UIScript 的设计与实现

学　　院　软件学院

专　　业　软件工程

年　　级　2004 级

姓　　名　邓晋丰

指导教师　张坤龙

2008 年 6 月 18 日

摘　　要

如何在现代软件系统的用户界面中方便快捷的实现信息和参数的传递、保存和更改等，是现代用户界面设计人员要迫切解决的问题。传统的开发方法需要的工作量大，而且对开发人员的技术水平要求比较高，要设计实现高质量的软件产品的用户界面不是一件容易的事。

本文定义了一种新的脚本语言即UIScript，同时设计实现了一个用户界面文件的解释器即UIViewer。利用UIScript可以设计生成两种文件，Sim文件和UI（用户界面）文件。其中sim文件负责系统中参数和命令的保存，用于UI文件的输入和输出。UI文件通过UIViewer的解析生成图形化的用户界面，用于完成软件系统和用户之间的的交流。这在一定程度上提高了用户界面的开发效率。目前UIScript已经成功应用于中国水利泥沙研究所开发的水利泥沙防洪系统中。

关键词：用户界面；脚本；可视化；UIScript；UIViewer；Sim；UI

ABSTRACT

How to achieve Visualization in the transferring, storage and changes of parameters in the user interface designing procedure is an urgent task. Traditional methods not only require the heavy work, but also ask for the high skills of the designer. In a word ,it is a difficult job for reailizing a high quality user interface.

This paper defines a new scripting language that is UIScript, while achieving a interpretation machine of user interface that is UIViewer. UIScript can generated two documents, sim documents and ui document. Sim document which is responsible for the preservation of system parameters and orders is used as the input files and output files of the UI documents. UI documents generated by the analytical UIViewer graphical user interface, systems and software used for the completion of the exchange between users. To certain extent, this improve the efficiency of the development of user interface. UIScript has been successfully applied to the present China Water Conservancy Institute for the development of water conservancy silt sediment in the flood control system.

Key words： user interface;script;Visualization;UIScript;UIViewer;Sim;UI

目　　录

第一章　　绪论

1.1　用户界面

软件设计可分为两个部分：编码设计与 UI(user interface，用户界面)设计。编码设计大家都很熟悉，但是 UI 设计还是一个很陌生的词，即使一些专门从事网站与多媒体设计的人也不完全理解 UI 的意思。UI 的本意是用户界面，是英文 User 和 interface 的缩写。从字面上看是用户与界面 2 个组成部分，但实际上还包括用户与界面之间的交互关系。

在软件的发展历史中，界面设计工作一直没有被重视起来。做界面设计的人也被贬义的称为“美工”。其实软件界面设计就像工业产品中的工业造型设计 一样，是产品的重要卖点。一个友好美观的界面会给人带来舒适的视觉享受，拉近人与电脑的距离，为商家创造卖点。界面设计不是单纯的美术绘画，它需要定位使用者、使用环境、使用方式并且为最终用户而设计，是纯粹的科学性的艺术设计。检验一个界面的标准不是某个项目开发组领导的意见也不是项目成员投票的结果，而是最终用户的感受。所以界面设计要和用户研究紧密结合，是一个不断为最终用户设计满意视觉效果的过程。

当前的软件产品，整体来说 UI 方面的薄弱可谓弱中之弱。大多软件产品开发中对编码比较偏重。目前中国的软件公司开创者，多为编程技术出身，因此对程序编码的宠爱有加也就成为自然，我们应该承认，产品的设计思想、运行速度和稳定性处于核心地位。然而，作为产品用户和计算机交互核心以及产品形象的 UI 环节一直尚未得到开发者应有的重视。产品一旦投放市场，当界面遭到用户非议的时候，才会想到花上可怜的一点时间和金钱把产品做一个“美化”（仅仅是美化）。而这种敷衍最终往往难以得到用户的接受。因此说 UI 绝对不是单单对产品美化的工作，而是科学地属于工业设计的范畴。

1.2　脚本语言

脚本语言（scripting programming languages）是为了缩短传统的编写-编译-链接-运行（edit-compile-link-run）过程而创建的计算机编程语言。此命名起源于一个脚本“screenplay”，每次运行都会使对话框逐字重复。早期的脚本语言经常被称为批处理语言或工作控制语言。一个脚本通常是解释运行而非编译。

