

一种基于 XML 的构件组装编译技术研究

田 宇, 陈松乔

(中南大学 信息科学与工程学院, 湖南 长沙 410083)

摘 要: 基于构件的开发方法的目的是将分布在 Internet 上的构件, 自动或半自动组装成为一个新的粒度更大的构件或者一个新的软件, 加快系统的开发周期, 降低开发成本, 提高系统的灵活性、可靠性、可扩展性和易维护性。在对构件组装编译的研究过程中, 提出一种新的基于 XML 的构件组装编译解决方案。利用 XML(Schema)对构件组装规约进行形式化描述, 采用数据绑定技术, 对用构件组装规约描述的 XML 文档进行组装编译的解析, 最后利用标准代码编译器进行编译。实例证明了该技术的可行性。

关键词: 构件组装; XML; 数据绑定; 编译

中图分类号: TP311.5

文献标识码: A

文章编号: 1673-629X(2006)09-0012-03

Research on XML - Based Component Assembly Compile Technology

TIAN Yu, CHEN Song-qiao

(College of Information Science & Engineering, Central South University, Changsha 410083, China)

Abstract: The purpose of component based software development is using the components on Internet to create a new coarse grain component or software automatically or semi - automatically. The development approach can accelerate the development cycle, reduce the development cost and improve the flexibility, reliability, expansibility and maintainability of systems. During the research on component assembly compile, a new solution for XML - based component assembly compile is proposed. This paper formally describes the XML - based component assembly specification, analyses the XML documents by using data binding technology and compiles the programming code by using standard coding compiler. The feasibility of the technology has been proved by an instance.

Key words: component assembly; XML; data binding; compile

0 引 言

Internet 的迅速发展和基于构件的软件开发方法促进了网络计算及网络化软件的发展, 基于构件的软件开发方法成为软件重用的主流技术^[1], 软件生产商将通过为用户提供有偿软件构件及其服务来获得利润。随着 Microsoft 和 Sun 公司分别提出的分布式计算平台 .Net 和 J2EE, 以及 CORBA 组件规范的出现, 这种软件开发方法已经趋向明朗, 随之而来的是在 Internet 上分布大量的软件构件以及为此软件开发方法服务的网络服务技术。如何充分利用分布在网络上的构件服务资源是软件重用研究领域的前沿课题。

国内外在构件技术方面已经取得了一定的研究成果: 北京大学软件工程研究所的青鸟工程^[2]、中科院软件研究所的信息化基础软件核心平台^[3]、上海普元的面向构件的互联网应用基础平台(EOS)^[4]、互联网实验室的《面向构

件的互联网应用基础平台研究报告》^[5]、Richard N. Taylor 等研制开发的用于 GUI 软件开发的基于消息总线的 C2 架构风格等等。基于构件的开发方法(Component Based Software Development)的目标是将分布在 Internet 上的构件, 自动或半自动组装成为一个新的粒度更大的构件或者一个新的软件, 加快系统的开发周期, 降低开发成本, 提高系统的灵活性、可靠性、可扩展性和易维护性。已有的构件组装机制以及构件组装工具, 从模块互联语言(MIL)到构件描述语言(CDL)再到构架描述语言(ADL), 均只能描述构件和连接器及其接口的语法语义、组装结构中系统的整体架构等等, 不能直接与具体实现相关, 仍需要使用适当的转换技术, 存在一定的局限性, 不能对组装编译过程提供很好的支持。

根据上述的问题, 文中提出一种新的基于 XML 的构件组装编译解决方案。方案基于 XML 的构件组装的形式化描述规范, 采用 XML(Schema)技术描述完整的构件组装结构, 包括构件、连接器、端口、消息等等, 形成可以解析编译的统一文档, 利用数据绑定技术, 对构件组装规约描述的 XML 文档进行组装编译解析, 自动生成构件组装程序源代码, 利用标准代码编译器进行编译形成应用软件的实体。

收稿日期: 2006-01-20

基金项目: 国家教育部博士点基金(20030533011)

作者简介: 田 宇(1978-), 男, 辽宁葫芦岛人, 硕士研究生, 主要研究领域为可复用构件组装技术; 陈松乔, 教授, 博士生导师, 主要研究领域为软件工程、CIMS、构件技术。

