基于 XML 和 JAVA 构建程序生成器

冯 少 荣
(厦门大学计算机科学系 厦门 361005)

摘 要 本文将 XML、JAVA、JSP 及程序生成器等技术与方法结合起来,通过域分析的思路、方法、过程,讨论了使用 XML 和 JAVA 创建程序生成器的方法和过程,并比较了不同方法的优劣。指出了将 XML 应用到程序创建和维护上的优势。

关键词 XML JAVA 程序生成器 域分析

CONSTRUCTING PROGRAM GENERATOR BASED ON XML AND JAVA

Feng Shaorong
(Department of Computer Science, Xiamen University, Xiamen 361005)

Abstract In this paper, the authors integrate XML, JAVA, JSP and program generator technique, and analysis the thoughts, method and process by domain analysis approach. Then discuss two different methods and procedures of building a program generator respectively by XML or JAVA. And the advantages and disadvantages of these two different methods are given. At last the authors list the benefits from applying XML to the creation and maintenance of programs.

Keywords XML JAVA Program generator Domain analysis

1 引 言

程序生成器可以加快程序编码产生的速度,产生规范和正确的代码,编写一个程序生成器,它意味着不仅仅是写一个程序,而是要写一个可以写出许多程序的程序。在用户界面、数据库、中间件、语法分析和语法分析等方面,程序生成器是其开发环境中的重要部分。程序生成器的思想已经使用了很多年了。比如:

® *语法分析器* 语法分析器读入一个标记序列,并且创建一个称为语法树的描述信息的数据结构。典型的为UNIX 使用程序 yacc,它读入一个语言的形式描述(用某种语法表示)连同看作是语法规则的动作,然后输出一个语法分析器程序。

® *有限状态机* 可以用显示不同状态、事件及状态转移的表和(或)图对程序进行描述和说明。典型的为 UNIX 实用程序 lex,就是一个有限状态机的程序生成器,并且用 yacc 来解析语言的标记。

® *用户界面* 现在大多数代码都是由 GUI 构造器和直观的编程工具自动生成。

® *数据库中的程序生成器* 给出数据表、关系、事务逻辑及报表模式的描述,就能生成依照指定的规格构造的数据库程序。如:报表生成器、菜单生成器、屏幕生成器等。

® *Web 页面生成* Java 服务器页面 JSP 是一种在 Web 上创建动态内容的工具。

而 XML 作为一种完全可移植的数据格式,将成为跨平台的不同系统之间的数据交换及数据显示、描述的标准。Java 具有面向对象、跨平台、分布式、简捷、健壮、安全等特点,功能强大且简单易学,正在逐步成为新一代网络编程的主要开发语言,Java 将是网络上的"世界语",今后所有用其它语言编写的软件统统都要用 Java 语言来改写。XML 和 Java 的结合对已有的程序生成器技术提供了新的应用背景,并且两者相互协调补充。以更简单而且优雅的风格、可靠的性能,提高程序生成器的开发效率。

2 程序生成器的结构

作为程序生成器的典型结构,它由获取数据、分析/转换数据、生成程序三个部分组成。

程序生成器经过如图 1 所示三种约束时间,完成程序的生成。

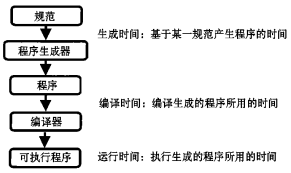

图 1 约束时间

3 域工程技术

域工程是一个用于高效创建一个应用(程序或软件组件)族成员的过程。它是一个确定某个专业的重要组成与需求的系统方法,对如何高效地建立一个满足用户需要的程序生成器是非常必要的。程序的自动创建依赖于对语法和期望目标的精确描

收稿日期:2003-09-01。冯少荣,副教授,主研领域:数据库技术。

述,域工程技术将此概念延伸到时间基础上分析一系列相关的程序,并能够生成和修改它们。生成系统的构造过程就是一个解决问题产生软件程序过程。因此,域既是一个有关联的问题的集合,也是一个有关的软件应用程序的集合。域工程分为两个过程:域分析及域实现。