虽然许多脚本语言都超越了计算机简单任务自动化的领域，成熟到可以编写精巧的程序，但仍然还是被称为脚本。几乎所有计算机系统的层次都有一种脚本语言。如操作系统层，计算机游戏，网络应用程序，字处理文档，网络软件等。

一个脚本可以使得本来要用键盘进行的交互式操作自动化。一个Shell脚本主要由原本需要在命令行输入的命令组成，或在一个文本编辑器中，用户可以使用脚本来把一些常用的操作组合成一组序列。主要用来书写这种脚本的语言叫做脚

本语言。很多脚本语言实际上已经超过简单的用户命令序列的指令，还可以编写更复杂的程序。

1.3 用户界面脚本语言

我们定义了一种新的脚本语言——用户界面脚本语言即 UIScript。应用这种新的脚本语言开发用户界面，可以方便快捷的完成简单的图形化用户界面的开发。传统的用户界面开发所需的时间长，工作量大，对开发人员的技术要求也比较高，而应用 UIScript 开发用户界面则简单许多。我们可以通过一个实例来了解下 UIScript 界面开发中的应用。

在中国水利泥沙研究所开发的一个水利决策支持系统当中，为水利模型准备模型参数的用户界面在很大程度上是不规范的，它与水利模型密切相关，最好能表现为对话框，并且对话框至少包含文本输入框、复选框、单选框、选择框等控件。为了便于对话框的设计与实现，我们应用 UIScript 开发此类用户界面。

UIScript 基于 XML 来设计。图 1-1 给出了一段用 UIScript 写的代码。这段代码在屏幕上显示一个文本输入框和一个单选框，用户界面如图 1-2 所示。用户完成输入后，其输入数据保存在文件 params.xml 中，如图 1-3 所示。

```
<dialog output=”params.xml”>
    <input type="text" name="modelname">
        <prompt>请输入模型的名称：</prompt>
    </input/>
    <input type="radio" name="dimension">
        <prompt>请选择模型的维数：</prompt>
        <value = "1D" checked>一维</value>
        <value = "2D" >二维</value>
        <value = "3D">三维</value>
    </input>
</dialog>
```

图 1-1　　对话框的 UIScript 源码

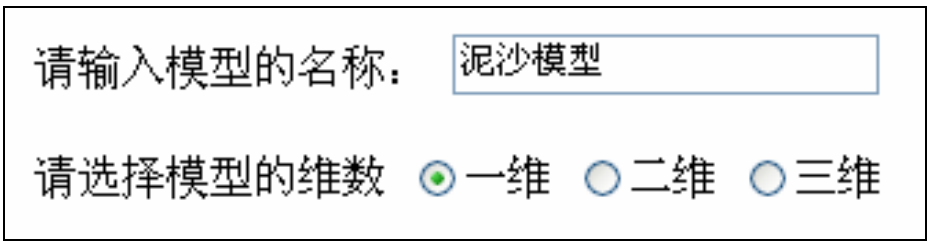

图 1-2　　对话框的用户界面

```
<param name="modelname" value="泥沙模型" />
<param name="dimension" value="1D" />
```

图 1-3　对话框的输入结果

如果需要输入的模型参数很多，那么可以提供多个对话框，并将这些对话框组织为向导。UIScript支持向导的设计和实现，例如最简单的方式就是将多个对话框简单地按出现顺序组成一个列表。

通过此例可以看出脚本语言在开发用户界面的优势，用户界面脚本语言将有广阔的发展空间。

1.4　论文的研究内容和意义

应用脚本语言开发用户界面目前还是一个新兴的研究领域,国内关于这方面的研究还非常少,但是其应用前景非常乐观。本文主要定义了 UIScript，同时提供了用户界面（UI）文件的解释器，通过 UIScript 在水利决策支持系统中的应用着手,采用从特殊到一般的研究方法，全面阐述了如何应用 UIScript 完成简单的图形用户界面的开发。论文的研究内容主要包括以下两方面。