1 构件组装流程描述

构件组装流程如图1所示。

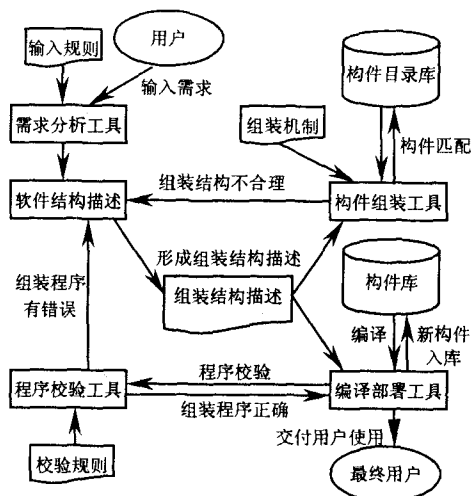


图1 构件组装流程示意图

(1)用户按照输入格式要求,根据用户的需求描述^[6],利用需求分析工具,在系统提供的输入规则引导下,逐步输入需求,形成构件组装结构描述。

(2)利用构件组装工具,根据组装结构描述,从构件库中搜索、评估并选取组装结构所需要的构件,并指导用户完成构件间连接子的设计,形成完整的构件组装结构。

(3)利用编译部署工具,根据完整的构件组装结构,对构件组装结构描述文档进行解析,生成程序原代码,再利用标准代码编译器进行编译形成应用软件的实体。

(4)利用程序校验工具,根据校验规则,对自动生成的软件实体进行测试。如果有错误或不能完全满足用户需求,那么回到步骤(2),对软件的组装结构、构件选取、连接子设计进行逐步调整,并继续步骤(3)、(4),直到得到用户最终满意的软件系统实体。

2 构件组装描述规范

形式化的软件结构描述用一套定义良好、便于扩充的符号描述软件系统^[7]。软件结构形式化描述使软件系统的组装结构和实现软件系统的软件构件完全独立,能够更好地隐蔽构件实现细节,对系统的描述层次更高,从而突出系统各部分之间的逻辑联系,大大提高软件构件复用性和组装机制灵活性。形式化的软件结构描述方法规定了软件各个部分以及组装结构的语法和语义描述方式。一般来说,语法规定了各部分之间的属性、接口、操作、参数列表等的描述格式,语义部分则是对构件功能和非功能属性的自然语言描述,没有固定的格式。

文中利用BNF范式分别给出构件、复合构件、连接子、端口、消息、构件行为、组装方案和构架这8种组装规约的形式化描述定义。

(1)构件:

$\langle \text{Component} \rangle ::= \langle \text{ComponentBasicInfo} \rangle \langle \text{ServicePorts} \rangle \langle \text{RequirePorts} \rangle \langle \text{Events} \rangle$

$\langle \text{ServicePorts} \rangle ::= \langle \text{ServicePort} \rangle \{ ; \langle \text{ServicePort} \rangle \}$

$\langle \text{RequirePorts} \rangle ::= \langle \text{RequirePort} \rangle \{ ; \langle \text{RequirePort} \rangle \}$

$\langle \text{Events} \rangle ::= \langle \text{Event} \rangle \{ ; \langle \text{Event} \rangle \}$

(2)复合构件:

$\langle \text{ComboComponent} \rangle ::= \langle \text{ComponentBasicInfo} \rangle \langle \text{Connector} \rangle \langle \text{Components} \rangle$

$\langle \text{Components} \rangle ::= \langle \text{Component} \rangle \{ \langle \text{ComboComponent} \rangle \{ ; \langle \text{Component} \rangle \} \langle \text{ComboComponent} \rangle \}$

(3)连接子:

$\langle \text{Connector} \rangle ::= \langle \text{ConnectorBasicInfo} \rangle \langle \text{ConnectorServicePort} \rangle \langle \text{ConnectorRequirePorts} \rangle$

$\langle \text{ConnectorServicePort} \rangle ::= \langle \text{ServicePort} \rangle \langle \text{CompositionMethod} \rangle$

$\langle \text{ConnectorRequirePorts} \rangle ::= \langle \text{ConnectorRequirePort} \rangle \{ ; \langle \text{ConnectorRequirePort} \rangle \}$

$\langle \text{ConnectorRequirePort} \rangle ::= \langle \text{RequirePort} \rangle$

