个 p_c 的 resource 以及存放 K_p 的测试数据 document。有了测试知识库,便可以着手生成测试文档了。

4 测试文档自动生成

对于所有的测试项目,测试文档均为符合标准的规范形式,这种形式一般用模板来表达。但是一般使用的静态报告模板存在以下几个问题。首先,难以处理动态变化的数据,例如对空的测试项进行去除,或对重复的测试项增加动态分节;其次,难以根据测试数据的特点按需展示内容,例如对于表格形式的数据,难以提供选择以最适合的图表(chart)形式来呈现;再有,人们习惯了纸质的测试文档,而作为电子形式的测试文档理应有更丰富的交互方式,例如通过插件或智能标签来裁剪或定位报告中的特定内容,或对测试结果进行查询统计。

为支持文档的动态生成能力,采用局部模板,以自底向上的方式来生成测试文档。为测试文档的各个部分设计了局部的模板 t_i,每个 t_i 可能有多个候选 $\{t_{i_1}, t_{i_2}, \cdots\}$,根据数据的呈现要求来选用,全部 t_i 组成测试文档的模板集合 T,

$t_i \in T, 0 < i < N$, N 为测试文档的成分总数。测试文档生成的算法如下:

ReportGeneration (K_p, T)
$i = 1$;
while $i <= N$
{
for each $t_i \in T$
{
定位 K_p 的位置,得到数据 d;
按 d 的特点和呈现要求选择 t_i 中的候选模板 t_{i_j}, $t_{i_j} \in t_i$;
用 d 填充 t_{i_j},生成测试文档的第 i 部分;
}
}
if 需要动态查询等扩展功能
{
构造基于 XQuery 查询程序;
生成查询插件或智能标签;
}

项目要求将测试文档保存成为 Microsoft Word 文档形式,为降低对 Microsoft Office 产品的依赖,没有采用 Office SDK,而基于 OOXML 的开源接口 POI[19] 来开发模板。如果需要生成 HTML 形式的测试文档,可使用 XSLT 构造测试文档的转换式样单,此处不作赘述。这种在不同介质上按需生成文档的能力,体现了复合出版的思想。图 8 是该文生成的测试文档部分内容。

产品质量测试报告

一、测试环境

1 环境适应性与电磁兼容性测试环境

1.1 检测说明与样品描述

表 1　硬件试验测试环境

测试客户端			
序号	配置/性能参数	机身序列号	软件名称及版本号
1	数量：3 机型：2019-288-1 CPU：FT-2000/4（单路） 核数：4 主频：2.6GHz 固件：昆仑固件 V4.0 内存：8GB 硬盘：M.2 512GB	1S5176201201365808 1S5176201201365835 1S5176201201365822	操作系统：统信 UOS V20

表 2　检测说明与样品描述

项目		样品编号	单项判定
电磁兼容测试	静电放电抗扰度	1#	合格
	电快速瞬变脉冲群抗扰度	1#	合格
	浪涌冲击抗扰度	1#	合格
气候环境试验	工作温度下限试验	2#	合格
	贮存运输温度下限试验	2#	合格
	工作温度上限试验	2#	合格
	贮存运输温度上限试验	2#	合格
	工作条件下的恒定湿热试验	2#	合格
	贮存运输条件下的恒定湿热试验	2#	合格
机械环境试验	振动试验	3#	合格
	冲击试验	3#	合格
	碰撞试验	3#	合格
	运输包装件跌落试验	3#	合格
功耗测试		2#、3#	—
噪音测试		2#、3#	—

第 3 页/共 31 页

产品质量测试报告

表 10　整机开机时间测试数据

测试项	结果记录（单位：秒）					
	第 1 次	第 2 次	第 3 次	第 4 次	第 5 次	均值
开机时间	35.6	34.62	35.67	35.84	36.09	35.56