一：定义了一种用户界面脚本语言——UIScript，并详细规定了它的语法特征，同时明确了 UIScript 可支持的窗体控件种类，包括文本输入框、复选框、单选框、选择框等控件，其中最具特色的是其对于 TreeView 控件的支持，通过支持 TreeView 控件可以使其开发的用户界面能为用户提供许多特色功能。遵循 UIScript 语法，我们设计了两种类型的文件，一种是应用于系统的图形显示方面的文件——UI 文件，一种是应用于系统参数和命令等保存传递方面的文件——SIM 文件。

二：设计实现了 UI 文件的解释器——UIViewer。仅仅定义了 UIScript 还不能完成界面的最终开发，我们需要在屏幕上显示用户期望看到的界面，这一功能可以用 UIViewer 来完成。UIViewer 是一个编译器，根据已经定义的 UIScriptr 的语法特征，它可以将 UI 文件解释成用户期望看到的图形界面，用户在此界面上进行系统各项参数的输入和输出，输入和输出的参数是以 SIM 文件的格式保存的。

通过完成以上两方面的工作，我们就可以用 UIScript 开发软件系统的用户界面。在水科院开发的水利决策支持系统中，我们用 UIScript 代替了传统的界面开发手段。实践表明，其出色的完成了系统间各个模块的通讯,传递参数、命令等工作。而且在开发界面的过程中由于其语法简单,结构清晰,所以开发起来相对比较容易.另一方面,采用了图形化用户界面的方式,改变了以前在系统中用命令行传递的方法,使操作人员的工作效率大大提高,同时减少了其出错的概率。

UIScript 将在未来的用户界面开发中发挥巨大的作用。

1.5 论文的内容结构

论文正文共五章，其内容结构是这样安排的：

第一章：文献综述主要是说明 UIScriptr 概念，介绍了论文的主要内容和研究意义，最后说明了论文的组织结构。

第二章：首先说明了 XML 语言的特点和使用方法，以及系统相关的 HTML 表单知识和 WPF 技术。最后定义了 UIScript 的语法结构，介绍了遵循 UIScript 语法结构设计的两种类型的文件——SIM 文件和 UI 文件，然后给出了 UI 文件经 UIViewer 解析生成用户界面的过程。

第三章：UIViewer 系统分析及设计，本章首先介绍了 UIViewer 系统中的需求关系，用例视图和模块划分，再说明了程序的运行流程，详细的界面分析，特征介绍以及操作方法，最后测试几个实例。

第四章：UIViewer 系统的实现，介绍了实现系统所用的几个关键技术,部分程序的源码详细说明以及功能实现的具体过程。同时详细说明了 TreeView 组件在系统中的应用。最后说明了 UIViewer 系统的另一特色，即可以做为一个大的软件系统的一个插件子系统。

第五章：总结，说明了 UIScript 的研究内容和意义，对系统的不足进行了了说明，同时对 UIScript 在未来用户界面开发中的应用做了一个展望。

第二章　UIScript 相关技术

2.1　XML

XML 即 Extensible Markup Language（可扩展标记语言）的缩写。XML 实际上是 Web 上表示结构化信息的一种标准文本格式，它没有复杂的语法和包罗万象的数据定义。XML 同 HTML 一样，都来自 SGML(标准通用标记 语言)。SGML 是一种在 Web 发明之前就早已存在的用标记来描述文档资料的通用语言。但 SGML 十分庞大且难于学习和使用。鉴于此，人们提出了 HTML 语言。但近年来，随着 Web 应用的不断深入，HTML 在需求广泛的应用中已显得捉襟见肘，有人建议直接使用 SGML 作为 Web 语言。但 SGML 太庞大了，学用两难尚且不说，就是全面实现 SGML 的浏览器也非常困难。于是 Web 标准化组织 W3C 建议使用一种精简的 SGML 版本——XML。XML 与 SGML 一样，是一个用来定义其他语言的元语言。与 SGML 相比，XML 规范不到 SGML 规范的 1/10，简单易懂，是一门既无标签集也无语法的新一代标记语言。

