基于 XML 的可定制的文档数据及行为模型

张 硕 顾毓清
(中国科学院软件研究所 北京100080)

摘 要 为了降低文档管理软件的维护费用,本文基于 XML 技术,通过增加软件各部分的独立性,把维护需求的改动限制在一定范围内,最大限度地减少了代码改动。

关键词 XML,独立性,行为,节点

Flexible Data-Document and Behavior Model Based on XML Technology

ZHANG Shuo GU Yu-Qing
(Institute of Software, Chinese Academy of Sciences, Beijing 100080)

Abstract For reducing maintenance cost in document-management system, based on XML technology this paper introduces a technique which keeps independence between the parts of the software, so limits the changes caused by maintenance cases and accordingly lower the cost of the code changes effectively.

Keywords XML, Independence, Behavior, Node

随着计算机在数据存贮领域的不断发展,用户对灵活配置数据存贮模型,以及对能提供二次开发和动态配置的应用程序的需求越来越广泛。本文基于 XML 技术,用 Java 语言对以上需求提出一个完整的解决方案。

1 案例

在 CMM 的实施过程中会遇到一个共同的问题:一个管理 CMM 文档的应用程序开发结束后,因为各个项目的实际需求不同,或因为 CMM 的成熟度的不断提高,需要对应用程序做出不间断的维护。这种维护工作包括文档存贮模型的变化以及对应用程序的功能做出增加、删除和修改。

以软件生存周期的第一步——需求分析为例,可以定义一个简单的文档模板,如表1。在这里需求分析文档是由项目经理负责制定,由主管经理和质量控制经理负责审阅。在 CMM 实施的最初阶段这个文档模板可能已经满足要求了,但随着 CMM 成熟度的不断提高,也许需要加入需求提出者的记录,也许需要丰富软件的功能,比如在审阅以后把审阅结果和审阅意见以电子邮件的形式发到项目经理的信箱里。这些需求的改动都会增加维护成本,怎样才能降低文档管理软件的维护和后期开发费用是本文探讨的重点。

表1 需求分析文档模板

主体内容			审阅信息			
版本号	时间	需求描述	主管经理审阅标识	主管经理意见	质量控制经理审阅标识	质量控制经理意见

2 XML 文档

W3C 对 XML 作了如下描述[1]:可扩展标记语言,缩写为 XML,描述了一类称为 XML 文档的数据对象,同时也部分地描述了处理这些数据对象的计算机程序的行为。与 HTML 相比,XML 的优势之一在于其可扩展性。XML 的定义只是由框架语法组成。当创建一个 XML 文档时,不必使用有限的预定义元素集,而是创建自己的元素[2]。当 XML 和 Java 结合使用时,XML 为 Java 的语义(行为)提供了一种统一的语法,简单地说就是,这意味着一个开发者可以创建不同的数据类型,并使得这些数据在 Java 程序的解释下表现出不同的行为,并且可以重用和修改这些描述。既然 Java 和 XML 都是可移植的标准,这两种技术结合的结果也是可移植的,可以重用数据和移植行为。

3 系统设计

由于系统的设计目标之一是要降低软件的维护和后期开发费用,本文从三个角度分析这个问题。第一,文档的内容的改变;第二,文档用户接口(UI)的改变;第三,文档行为的改变。如果用户的这些需求都能最小限度地修改代码,就能很好地控制软件的维护和后期开发费用。

3.1 独立性

在系统设计中保持各部分的独立性是当改变发生时把改变限制在最小范围的有效方法。在这里引入三种独立性,分别是:数据和数据的用户接口的独立性,语法和语义的独立性,文档行为和系统架构的独立性。

数据/数据的用户接口的独立性就是由于数据和数据显示信息的多对一的关系,把数据的显示信息在 XML 文档中独立出来,这样当数据发生改变时就不会影响数据的显示。同理,当数据的显示模式发生改变时也不会影响数据本身。

语法/语义独立性就是文档管理系统在逻辑上分为两层。XML 作为系统的语法层,定义文档的组织规则;Java 程序作为系统的语义层,解释语法规定的原则,实现行为框架。语法层在设计阶段根据需求预定义尽可能完备的系统行为框架,灵活的文档组织形式,并在语义层实现。这样当文档模板需要扩充的时候就可以把改动局限在 XML 文件里,Java 应用程