域分析 是一个用于确定域的术语、范围、共性及变性的过程。

域实现 是指在域中按照指定的要求高效构造应用及创建工具,在域分析结束时即开始。这其中包括程序生成器。

任何开发过程都要经历一系列的决策。在需求分析、软件结构、界面开发、软件设计(算法、数据结构和数据表示)、软件编码与测试以及软件的使用过程中都需要作出决策。有关的最艰难的工作是作出决策而不是编码。对于域工程,最困难的是判断哪些是重要的决策,由谁来制定,什么时候制定。并且要能区分其中的重要决策和不重要决策。决策的制定,要考虑域工程的过程中下列三种重要角色的作用及其作用时间(约束时间)。

域工程师 定义并建立如程序生成器那样的过程和工具。

应用工程师 利用如程序生成器那样的工具创建应用程序。

应用程序用户 使用应用程序。

域工程时间 由域工程师为每个应用程序族作出决策的时间。包括:域分析时间,域执行时间。

约束时间 由应用工程师为每个应用程序作出决策的时间。包括:规划时间,生成时间,编译时间,设计时间,链接时间。

运行时间 由应用程序用户为每个特定的使用作出决策的时间。包括:安装时间,初始化时间,真正运行时间。

决策分解是域实现的基本概念,它简单地分解每个独立的决策,并将其作为一个独立元素或系统组件。如决策的物理分解可产生3种信息:域信息、用户信息、应用信息。有效的决策分解将产生一个既易于改变又易于构造的软件结构。决策分解基于抽象化。抽象化是一种使软件更具有一般性、灵活性、可理解性和可重用性的主要技术。

决策分解技术主要包括:

(1) 物理分解:如,应用程序可以分解出不依赖于机器的部分和依赖于机器的部分。

(2) 典型过程的抽象化:如,子程序、宏及可重用软件包的创建。

(3) 面向对象的抽象化:如,隐含在对象中的隐藏决策。

(4) 继承方法:公共关系在父类和子类中被共享使用。

(5) 应用程序框架:可看作一个软件重用的自顶向下的方法。

(6) 规范驱动技术:应用规范层信息创建或直接执行应用程序。

域工程是创建应用程序族的过程,因此,不仅要考虑单一的应用会如何随时间而改变,而且也要注意到域领域应用的整个范围,从而确定这些应用之间的差别。这些差别称为域工程的可变性,它是域工程的核心。对于域工程需要理解一个应用族中什么是不变的,什么是可变的。不变的成分称为共性,最困难的就是确定并组织一个域的所有共性与可变性。

共性 是一个在域工程期间关于什么是整个域的共同性问题所作出的决策或假设。共性可以确定域的范围、软件的功能、与其它域的分界面、操作环境、软件的限制标准以及应用软件公共部分实现的细节。共性和标准有许多共同点。标准是做某件事情的共同方法。许多为标准所做的努力根本上就是一个域分析。

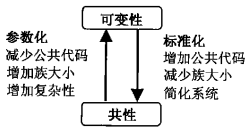

图2 共性与可变性的转换

相反域分析工作附带产生一些标准,以帮助创建更大的将为一大群人所接受的共性集合。

可变性 是一个在域分析期间确认但直到建立或运行时间才确定的决策。

共性和可变性经常可以相互转化,而使域分析保持平衡。如图2所示。

可变性包括以下五种:

① 建立时的可变性

表示程序族的全部程序中的差别是什么。建立时的可变性集合是为建立一个程序生成器所需要的信息中最重要的部分,它可以用于定义一个规范语言和一个支持可变性的结构框架。

② 运行时的可变性

是在运行时间内确定的决策。运行时的可变性建立在应用软件的公共部分。由许多方法表示和控制。比如:

® 资源和配置文件:这些文件包含信息并在运行时间内读入,一般保存在一个外部文件。

® 数据库:查询数据库获得读入信息。

® 用户界面:通过与用户交互获得信息。

® 动态加载类:常见的变化可以被确定、创建、编译,并且直接加载到一个正在运行的Java程序中。

③ 编译时可变性

是在编译时间确定决策。可以由许多方式实现。比如:

