

基于XML的业务指令分解技术研究

郭小明, 雷 电

(上海大学 自动化系, 上海 200072)

摘 要:业务通知单数据分解为对交易主机操作的控制指令在证券等行业中具有重要的作用,XML技术为此提供了完美的解决方案。介绍了如何把各种业务通知单数据通过分解规则和变换器(XSLT)分解为所需要的操作指令的实现过程,研究了定制各种业务通知单的输入界面、业务集的XML语言描述,以及指令分解规则的通用架构和指令分解变换器的设计技术。实际应用证明此技术方法不但极大地简化了软件开发,而且能适应各种复杂的业务分解,具有很强的应用价值。

关键词:XML;XSLT;架构;业务指令分解

中图分类号:TP311.52

文献标识码:A

文章编号:1673-629X(2009)07-0234-04

Instruction Decomposition Technology Based on XML

GUO Xiao-ming, LEI Dian

(Automation Department of Shanghai University, Shanghai 200072, China)

Abstract:Decompose the control instructions from notifications data plays an important role in the field of security and so on. XML technology provides perfect solution for this. Introduces the realization process of how to get the operation instructions from notifications data throughout transformation rules and converter (XSLT), researchs on the input interfaces design, data items description with XML language, the design of transformation rules' universal framework and converter of instructions decomposition. This method has been proved by applying that it not only can greatly simplify the software development, but also can adapt to all kinds of complicated instruction decomposition, it has great application value.

Key words:XML;XSLT;framework; instruction decomposition

0 引 言

随着计算机软件技术的不断发展,人们对自动化程度的要求越来越高,如何把各行各业中至今还在人工操作的指令分解工作实现自动化,这是一个急待解决的问题。这不仅能提高企业的经济效益,还能把工作人员从繁重的指令分解工作中解放出来。近年来XML已经作为一种新技术迅速发展起来,它的出现为应用程序之间特别是基于网络的应用程序的数据交换提供了一个方便快捷的方法。目前研究较多的是应用XML的自我描述能力实现异构系统^[1]间的数据交换。其主要解决的问题是同一单位的各个部门所使用的数据库不同和数据文件的存放格式不一样所带来的数据交换和传递的困难。然而对于需要通过转换规则的指令分解技术,还少有研究。文中研究的系统就是通过

定制具有统一架构的业务分解规则和和变换器来实现证券业务的指令分解。

1 应用研究背景

证券部门对各类业务通知单(如发行、上市、停复牌等),分解成对交易主机操作的各种指令现采用纯手工的处理,不但效率低,且需要处理的通知单类型越来越多,为安全运行埋下了隐患。有一套可定制的业务指令分解系统具有重要作用,不但能提高工作效率,而且能降低出错率。工作人员只要把业务通知单数据输入系统后便可根据预先设定的业务对应的分解规则,自动产生对交易主机操作的分解指令,用于主机操作。其业务流程如图1所示。

2 系统的设计与实现

2.1 系统设计的关键问题

本系统主要实现根据实际配置的业务规则完成从业务通知单数据输入到操作指令的输出,那么如何才能把各种各样的业务通知单数据分解成对交易主机操

收稿日期:2008-09-23;修回日期:2009-01-01

基金项目:上海市重点学科建设项目资助(T0103)

作者简介:郭小明(1982-),男,浙江富阳人,硕士研究生,研究方向为工业控制组态软件及计算机应用;雷 电,硕士研究生导师,教授,研究方向为工业控制组态软件及计算机应用。

作的指令呢? 此系统的设计首先需要考虑如下三个关键问题:

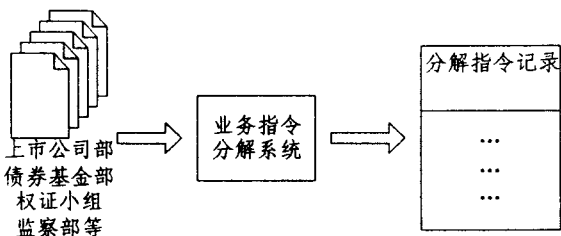


图 1 业务流程图

1)业务通知单类型多种多样,每一类通知单都有自己的转换规则,那么是否就必须对每一类业务通知单单独地定制转换规则和变换器(XSLT)呢? 如果是这样的话,此系统的开发显然就变得很繁琐。那么是否有更好的方法来简化此软件的开发呢?

2)各种业务通知单的转换规则都不一样,但所有业务通知单的转换规则都可归结为以下三种情况:

- (1)变换后得到的值是固定的,即常量。
- (2)变换后的值直接来源于输入的业务通知单或数据库的数据,即变量。
- (3)变换后的值需要获得业务通知单和数据库的数据然后进行运算得到,即函数值。那么如何用 XML 语言描述转换规则的三种取值问题呢?