张 硕 硕士研究生,主要研究领域为软件工程;顾毓清 博士生导师,主要研究领域为软件工程。

序保持一定的独立性。

文档行为/系统架构的独立性就是文档自身包含文档行为的描述，Java 应用程序实现基本功能和对文档行为提供架构上的支持。文档的行为由一个个插件具体实现。

3.2 XML 文档模板的结构

(1)总体架构　由于文档数据和文档的显示属性以及文档管理系统的行为是紧密相关的，简单起见在本文中把文档的实际数据和文档的显示信息和行为在同一个 XML 文件(文档模板)里定义。文档模板是基于 XML 的树状结构，总体架构如图1。Body 域代表文档的根，包括三个分支，分别为 UIInfo、DataTemplate、DataInfo。UIInfo 描述系统的用户界面，DataTemplate 为数据生成模板，DataInfo 是文档所存储的实际数据。

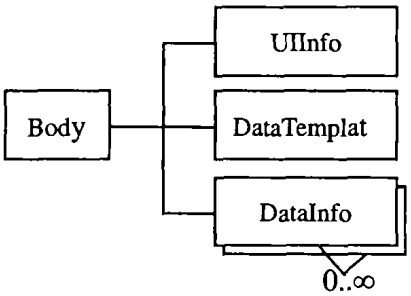

图1　总体架构图

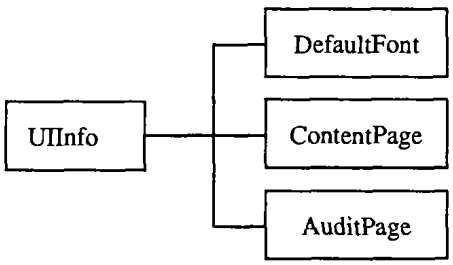

图2　UIInfo 节点结构图

(2)UIInfo 节点　UIInfo 节点如图2所示，这个分支用来描述文档管理系统的用户界面，对实际存储的数据节点进行进一步描述，简单地说就是描述数据的各种和显示相关的属性和数据的外部组织形式。通过 UIInfo 和 DataInfo 的映射关系，把数据的存储的内模式和面向用户的外模式独立开来。

(3)DataTemplate 节点　DataTemplate 是数据节点的模板，当用户需要追加一个新的数据时，通过读取 DataTemplate 的描述，生成一个只含有预定义值的数据节点。在本文中 DataTemplate 分支的内容就是一个含有预定义值的 DataInfo 节点。

(4)DataInfo 节点　DataInfo 节点如图3所示，描述数据的内部组织形式。根据案例中需求分析文档的业务需求，文档分为两个部分。第一个部分是需求的内容以及相关信息，包括：版本号、时间、需求描述。第二部分是审计信息，包括：主管经理审阅标识、主管经理审阅意见、质量监督经理审阅标识、质量监督经理审阅意见。

在这里的 ContentPage、AuditPage 分别对应 UIInfo 分支的 ContentPage、AuditPage。UIInfo 对 DataInfo 的数据提供外模式的描述。在 Body 里存在多个 DataInfo 节点，VerNum 为主码，作为不同的需求记录的唯一标识。而 UIInfo 节点和 DataInfo 节点是一对多的关系，最大限度地减少了文档的冗余。