图 10　开机时间对比折线图

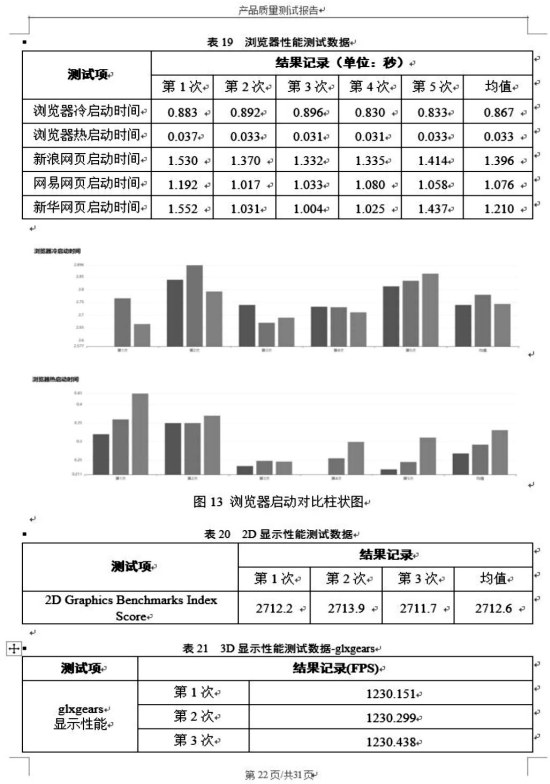
表 11　SPEC CPU 测试数据

测试项	结果记录	
	speed-base	rate-base
整型性能（int）	19.5	88.7
浮点性能（float）	22.1	61.6

表 12　LMbench 测试数据

测试项		结果记录（单位：微秒）			
		第 1 次	第 2 次	第 3 次	均值
Shell 命令启动时间(sh proc)		2277	2263	2206	2248.667
系统信号处理时间(sig hndl)		1.44	1.45	1.44	1.443
2p/16K 的上下文切换性能(2p/16K ctxsw)		6.55	7.03	4.79	6.123
16p/64K 的上下文切换性能(16p/64K ctxsw)		11.9	11.9	12.1	11.967
0K/10K 文件创建时间	0K File Create	31.5	31.4	31.4	31.433
	10K File Create	54.9	58.8	58.2	57.300
0K/10K 文件删除时间	0K File Delete	33.8	34.3	34.2	34.100
	10K File Delete	40.2	40.7	40.6	40.500

第 17 页/共 49 页



产品质量测试报告

表 19　浏览器性能测试数据

测试项	结果记录（单位：秒）					
	第 1 次	第 2 次	第 3 次	第 4 次	第 5 次	均值
浏览器冷启动时间	0.883	0.892	0.896	0.830	0.833	0.867
浏览器热启动时间	0.037	0.033	0.031	0.031	0.033	0.033
新浪网页启动时间	1.530	1.370	1.332	1.335	1.414	1.396
网易网页启动时间	1.192	1.017	1.033	1.080	1.058	1.076
新华网页启动时间	1.552	1.031	1.004	1.025	1.437	1.210

图 13　浏览器启动对比柱状图

表 20　2D 显示性能测试数据

测试项	结果记录			
	第 1 次	第 2 次	第 3 次	均值
2D Graphics Benchmarks Index Score	2712.2	2713.9	2711.7	2712.6

表 21　3D 显示性能测试数据-glxgears

测试项	结果记录(FPS)	
glxgears 显示性能	第 1 次	1230.151
	第 2 次	1230.299
	第 3 次	1230.438

第 22 页/共 31 页

图 8　生成的测试文档(部分)

5　结束语

运用数据到文本的生成理论,提出了一种基于知识提取的测试文档生成方法。该方法首先对原始的测

试记录数据进行分析和理解,重点进行电子表格的表头识别和单元格关联关系识别,抽取出 XML 形式的表格逻辑关系分析结果;其次按照预先设计的、标准化的测试本体,将表格逻辑关系分析结果转换成规范的知识表达,存放在 XML 数据库中,形成测试知识库;最后依次用知识库中的数据填充文档局部模板,逐步获得整体的测试文档。该方法体现了如下的创新性:

较好的通用性和“低代码”,可以支持多类测试和多个测试项目。在不同的测试类型之间切换时,只需要自动识别新的电子表格测试记录,并替换第 3 节中的 T_p 即可。另外,如果测试项目对测试文档的展现形式有特殊的要求,可以定制模板和脚本,此外无需更多的编程,避免了重复开发工作。