3)变换器(XSLT)需要完成业务通知单数据根据转换规则得到操作指令的过程,那么在变换器中又是如何实现这三种取值的变换呢?

2.2 各功能模块实现

针对以上三个问题,对指令分解系统进行设计。指令分解系统的功能实现过程如图 2 所示。

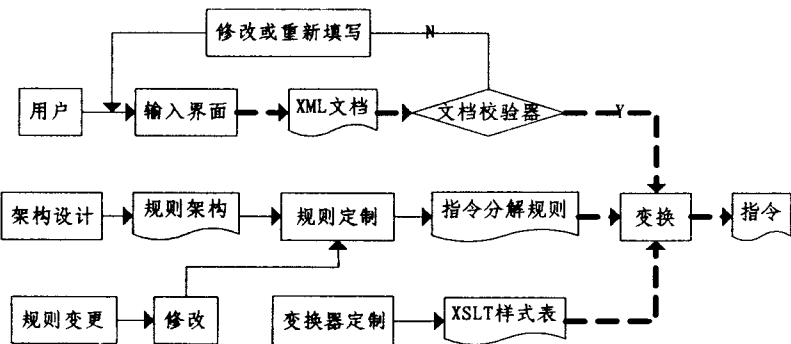


图 2 指令分解的实现框图

2.2.1 业务集描述

不同的业务通知单(如发行、上市、停复牌等)定制不同的输入界面,界面样式类似各业务部门提交的通知单。业务集描述^[2]即对交易运行部所收到的所有业务通知单类别,采用 XML 语言的自描述功能,对每个具体业务的内容进行描述,定义该业务中包括的所有业务项、业务项类型、约束等。当在界面中输入通知单

数据后,就得到如下 XML 结构文档^[3],并保存至数据库。

```
< theData>
  < 字段名>值< /字段名>
  .....
< /theData>
```

例如在新股发行业务的界面中输入:
主机操作 证券代码:600050;证券简称:中国联通;
发行价格:100 元;发行日期:2008 - 09 -
12 等等,输完后按确认按钮,保存这些数据,XML 文档获取输入的数据后得到如下文档:

```
< theData>
  < SECURITY_CODE>600050< /SECURITY_CODE>
  < SECURITY_ABBR>中国联通< /SECURITY_ABBR>
  < RATED_PRICE>100< /RATED_PRICE>
  < RATED_DATE>2008 - 09 - 12< /RATED_DATE>
  .....
< /theData>
```

2.2.2 规则的架构和定制

规则架构就是给以 XML 文档描述的业务指令分解规则一个统一的模板。正是有了这个统一的模板,所以不同的业务,虽然转换规则不同,但其结构都是相同的,只要设计一个通用的变换器(XSLT 样式表)就可以来实现各种不同业务的指令分解,这就大大简化了此软件的开发。在此系统中,用 XML Schema^[4]对转换规则设计了架构。业务指令分解的转换规则架构如图 3 所示。

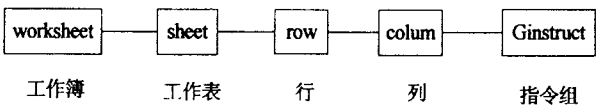


图 3 分解规则架构

规则定制就是建立业务与指令集的对应关系,每个业务可分解成多条指令,分解规则将定义业务如何分解成多条指令,指令中的各指令项源于业务通知单的哪项内容,建立业务同指令的映射^[5]关系。针对各个业务,通过指定操作日期、操作时间、代码、价格、操作指令等项,定义单个业务对应的分解指令,不同的业务具有不同的规则,但所有描述规则的 XML 文档都必须符合图 3 的架构。此架构的根元素为 Worksheet(工作簿),Worksheet 元素下可以包含一到多个 Sheet(工作表)元素,每个 Sheet 元素又可以包含零到多个 Row(行)元素,Row 元素下又有若干个 Colum(字段)元素,Colum 元素下就是变换规则 xml 表达式,包括常量、变量和函数。这三种取值的

表示分别如下:

(1)取的值为常量,则分解规则表示为<const value="常量"/>。

(2)取的值为变量,即此值来源于业务通知单或数据库表某字段的值,分解规则表示为<dataFld name="数据库字段名"/>。

(3)取的值为函数值,即需要获得业务通知单和数据库的数据然后进行运算得到。函数调用在规则中具体表示为:

```
<callFunc name="函数名">
  <with_param>
    .....
  </with_param>
  .....
</callFunc>
```