® 编译时的常量:可以利用好的编译器来优化程序。

® 面向对象技术:基本类定义共性,子类提供可变性。特别是编译时的可变性。

④ 生成时可变性

信息通过一个创建定制程序的程序生成器读入。

⑤预处理时可变性

预处理发生在编译时间前,预处理器用于扩展宏功能。依据分开的头文件给程序构造可变性。

域工程经常从应用工程周期获得适当的反馈而不断展开。形成域工程周期。应用工程周期和域工程周期的关系如图3所示。

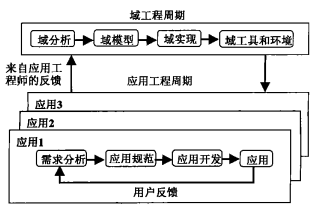

图3 域工程与应用工程周期

4 程序生成器的实现方法

4.1 利用 DOM 生成程序

使用XML文件和DOM构建一个程序生成器,如图4所示。

使用XML语法分析器读入和存储规范,读入XML文件并对其进行语法分析,创建DOM对象。XML语法分析器需要三种输入:

® 由W3C提供的DOM接口的输入;

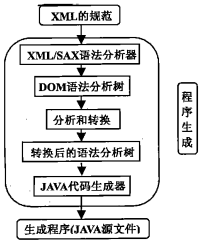

图4　使用 XML 文件和 DOM 构建一个程序生成器

® 针对 IBM 的语法分析器的输入;

® 用于读入文件的标准的 Java 输入数据包。

一旦 XML 文档进行过语法分析, 并且作为一个对象使用时, 就可以在该对象上执行分析和判断。分析包括:

® 检错　在规范中查找句法错误。

® 警告　查找公共错误或潜在的不明显的错误。

® 模型分析　分析规范更深层的语义。

® 性能分析　决定生成一个优化程序结构或代码途径的规范。

® 缩写的扩展　规范中常提供缩写或快捷方式。

® 扩展成标准形式　有些规范允许以各种方式说明同一件事情。

® 优化　转换对象。

一旦 DOM 数据结构存储于内存中, 就可以把基于 XML 文档的代码用于分析和转换结构, 最终代码生成器直接从 DOM 数据结构中获取信息。

4.2　利用 JSP 生成程序

JSP 是普遍应用于 Internet 中的重要的程序生成器。JSP 是一种非常简单的规范语言, 包含了输出到 Web 页面的静态文本和调用基本实现语言的转义符, 它没有高层次的抽象化并且不涉及语言结构。转义符提供很大的灵活性, 同时又只产生最小的影响。JSP 是一种既简单又强有力的技术, 用于在 Web 服务器端生成动态的 HTML 页面。JSP 提供了一个非常简单的利用“模板”创建程序生成器的途径。尽管 JSP 是用于设计和实现 Web 页面的, 但原则上可以传送任何其它内容, 特别是它可以传送 Java 程序。

4.3　利用 Xpath 和 XSLT 生成程序

XSLT 和 Xpath 能够不利用任何 Java 代码就可以创建程序生成器。

图 5　XSL 处理

Xpath 是一种用于从 XML 文档提取信息的语言, 用在 XSLT 中选择一个 XML 文档的不同部分。

XSLT 又称可扩展的样式表转换语言, 可以将 XML 文档翻译成其它不同结构的 XML 文档或纯文本文件。XSLT 提供了一种转换和操作 XML 数据的机制。如图 5 所示。

XML 构成了信息互交换标准的基础。XML 提供了信息构成的结构, XSLT 与 Xpath(XML 路径语言) 提供了提取、重建和熟练使用 XML 中信息的手段。Xpath 使用一种简单的路径语言来对 XML 文档的各个部分进行寻址, XML 提供一系列的操作和操作方法。而 Xpath 保证了选择和寻址的准确度。

4.4　模板语言

JSP 和 XSLT 方法对于构造程序生成器是合理而有效的, 然而它们都不是专门为编写程序生成器而设计的。在应用时都有一些不足。因此, 可以开发一种新的模板语言结合 JSP 和 XSLT 的优点同时避免缺点。可以考虑用 Java 语言作为基本实现语言(但不必为输出语言) 的程序生成器模板, 把 XML 作为输入的规范语言。

新的模板语言的设计和基本原理应具有以下设计目标:

® 具有简单强大的表达式语言, 以此从 XML 文档获得信息。该语言应该允许通过 XML 树来选择元素和属性的特定子集。

® 具有一种把来自 XML 规范的信息插入到生成的输出中的简洁方式, 也应该有重复规范中的元素的简洁方式, 并且应该具有条件语句生成部分。

® 具有表示复杂而独特的重复以及其它控制流结构的方式。

® 具有对 Java 的所有特征的完整而直接的端口。包括可以和其它组件和库结合的能力。

® 提供与 DOM API 的接口。用于直接在 XML 树上进行转换。

® 在不过分妨碍程序生成的情况下, 允许进行空白字符控制。

® 使程序设计语言中的转义符达到最小。

® 在已用命令行自变量指定输入输出文件的命令行处理器下是可执行的。使程序生成器可以很容易的嵌入其它软件设计工具。这种程序生成器对于其它 Java 对象也是可用的, 特别是一个模板可以使用其它模板。