通过对测试数据的知识提取,避免了原始测试记录的不稳定性,利于测试结果有效利用。虽然在测试过程中受各种因素影响,难以保证测试人员自始至终采用一致的电子表格记录数据,但是只要表格的逻辑结构不变,总能从中得到正确的知识表示。另外规范化的知识表示可以用于后续对测试结果的各种查询、统计和分析。

在填充局部模板的基础上生成整体测试文档,避免了传统静态模板带来的缺点,能够较好地处理动态的数据,并按需展示数据内容,同时可以扩充交互式的数据访问能力。

系统有较好的跨平台能力,采用开放的标准和开源软件进行开发,不依赖于特定的商业软件。

取得的成果已经通过验收,成功应用于中国电子技术标准化研究院的产品测试部门,从浩繁的测试数据中生成一份测试文档只需要几分钟的时间,在近期大负荷的信创产品测试任务中发挥了重要作用。

在未来可以继续改进的方面有:

(1)考虑将 RDF 等知识表示语法用于测试知识的表示,以期更高效地实现对测试结果的语义检索;

(2)构建更为灵活的文档模板集合,配合智能化的作文规则,进一步提高测试文档的可读性;

(3)将成果推广到其他领域的文档编制,针对敏捷开发和 DevOps 等开发模式特点,探讨从不同来源的原始数据生成完整的、规范的产品文档的可能性。

参考文献:

[1] 黄 申. 我国计算机标准的演进[J]. 电子质量,2017(8):62-63.

[2] 中国标准出版社. 计算机软件工程国家标准汇编,软件开发与维护卷[M]. 北京:中国标准出版社,2007.

[3] 金泽锋,张佑文,叶文华,等. 面向完整价值交付的文档 DevOps 应用研究[J]. 软件学报,2019,30(10):3127-3147.

[4] 李 宁,刘京志,胡景凡,等. 软件结构化文档编制工具的研究与实现[J]. 微计算机信息,2007,23(36):213-214.

[5] 曹 原. 利用版本管理工具实现定制软件文档的自动化生成[J]. 信息通信,2016(12):179-180.

[6] ANDERSON R D, ELOVIRTA J. DITA open toolkit[EB/OL]. (2011-12-22). https://www. dita-ot. org/.

[7] CONE M. Markdown guide[EB/OL]. (2022). https://www. markdownguide. org/.

[8] GOODGER D. reStructured text markup syntax and parser component of docutils[EB/OL]. (2021-10-22). https://docutils. sourceforge. io/rst. html.

[9] MACK S D. Amplify agile with DevOps[EB/OL]. (2020-10-28). https://www. linkedin. com/pulse/amplify-agile-devops-sean-d-mack-mba.

[10] KLAHOLD A, FATHI M. Computer aided writing[M]. Switzerland:Springer,2020:131-132.

[11] BELZ A, KOW E. System building cost vs. output quality in data-to-text generation[C]//Proceedings of the 12th European workshop on natural language generation (ENLG 2009). Athens:EACL,2009:16-24.

[12] 陈玉敬,吕学强,周建设,等. NBA 赛事新闻的自动写作研究[J]. 北京大学学报:自然科学版,2017,53(2):211-218.

[13] CCF 中文信息技术专委会. 文本自动生成研究进展与趋势[C]//CCF2014-2015 中国计算机科学技术发展报告会. 北京:中国计算机学会,2015.

[14] REITER E. An architecture for data-to-text systems[C]//Proceedings of the eleventh European workshop on natural language generation (ENLG 07). Schloss Dagstuhl:EACL,2007:97-104.

[15] Single Source Publishing Community. Welcome to the single source publishing community[EB/OL]. (2021). https://singlesource. pub/.

[16] 孙 培. 军用软件测试文档生成设计与实现[J]. 电子测试,2017(12):39-41.

[17] 张 林. 流式文档表格结构识别研究[D]. 北京:北京信息科技大学,2020.

[18] WolfgangMeier. eXist-db open source native XML database[EB/OL]. (2022-02-09). http://exist-db. org/.

[19] Apache SoftwareFoundation. ApachPOI[EB/OL]. (2022-02-08). https://poi. apache. org/components/.