当然参数的取值又会涉及到这三种情况,即取的值为常量、变量以及函数值,那么它的表示方法就是把以上三种取值表示法嵌套进去。具体每个字段的规则定制通用方法如图 4 所示。

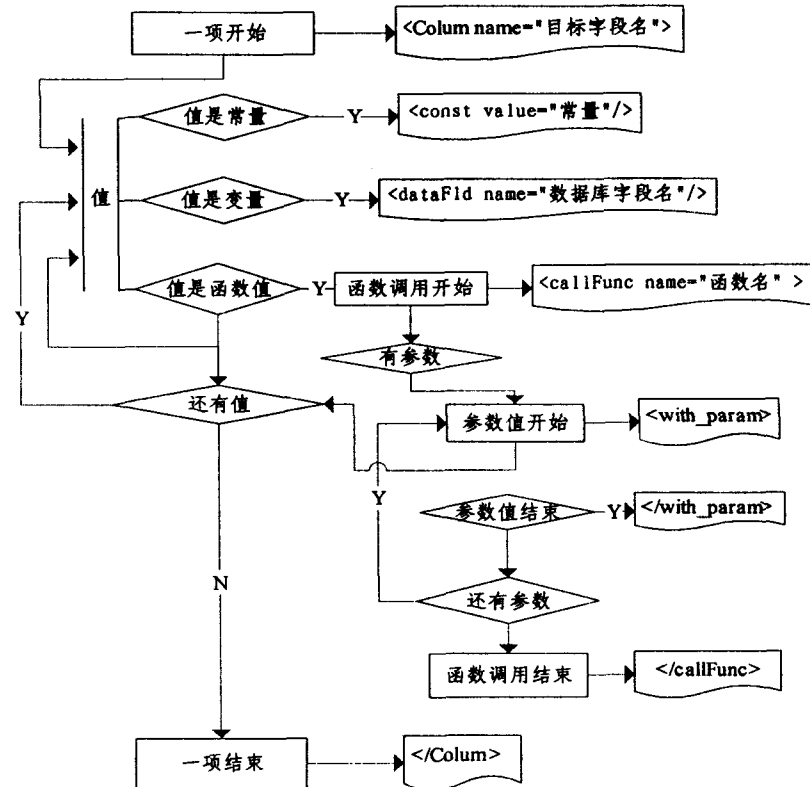


图 4 字段的规则定制通用方法

以下是根据架构制定的一个转换规则实例的一部分:

```
<Worksheet name="新股发行通知">
  <Sheet name="接通知单当日">
    <Row>
      <Colum name="内容">
```

```
<const value="放 N"/>
</Colum>
<Colum name="SECURITIES_NAME">
  <dataFld name="RATED_ABBR"/>
</Colum>
<Colum name="T+1">
  <callFunc name="getDate">
    <with_param data="T+1"/>
    <with_param>
      <dataFld name="RATED_DATE"/>
    </with_param>
  </callFunc>
</Colum>
.....
</Worksheet>
```

2.2.3 通用变换器(XSLT)

XSLT 转换^[6]通常涉及三个文档:源文档、目标文档和含有模板规则的文档即 XSLT 转换样式表^[7]。XSLT 转换器以 XML 源文档输入,利用预先编写好的转换模板(XSLT 转换样式表)把它转换成目标文档,在此系统中 XML 源文档包括从界面输入所产生的 XML 文档和指令分解规则的 XML 文档。XSLT 转换文档设计如下,首先执行根节点下的 Worksheet 模板:

<xsl:template match="/">
 <xsl:apply-templates select="//Worksheet"/>
</xsl:template>

接着就是执行 Worksheet 模板下的每一个 sheet 模板:

```
<xsl:template match="Worksheet">
  <指令集>
  <xsl:apply-templates/>
</指令集>
</xsl:template>
```

以此类推,一直往下一级模板执行,最后执行到每一个字段(Colum),即

```
<xsl:template match="Colum">
  <xsl:element name="{@name}">
    <xsl:apply-templates/>
  </xsl:element>
```

```
</xsl:template>
```

到这里,碰到了一个取值问题即<xsl:element name="{@name}">,这个取值比较简单,例如在上文的规则实例中有这样的代码:<Colum name="内容">,那么在执行<xsl:element name="{@name}">后,就在此字段中直接填入“内容”两字,它就是把在

规则中定义的 Colum name 的值显示出来。接着变换器就要往下一级模板运行,这就遇到了前文中所说的三种取值,那么在变换器中是如何实现这三种取值的变换的呢?