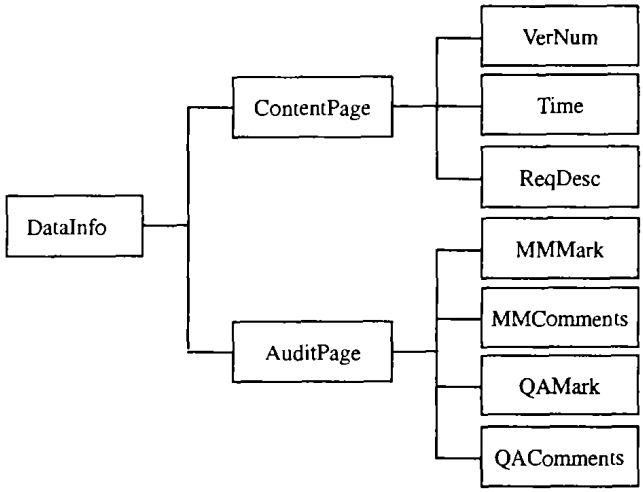

图3　DataInfo 节点结构图

4　系统实现

4.1　数据节点

关键字 unique 作为 DataInfo 节点的属性指定了它的一个子节点作为主码。这里是 VerNum，表示需求分析的版本号是需求文档的唯一标识。而版本号、时间、需求描述是内容页(ContentPage)的子节点，表示版本号、时间、需求描述在存储的内模式上分为一组，和审计信息独立开来。

```
〈DataInfo unique="VerNum"〉
  〈ContentPage〉
    〈VerNum〉1.0〈/VerNum〉
    〈Time〉2003/07/20〈/Time〉
    〈ReqDescription〉This is a test.〈/ReqDescription〉
  〈/ContentPage〉
  〈AuditPage〉
    〈MMMark〉true〈/MMMark〉
    〈MMComments〉Good job.〈/MMComments〉
    〈QAMark〉false〈/QAMark〉
    〈QAComments/〉
  〈/AuditPage〉
〈/DataInfo〉
```

4.2　外模式节点

UIInfo 节点为系统的外模式节点。可以看到数据节点(DataInfo)的子节点和外模式(UIInfo)的子节点一一对应。也就是说 UIInfo 为数据节点一一描述了显示模式。首先关键字 type 描述了内容页(ContentPage)的总的显示模式——Caption_Page。Caption_Page 表示一个内容页节点是由一个带标题的页组成，所以它的子节点分别是 Caption 和 Page。Caption 节点的值是"内容"，它的值就出现在用户界面上，为用户提供一个简单的索引。Caption 和 Page 节点在 Java 语义层被解释为一个标题为"内容"的 JTabbedPane 组件的一页。而 JTabbedPane 组件的 Tab 就作为 Page 的子节点的容器。

VerNum 的属性 type 的值为 Caption_Content，相应地对应 Caption 和 Content 两个子节点。在这里的 Caption 和 Content 分别在 Java 的语义层解释为 JLabel 和 JTextPane，表示一个带有说明文字的可由用户填写的域。

```
〈ContentPage type="Caption_Page"〉
  〈Caption〉内容〈/Caption〉
  〈Page〉
    〈VerNum type="Caption_Content"〉
      〈Caption height="30" width="80" x="20" y="20"〉
        〈Font color="red" name="宋体" size="20" style="
        PLAIN"/〉
        版本号:
      〈/Caption〉
      〈Content height="30" width="80" x="120" y="20"〉
        〈Action name =" FocusListener" source =" FXML-
        Doc.plugin.VerNumCheckListener"/〉
      〈/Content〉
    〈/VerNum〉
```

```
    〈Time type="Caption_Content"〉
      ……
    〈/Time〉
      ……
  〈/Page〉
〈/ContentPage〉
```

4.3 动态文档的实现

如前所述，由于数据和数据的用户接口的独立性，当文档的内部结构需要更改的时候，比如增加一个需求的提出者，系统只需要修改 XML 文件就可以达到要求。

这里需要对应修改三个节点，首先是在 DataInfo 和 DateTemplate 的子节点 ContentPage 下增加个名为 Announcer 的子节点。Announcer 节点的数据就是系统增加的需求提出者。其次是修改 UIInfo 节点，同样在 ContentPage 的 Page 节点下增加 Announcer 节点，并调整 Announcer 节点和其余节点的属性，使"提出者"这个数据位于页面的合适位置。

当文档的内部结构没有改变仅仅需要调整数据的显示属性时也可以通过修改 XML 文件来实现。这时就要修改 UIInfo 节点，UIInfo 节点包括系统的默认字体，每一个数据用什么方式显示，数据所在位置，数据的外部组织形式等等信息。比如，我们在"需求文档生成时间"的位置想去掉"时间："这个标识，我们可以把相应的节点改为：

```
〈Time type="Content"〉
  〈Content height="30" width="160" x="120" y="60"〉
    〈Action name="FocusListener" source="FXML-
    Doc.plugin.DateCheckListener"/〉
  〈/Content〉
〈/Time〉
```

4.4 动态行为的实现

由于文档行为和系统架构之间的独立性，文档自身定义自己的行为，而在应用程序目录\plugin 下的一个个 class 文件才是行为的真正实现。

文档的 Action 节点就是系统行为的定义。如 Time 的子节点：

```
〈Action name="FocusListener" source="FXML-
Doc.plugin.DateCheckListener"/〉
```

Action 节点包括两个属性 name 和 source，name 指定的是行为的类型名，在这里就是 FocusListener；source 指定的是实现具体行为的类，在这里就是 FXMLDoc.plugin.DateCheckListener。