® 允许程序生成器有多重输入输出, 它允许在多个文件中利用一个规范来组织这些文件, 并且允许一个已生成的程序跨越多个文件。多重输出对于生成无代码文件, 如文档、测试脚本及其它文件是非常重要的。

® 能与 XML 进行相互转换, 以便 XML 的所有工具可以被模板所使用。JSP 的 XML 版本就是具有这个特征的一个示例。

® 允许选择编译或解释模式。

4.5　面向组件的程序设计

大多数程序生成器可以生成组件, 这些组件与其它组件相结合以产生软件系统。组件以及它们在软件系统中的构成称为面向组件的程序设计。组件、接口和连接器是面向组件的程序设计的 3 个主要因素。程序生成器有时也作为产生组成系统代码的工具, 即作为一个将系统关联在一起的“胶合剂”。软件组件是仅由协议特定的接口和明确的语境构成的软件单元。它既可以单独使用也可以被第三方调用。JavaBeans 模式是面向组件程序设计的一个示例。接口是组件和其它软件元素间的一种交互。接口的语义描述组件的运行时行为, 接口描述语言(IDL) 在语法层上描述组件的接口。模块互连语言(MIL) 也是一个描述一系列组件相互连接的 IDL。两个组件之间的连接有时叫作连接器。IDL 描述组件和接口, MIL 描述组件、接口和连接器。连接器将一个组件的导入与另一个组件的导出相连接。接口适配器通过调用组件和直接访问被调组件的适当方法来实现接口。

5　程序生成器的设计风格

程序生成器可以使用如下程序设计风格:

面向对象的驱动风格　使用面向对象技术来组织程序的结构。

代码驱动风格　在任何需要的地方生成代码并直接嵌入数据, 不必多担心程序结构, 而只需将考虑的重点放在简单高效的代码上。

表驱动风格　将数据与代码分离。规范信息存储在专门设

(下转第 126 页)

数据以产生 HASH 值。实现中 pbData 接收的就是 word 文档的正文内容。

调用完成后,合同内容被散列成 128 位散列值。

4.4 加密算法

最常见的加密算法是对称加密,即加密和解密使用相同的密钥,这种方法的加密效率是很高的,但是有个致命缺点就是如果密钥丢失,那么加密的内容将不再安全,而且对称加密无法确认加密者的身份。1975 年,Whitfield Diffie 和 Martin Hellman 提出了另一种不同类型的加密和解密算法,使用两个不同但是相关的密钥来执行加密和解密,这种方法被称为非对称加密算法。加密者一般使用私钥对信息进行加密,私钥是不公开的,加密者必须妥善保存。加密者把与私钥对应的公钥分发出去,用于解密者对密文进行解密。这种算法的一个保证是无法从公钥中用计算上可行的方法派生出私钥。所以即使第三方拿到了公钥,也不能篡改和伪造加密者发送的信息。但非对称加密算法的计算量很庞大,一般无法使用它对大信息量进行加密。

本系统使用的是非对称加密算法,客户通过从密钥文件中取得的私钥对前面生成的 128 位散列值进行加密形成密文,密文长度仍旧为 128 位。

4.5 加盖水印

本系统采用的数字水印算法是最低有效位方法,将经过加密后的 128 位密文隐藏入 24 位位图图片的最低有效位中。本方法的原理是对于 24 位位图文件,除去文件头的 56 个字节,从文件的第 57 个字节开始的数据区,每三个字节代表一个像素点的颜色信息,这三个字节分别代表蓝、绿、红三基色在此像素中的亮度,而每个字节的 8 位中的最低一位的值是 0 或者 1,对于这个像素点的颜色影响微乎其微,肉眼根本无法察觉,所以就将密文隐藏于数据区字节的最低一位上。