(4)端口:

$\langle \text{ServicePort} \rangle ::= \langle \text{PortName} \rangle \langle \text{MsgID} \rangle$

$\langle \text{RequirePort} \rangle ::= \langle \text{PortName} \rangle \langle \text{MsgID} \rangle$

(5)消息:

$\langle \text{Message} \rangle ::= \langle \text{MsgID} \rangle \langle \text{Parameters} \rangle$

$\langle \text{Parameters} \rangle ::= \langle \text{Parameter} \rangle \{ ; \langle \text{Parameter} \rangle \}$

$\langle \text{Parameter} \rangle ::= [\text{in}|\text{out}] \langle \text{ParamType} \rangle \langle \text{ParamName} \rangle$

(6)构件行为:

$\langle \text{Behavior} \rangle ::= \langle \text{ID} \rangle \langle \text{ComponentLoc} \rangle \langle \text{Message} \rangle$

$\langle \text{ComponentLoc} \rangle ::= \langle \text{ComponentID} \rangle \langle \text{HostName} \rangle$

(7)组装方案:

$\langle \text{CompositionMethod} \rangle ::= \langle \text{CompositionExpress} \rangle \langle \text{Behaviors} \rangle$

$\langle \text{Behaviors} \rangle ::= \langle \text{Behavior} \rangle \{ ; \langle \text{Behavior} \rangle \}$

(8)构架:

$\langle \text{Architecture} \rangle ::= \langle \text{Components} \rangle \langle \text{Connectors} \rangle$

$\langle \text{Connectors} \rangle ::= \langle \text{Connector} \rangle \{ ; \langle \text{Connector} \rangle \}$

在组装规约的形式化描述中,构架由构件集合 Components 以及连接子集合 Connectors 构成,其中构件可以是原子构件 Component,也可以是复合构件 ComboComponent。连接子 Connector 的服务端口 ConnectorServicePort 中定义了连接子下挂的协作构件之间的组装方案 CompositionMethod。组装方案 CompositionMethod 主要由组装表达式 CompositionExpress 与构件行为集合 Behaviors 构成。其中 $\text{CompositionExpress} = \sum_{op \in \{ \downarrow, \vee, *, \| \}} \text{OP}(\text{BehaviorList})$, 表达式中的 BehaviorList 中表示一个、两个或多个服务构件的行为标识构成的列表,各个行为标识之间用“,”号隔开。 $\text{OP} \in \{ \downarrow, \vee, *, \| \}$, 分别代表顺序组装 (SEQ)、分支组装 (CHO)、循环组装 (ROL)、并行组装 (PAR) 这四种组装操作,即 BehaviorList 中的多种行为通过某种 OP 组装操作,可以形成复合行为。在组装方案 CompositionMethod 中,行为集合 Behaviors 对每个参与组装的构件行为进行了具体描述,包括构件行为的标识 ID、构件定位 ComponentLoc 以及驱动该行为的消息定义

Message。

然后,文中利用 XML 标记语言的文档格式化标准 (Schema),对构件和组装方案等 8 种组装规约描述建立统一的 XML Schema 文档(*.XSD)。在组装实例研究中,以实例建立相应的 XML 文档(*.XML)。

下面,文中以构件的规约描述为例,建立相应的 XML Schema 文档(Component.xsd)。

```
<? xml version="1.0" encoding="UTF-8"? >
<xsd:schema xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<xsd:element name="Component">
<xsd:complexType>
<xsd:sequence>
<xsd:element name="ComponentBasicInfo" type="xsd:string"/>
<xsd:element name="ServicePort" type="xsd:string"/>
<xsd:element name="RequirePort" type="xsd:string"/>
<xsd:element name="Event" type="xsd:string"/>
</xsd:sequence>
</xsd:complexType>
</xsd:element>
</xsd:schema>
```

3 构件组装编译及实例研究

在构件组装编译过程中,文中根据上述得到完整的构件组装描述的 XML 文档(*.XML),采用数据绑定技术 (Castor),对 XML 文档进行解组——就是把 XML 文档转化成 Java 对象。最后利用标准代码编译器(JDK)进行编译形成应用软件的实体。通过软件测试后,若不能满足用户需求,又可以通过编组把 Java 对象转化成 XML 文档。直到完全满足用户需求。