1) 常量取值的变换实现:

```
<xsl:template match="const">
  <xsl:value-of select="@value"/>
</xsl:template>
```

当执行到元素为 const 的模板时,就把预先定制的常量值赋给它。例如在上文的规则实例中有这样的代码<const value="放 N"/>,运行 const 模板后就会在此字段填入“放 N”两字,这种取值的变换和上面一种类似。

2) 变量取值的变换实现:

```
<xsl:template match="dataFld">
  <xsl:apply-templates select="//theData/*[name()=current()/@name]"/>
</xsl:template>
```

当执行到 dataFld 元素时,就把 dataFld name 对应的字段名的具体值赋给它,这个值就是 theData 目录下的对应字段名的值。例如执行<dataFld name="RATED_DATE"/>,我们就去 theData 元素下找 RATED_DATE 字段,找到后发现它对应的值是 2008-09-12,我们就把这个值输出。

3) 函数调用的变换实现:

函数调用的变换实现较前两种取值复杂,在 XSLT 文档中其主要代码如下:

```
<xsl:template match="callFunc">
  <xsl:variable name="size" select="count(with_param)"/>
  <xsl:variable name="call">
    <xsl:value-of select="@name"/> (
    <xsl:for-each select="with_param">
      <xsl:apply-templates select="."/>
    <xsl:if test="position() < $size">, </xsl:if>
    </xsl:for-each>);
  </xsl:variable>
  <xsl:value-of select="L:runCommonCode(string($call))"/>
</xsl:template>
```

首先定义一变量 size,取得具体的被调函数的参数个数,然后把值赋给 size。然后定义一变量 call,取得被调函数的函数名和具体的参数值,然后赋给 call,运行 call 函数,即得到目标函数值。runCommonCode 是在变换器中定义的脚本函数,其功能是运行用户的 call 函数。

下面用一个例子来说明函数调用的变换实现,例如被调用的函数代码如下:

```
function getDate(TnDate,Tdate)
{
  var obj = new ActiveXObject("ELogObj.LTradeDate.1");
  var date = obj.getTradeDate(TnDate,Tdate);
  return date;
}
```

在规则中的代码为:

```
<callFunc name="getDate">
  <with_param data="T+1"/>
  <with_param>
    <dataFld name="RATED_DATE"/>
  </with_param>
</callFunc>
```

假设从界面输入的 RATED_DATE 字段的值为 2008-9-12,在变换过程中首先匹配<xsl:template match="callFunc">模板,该模板内的<xsl:value-of select="L:runCommonCode(string(\$call))"/>即调用 runCommonCode(\$call)函数,此处\$call=getDate(T+1, 2008-9-12),由此可得函数调用结果为 2008-9-16,即下一个交易日是 2008-9-16(中秋放假)。

至此,通用变换器(XSLT 样式表)的定制就得以实现。

3 应用举例

通过输入界面的定制和对业务数据的描述、分解规则的架构设计以及具体业务的分解规则定制再加上通用转换器的设计,这使得系统能够把具体的业务数据转换成对交易主机操作的控制指令。下面以新股上市业务通知单为例来说明系统的指令分解功能。例如在新股上市业务输入界面中输入以下数据:证券代码:600050;证券简称:中国联通;上市日期:2008-06-19;发行价格:100;通知单日期:2008-06-19,输入完毕后保存这些数据,可得到输出的部分指令如表 1 所示。

表 1 新股上市业务的部分指令

日期	证券代码	证券简称	内容	时间
2008-06-19	600050	中国联通	代码生成,价格 100	白
2008-06-19	600050	中国联通	放 N	前
2008-06-19	600050	中国联通	复名中国联通 10%	后

4 结束语

提出了基于 XML 的业务指令分解技术,利用 XML Schema,XSLT 建立了通用的业务指令分解规则架构和变换器。从而完成了从业务通知单数据到目标

表 2 典型 P2P 应用 Payload 特征

应用类型	流量类型	Payload 特征
PPLive	TCP	连接建立后的第 1 个报文为 4 字节,内容为 0x39 00 00 00
	UDP	前两个字节是 0xe9 03
BitTorrent	TCP	包含:0x19BitTorrentprotocol
QQLive	UDP	第 1 个字节为:0xfe
eMule	TCP	第 1 个字节为:0xe3 或 0xc5 或 0xd4