系统的行为框架是由一个叫 ActionAssembler 类实现的。这个类负责解析 Action 节点并生成相应类的实例，并根据行为的名字绑定到指定的 UI 组件上。由于在这里用到的是 JComponent 的 addFocusListener 方法，相应的 DateCheckListener 是从 FocusListener 继承而来。

```
public static void assemble(JComponent talker,Node actionNode)
{
  //解析 Action 节点得到_actionName 和_actionClass
  parseActionNode(actionNode);
  if(_actionName.compareTo(FOCUS_LISTENER)==0)
  {
    try
    {
      //生成一个_actionClass 的类的对象
      Class newClass=Class.forName(_actionClass);
      //生成对象的实例，并和 talker 这个 UI 组件绑定起来
      talker.addFocusListener ((FocusListener) newClass.new-
      Instance());
    }
    catch …
  }
  else if …
}
```

结束语 对于文档管理系统，软件维护需要的工作量占软件生命周期的比重非常大。如何有效地降低维护费用，延长软件的使用寿命，本文基于 XML 技术通过提高软件各部分的独立性，做到软件模块高内聚、低耦合，在降低软件的维护费用方面做了一些工作。

参考文献

1 Bray T, Paoli J, Sperberg-McQueen C M. Extensible Markup Language 1.0, Feb. 1998
2 Aviram M H. XML and Java: A powerful combination
3 Elliotte Rusty Harold, XML Bible, June, 1999

（上接第156页）

在对计算结果的分析中，我们注意到运用基于信息增益的贝叶斯重叠补缺模型补到第三个属性时，形成数据集 D'，此时模型预测准确率和长期类预测准确率最高。如果需要对剩余的全部缺失数据进行补缺，那么在数据集 D' 的基础上，运用贝叶斯补缺模型预测剩余的具有缺失数据的属性，生成最终数据集 D，其结果为：模型预测准确率55.53 %，短、中、长三类的类预测准确率分别是45.4%、69.7%和29.9%，计算结果亦令人满意。

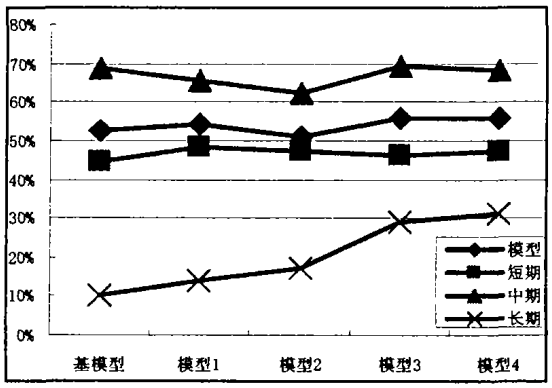

图1 模型及类预测准确率汇总图

根据表2可以画出模型及类预测准确率(图1)。可以看到，本文所提出的缺失数据处理模型不仅可以改善总体模型的预测准确率，而且能使长期类的预测率得到更大提高，其补缺收益为211%，效果十分明显。这说明对提高长期类预测准确率来说，缺失数据的处理起到了关键作用。

结束语 本文重点讨论了几种不同的缺失数据处理方法，建立了四个缺失数据处理模型，及其组合应用。通过将四种模型应用于医疗领域，预测病人住院期的长短，来说明本文所建立模型的有效性。

然而，除了信息增益之外，还有很多因素会影响补缺的性能，例如各属性中缺失数据所占比重及重要性。对本文所提出的模型进行改进，找出一个更加有效的属性补缺顺序的排列标准，还需要进一步的研究工作。

参考文献

1 Cios K J, Kurgan L A. Trends in Data Mining and Knowledge Discovery. In: Knowledge discovery in advanced information systems, Pal, N.R., Jain, L. C., Teodoresku N. eds. Springer, 2002
2 H Liu, Motoda H. Feature Extraction, Construction and Selection: A Data Mining Perspective, Kluwer Academic, Boston: MA, 1998
3 Troyanskaya O, et al. Missing value estimation methods for DNA, Bioinformatics, 2001. 520～525
4 Kantardzic M. Data Mining Concepts, Models, Methods and Algorithms, Wiley-IEEE Computer Society Pr, 2003
5 Ian H. Witten and Eibe Frank, Data Mining Practical Machine Learning Tools and Techniques with Java Implementations, Morgan Kaufmann Publishers, 2000
6 Marshel A H. Bayesian Belief Network Using Conditional Phase-type Distributions: [PhD Thesis]. University of Ulster, 2001