XML 数据绑定(Data binding)是一种处理 XML 文件的技术,它可以从应用程序读取 XML 文件的内容,甚至新增、修改、删除 XML 文件内某笔数据。这样一来,XML 文件将可以作为不同应用系统之间交换数据的媒介。数据绑定技术是通过 Marshalling 技术来处理 XML 文件内容的。Marshalling 技术分为两部分,即 Marshalling:将 Java 对象的内容转换成 XML;Unmarshalling:将 XML 文件内容转换成 Java 对象。由此可以提高组装编译的灵活性和可维护性。

通过分析一个实例来说明该构件组装编译技术的实际应用,这个例子结合了 JSP,Servlet,EJB,DB 等技术。在这个应用中,实现一个学生成绩管理系统。该系统包含 6 个服务复合构件,分别完成成绩录入、成绩查询、成绩修改、成绩删除、成绩数据分析、成绩打印功能。

根据构件规约描述建立的 XML Schema 文档(Component.xsd),文中以成绩数据分析构件为例,建立相应实例的 XML 文档(Component.xml)。

```
<? xml version="1.0" encoding="GB2312"? >
```

```
<Component><!--数据运算构件-->
```

```
<ComponentBasicInfo>Integrate</ComponentBasicInfo><!--合并数据表-->
```

```
<ServicePort>Table</ServicePort>
```

```
<RequirePort>Table</RequirePort>
```

```
<Event>Union</Event>
```

```
</Component>
```

```
<Component><!--数据运算构件-->
```

```
<ComponentBasicInfo>Average</ComponentBasicInfo><!--数据表求平均-->
```

```
<ServicePort>Table</ServicePort>
```

```
<RequirePort>Table</RequirePort>
```

```
<Event>Average</Event>
```

```
</Component>
```

```
<Component><!--数据运算构件-->
```

```
<ComponentBasicInfo>Rank</ComponentBasicInfo><!--数据表排序-->
```

```
<ServicePort>Table</ServicePort>
```

```
<RequirePort>Table</RequirePort>
```

```
<Event>Rank</Event>
```

```
</Component>
```

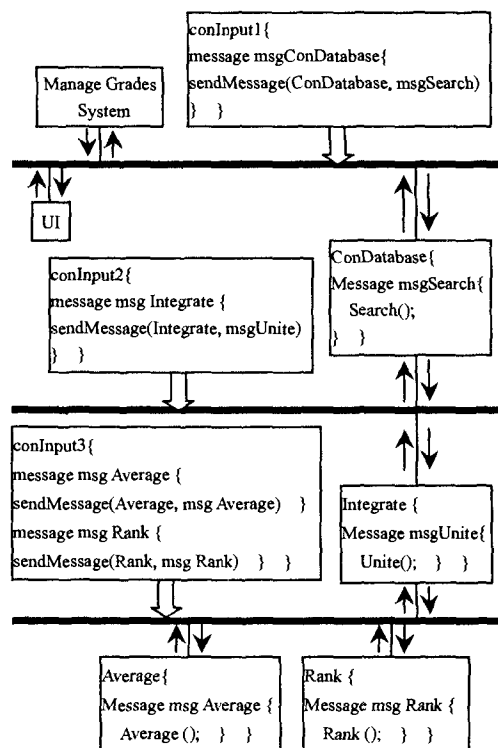


图 2 构件交互示意图

图 2 是实例中 C2 构架风格的组装结构图(构件交互示意图)。在实例的组装结构中,包含 5 个构件:UI,conDatabase,Integrate,Average,Rank;3 个连接器:conInput1,conInput2,conInput3。其中构件 Integrate,Average,Rank 和连接器 conInput1,conInput2,conInput3 共同组成复合构件——成绩数据分析构件。这些构件通过接口连接到构架,保证了系统的灵活性和可扩展性。

(下转第 17 页)

er,url);

destinationDB.insertIntoDB(sql);