3.2 系统测试

在实验室环境下以拥用 10 台电脑的局域网作为检测对象分别对 PPLive、BitTorrent、QQLive、eMule、Kugoo 和非 P2P 应用进行连续 24 小时测试。Payload 特征法和双重特征法采用的特征串中,前四种 P2P 使用表 2 中 Payload 特征作为匹配字符串,Kugoo 无特征字段。采用虚警率和误报率作为比较标准,实验结果如表 3 所示。

表 3 双重特征法与 DPI 法的虚警率
和误报率对比表

应用类型	数据量 (GB)	Payload 虚警率(%)	Payload 误报率(%)	双重特征 虚警率(%)	双重特征 误报率(%)
PPLive	30.6	2.20	0	3.41	0.14
BitTorrent	20.3	62.51	0	2.52	0.11
QQLive	32.5	0.51	0	0.51	0.09
eMule	10.9	2.82	0	1.95	0.02
Kugoo	2.5	100	0	4.17	0.17
传统应用	5	0	0	0	0.04

由表可知,基于双重特征的 P2P 流量检测法相比于单纯的 DPI 法拥有较低的误报率,特别是对于 Payload 特征字段落后的 Bit Torrent 和无 Payload 特征字段的 Kugoo,仍有较高的识别率。虽然双重特征的 P2P 流量检测法有一定的虚警率,但是仍一直维持在可接受的范围之内。

(上接第 237 页)

指令的成功分解。提出的具有统一架构的指令分解规则以及通用转换器定制的方法灵活适应了业务需求的变化,当有新的业务产生时,只要制定新的输入界面和按照架构定制新规则[0]就可以完成新的业务指令的分解工作,当规则发生变化时,只要按照架构修改旧规则即可,而无需对系统进行修改。此方法也为其他领域的数据库转换技术提供了新的思路和方法。

参考文献:

[1] 盛贤良,瞿有甜. 基于 XML 的异构构件组装技术研究[J]. 计算机技术与发展,2008,18(2):83-87.
[2] 元祥波,南琳,张福顺. 基于元数据和 XML 的信息抽取

4 结束语

基于双重特征的流量检测方法不仅继承了 Payload 特征法检测准确、误报率低的优点,而且通过选取合适的流量特征能够识别经过加密和未知的 P2P 流,拥有较高的识别率。

为了适应 P2P 应用进一步发展,进一步降低误报率和漏报率,需要发现新的更具代表性的 P2P 流量特征,以及采用机器学习、数据挖掘和硬件实现该检测算法的新方法,使之能够应用于大规模流量环境中,提高网络服务质量。

参考文献:

[1] 杨磊,姜浩. P2P 技术在网络文件管理中的应用[J]. 计算机技术与发展,2006,16(12):127-129.
[2] Sen S, Spatschecko, Wangdm. Accurate, Scalable In - Network Identification of P2P Traffic Using Application Signatures [C]//WWW2004. New York, USA:[s.n],2004.
[3] Karagiann I T, Brodo A, Faloutsosm. Transport Layer Identification of P2P Traffic[C]//Proceedings of the 4th ACM SIG - COMM Conference on Internet Measurement. New York: ACM Press,2004:121-134.
[4] 王锐,王逸欣. 一种跨层 P2P 流量检测方法[J]. 计算机应用,2006,26:30-32.
[5] 柳斌,李之棠,李佳. 一种基于流特征的 P2P 流量实时识别方法[J]. 厦门大学学报:自然科学版,2007,46:132-135.
[6] 温超,郑雪峰,戚翔,等. 基于流量分析的 P2P 协议识别方法的研究[J]. 微计算机应用,2007(7):714-717.
[7] 石萍,陈贞翔. 基于对等特征的 P2P 流量识别方法[J]. 网络测量与规划,2007(2):36-38.
[8] 蒋海明,张剑英,王青青,等. P2P 流量检测与分析[J]. 计算机技术与发展,2008,18(7):74-76.

与集成技术研究[J]. 信息与控制,2008,37(1):52-57.
[3] Florescu D D, Kossmann D. Storing Querying XML Data using an RBMS[J]. IEEE Data Engineering Bulletin,1999,22(3):27-34.
[4] 王大刚,谢荣传,彭俊. 基于 XML Schema 的数据匹配方法的研究[J]. 计算机技术与发展,2008,18(6):28-31.
[5] 黎建辉,吴威,阎保平. 一种基于 XML 的元数据映射与转换方法[J]. 微电子学与计算机,2008,25(1):34-38.
[6] 孙国强. XML、XSLT 在应用系统集成中的功能研究[J]. 科学技术与工程,2007,7(13):3124-3128.
[7] Gardner J R, Rendon Z L. XSLT and Xpath: a guide to XML transformations[M]. [s.l.]:Prentice Hall,2002.