以下的程序段演示了如何将水印信息藏入图片中:

```
while(!feof(f_encryptedFile))
{// f_encryptedFile 指向存放密文的文件。
  eChar= fgetc(f_encryptedFile);
  // 读取密文的一个字节
  fread(b_bufs, sizeof(char), 8, f_bmp);
  // 从图片中读取 8 个字节放到缓存 b_bufs 中,指针 f_bmp 已经指向了第 57 个字节处。
  for(i= 0; i< 8; i++)
  {
    if(eChar& mask[i])>> i)
  // mask[i]为掩码,表示一个第 i 位为 1 其余位为 0 的字节。
      b_bufs[i]|= 1;
    else b_bufs[i]&= 0xfe;
  }
  fwrite(b_bufs, sizeof(char), 8, f_desfile);
  // 将加入了密文的 8 个字节写入目标文件
}
```

4.6 完整性和不可否认性

整个系统从客户告知公司需要的货物,公司根据需求拟定合同,加盖公司水印之后传递给客户。客户取得这份合同之后,使用公司的加密公钥,验证合同中的水印,确保公司传来的合同是未经修改的、完整的以及这份合同是公司签发的。然后客户如果接受合同内容就在其上加盖自己的数字水印,并将合同传回给公司。公司通过该客户的公钥,验证合同中的水印确认客户传来的合同是未修改的、完整的以及这份合同是该客户签发的。这样在整个合同的签订过程中,双方都可以确认这份合同是对方签发的而且是未被第三者改动过,从而保证了合同的完整性和不可否认性。

5 结 语

本文提出了一种采用数字水印技术达到保证电子合同完整性和不可否认性的解决方案,文中对应用到的关键技术进行了比较详细的论述。这套安全电子合同系统,在实际应用中,加速了公司的业务流程,得到了公司以及客户的广泛好评。但是本系统还存在一个比较脆弱的一环,系统数字水印部分采用的最低有效位方法是比较脆弱的一种水印,任何对位图的有损无损的处理都有可能导致水印信息的丢失,这些都将在下一步的工作中完善。

参 考 文 献

[1] Ingemar J. Cox, Matthew L. Miller, Jeffrey A. Bloom, Digital Watermarking, 电子工业出版社, 2003. 7.

[2] 陈明奇、钮心忻、杨义先,“数字水印的研究进展和应用”,《通信学报》, 2001. 5.

[3] Microsoft VisualStudio. Net 文档.

[4] 李海泉、李健,计算机网络安全与加密技术,科学出版社, 2001. 3.

(上接第 59 页)

计的数据结构中,并且代码在适当的时候参照数据结构恢复数据。

6 结束语

程序生成器思想已产生许多年了,并且大多用 C 语言等纯文本语言实现,而通过 XML 和 JAVA 等作为实现的技术并不多见,随着 XML 和 JAVA 技术的不断成熟,以及在各个领域的广泛应用,XML 和 JAVA 给这些技术提供了新的应用背景。XML 可以用来生成独立于应用程序和平台的数据,JAVA 技术可以用来实现独立于应用程序和平台的事务逻辑。两者相互协调补充。本文通过域分析的思想、方法,结合 JAVA 和 XML 技术,提出了利用 JAVA 和 XML 开发程序生成器的几种方法,并对其进行了比较分析。可以预见,JAVA 和 XML 技术必将使程序生成器的开发变得更加通用、简捷和规范。

参 考 文 献

[1] Krzysztof Czarnecki, Ulrich Eisencker, Generative Programing: Methods, Tools, and Applications, Addisionr Wesly, 2000.

[2] Jag Sodhi, Prince Sodhi, Software Reuse: Domain Analysis and Design Process, McGraw Hill, 1999.

[3] J. W. Cooper, Java Design Patterns: A Tutorial, Addisonr Wesley, 2000.

[4] Charles F. Goldfarb, Paul Prescord, The XML Handbook, Prentice Hall, 2001.

[5] H. Maruyama, K. Tamura, N. Uramoto, XML and Java, Developing Web applications, Addisionr Wesley, 1999.

[6] J. Craig Cleaveland, Janet A. Fertig, and George W. Newsome, “Dividing the Software Pie,” AT&T Technical Journal, Vol. 75, No. 2, March 1996, pp. 8 ~ 19.

[7] J. Craig Cleaveland, “Building Application Generators,” IEEE Software, July 1988; 曾发表在 Domain Analysis and Software System Modeling by Prieto-Diaz and Arango, 1991.