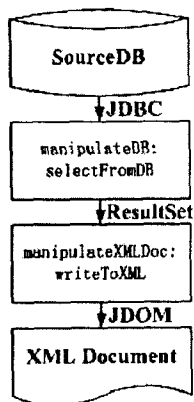


图2 读取数据写入 XML 文件

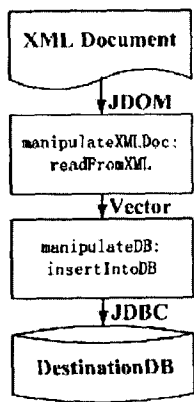


图3 读取 XML 文件 写入数据

4 应用实例

本例中使用 JBuilder9 作为开发平台,实现了不同机器之间从 Access 2003 到 SQL Server 2000 数据库之间的数据移植。在 Access 中建立一个 student 数据库,并在其中建立一名为 student 的表,并输入若干数据。

先执行 readData 类生成 XML 文档,其格式如下:

```
<? xml version="1.0" encoding="GBK"? >
<Students>
  <Student>
    <Name>ZHE</Name>
    <ID>2005786</ID>
    <Sex>M</Sex>
    <City>Jiangsu</City>
    <Telephone>05163889086</Telephone>
  </Student>
  .....
</Students>
```

再将此 XML 文档拷入目标机器,执行 writeData 类将

数据存入 SQL Server 数据库,数据移植操作即完成。

5 小结

文中尝试使用 JDBC 和 JDOM 技术实现了一个用 XML 文件作为中介进行数据移植的应用,而且通过一个实例演示了如何使用 JDOM 对 XML 文件进行读写操作。

当今世界的数量正在飞速膨胀,应用系统也在不断增大,应用与数据库之间、数据库与数据库之间的数据交换日益频繁,数据格式的不统一已经成为了软件业发展的障碍,因此将杂乱无章的数据处理成格式规范的 XML 格式文件,将为数据的交换和处理提供极大的方便。Java 作为现在的一种主流开发语言,其强大的功能加上 XML 在数据表示方面的优势,极大地提高了应用的易用性和可移植性。因此有理由相信,Java 与 XML 的协同运用在将来的系统开发中将会得到更为广泛的应用,同时 JDOM 作为一种专为 Java 语言设计的处理 XML 的方式,将会更加完善成熟。

参考文献:

- [1] 陶以政,唐定勇,何铁宁,等.基于 Java 和 XML 技术的异构信息系统数据集成框架应用研究[J]. 计算机应用研究, 2004(5):38-40.
- [2] McLaughlin B. Java 与 XML[M]. 北京:中国电力出版社, 2004.
- [3] Hunter J. JDOM and XML Parsing[EB/OL]. <http://www.oracle.com/technology/oramag/oracle/02-sep/o52jdom.html>, 2002.
- [4] Hunter J, McLaughlin B. Easy Java/XML integration with JDOM[EB/OL]. <http://www.javaworld.com/javaworld/jw-05-2000/jw-0518-jdom.html>, 2000.
- [5] Biggs W, Evans H. Simplify XML programming with JDOM [EB/OL]. <http://www-128.ibm.com/developerworks/java/library/j-jdom>, 2001.

(上接第 14 页)

4 结束语

基于构件的软件开发方法(Component Based Software Development)有利于在软件开发中减轻重复劳动,提高软件的开发效率、质量和可维护性。

文中提出的基于 XML 的构件组装编译的解决方案,将构件组装机制中的组装描述进一步完善,对构件组装编译过程提供更好的支持,提高了软件的开发效率,增强了系统的灵活性、可扩展性和易维护性。改善和开发新的构件组装技术规范具有重要的理论和应用意义。

参考文献:

- [1] 王 斌,王建新,张尧学,等.一种基于多 Agent 的 Internet 上 JavaBean 构件挖掘方法[J]. 小型微型计算机系统,

2003,24(12):43-52.

- [2] 杨美清,梅 宏,李克勤.软件复用与软件构件技术[J]. 电子学报,1999,27(2):68-75.
- [3] 中科院软件研究所.中科院软件研究所网站[DB/OL]. <http://www.ios.ac.cn>. 2003.
- [4] 普元公司.普元公司网站[DB/OL]. <http://www.primeton.com>. 2004.
- [5] 互联网实验室.互联网实验室网站[DB/OL]. <http://www.chinalabs.com>. 2004.
- [6] 窦郁宏,陈松乔.程序挖掘中需求描述的研究[J]. 计算机工程与应用,2002(10):53-56.
- [7] 任洪敏,钱乐秋.构件组装及其形式化推导研究[J]. 软件学报,2003,14(6):1066-1074.