XML 继承了 SGML 的许多特性，首先是可扩展性。XML 允许使用者创建和使用他们自己的标记而不是 HTML 的有限词汇表。这一点至关重要，企业可以用 XML 为电子商务和供应链集成等应用定义自己的标记语言，甚至特定行业一起来定义该领域的特殊标记语言，作为该领域信息共享与数据交换的基础。其次是灵活性。HTML 很难进一步发展，就是因为它是格式、超文本和图形用户界面语义的混合，要同时发展这些混合在一起的功能是很困难的。而 XML 提供了一种结构化的数据表示方式，使得用户界面分离于结构化数据。所以，Web 用户所追求的许多先进功能在 XML 环境下更容易实现。

第三是自描述性。XML 文档通常包含一个文档类型声明，因而 XML 文档是自描述的。不仅人能读懂 XML 文档，计算机也能处理。XML 表示数据的方式真正做到了独立于应用系统，并且数据能够重用。XML 文档被看作是文档的数据库化和数据的文档化。 除了上述先进特性以外，XML 还具有简明性。它只有 SGML 约 20%的复杂性，但却具有 SGML 功能的约 80%。XML 比完整的 SGML 简单得多，易学、 易用并且易实现。另外，XML 也吸收了人们多年来在 Web 上使用 HTML 的经验。XML 支持世界上几乎所有的主要语言，并且不同语言的文本可以在同一文档中混合使用。所有这一切将使 XML 成为数据表示的一个开放标准，这种数据表示独立于机器平台、供应商以及编程 语言。它将为网络计算注入新的活力，并为信息技术带来新的机遇。目前，许多大公司和开发人员已经开始使用 XML，包括 B2B 在内的许多优秀应用已经证实了 XML 将会改变今后创建应用程序的方式。

2.2 HTML 表单

HTML 表单可以用来在网页中发送数据,特别是经常被用在联系表单——用户输入信息然后发送到 Email 中。表单本身是没有什么用的。这需要编一个程序来处理输入表单中的数据。如果使用网络服务器来放置 HTML，能开发一个服务器端的程序使一个发送到 Email 的表单工作。实际用在 HTML 中的标签有 form、input、textarea、select 和 option。表单标签 form 定义的表单里头，必须有行为属性 action，它告诉表单当提交的时候将内容发往何处。可选的方法属性 method 告诉表单数据将怎样发送，有 get（默认的）和 post 两个值。常用到的是设置 post 值，它可以隐藏信息（get 的信息会暴露在 URL 中）[2]。所以一个表单元素看起来如下所示：

<form action="processingscript.php" method="post"> </form>

表单是一个能够包含表单元素的区域。表单元素是能够让用户在表单中输入信息的元素（比如文本框，密码框，下拉菜单，单选框，复选框等等）。表单是用<form>元素定义如图 2－1 所示。

```
<form>
<input>
<input>
</form>
```

图2-1　　表单元素

其中最常用的表单标签是<input>标签。Input的类型用type属性指定。最常用的input类型解释如下：文本框.在表单中，文本框用来让用户输入字母、数字等等。代码如图2－2所示。

```
<form>
First name:
<input type="text" name="firstname">
<br>
Last name:
<input type="text" name="lastname">
</form>
```

图2-2　　表单代码

在浏览器中显示如图2－3所示。

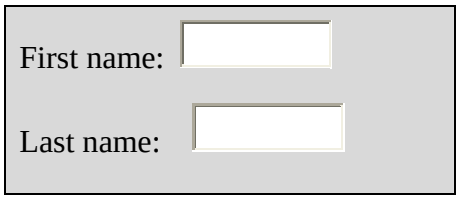

图2-3　浏览器中显示界面

注意，表单本身并不可见。另外，在多数浏览器中，文本框的宽度默认是20个字符。单选按钮:当需要用户从有限个选项中选择一个时，使用单选按钮代码如图2－4所示。

```
<form>
<input type="radio" name="sex" value="male">Male
<br>
<input type="radio" name="sex" value="female">Female
</form>
```

图2-4　单选按钮代码

在浏览器中显示如图2－5所示。

图2-5　单选按钮显示界面

选项中只能选取一个。复选框:当需要用户从有限个选项中选择一个或多个时，使用复选框。代码如图2－6所示。

```
<form>
<input type="checkbox" name="bike">
I have a bike
<br>
<input type="checkbox" name="car">
I have a car
</form>
```

图2-6　复选框代码

在浏览器中显示如图2－7所示。

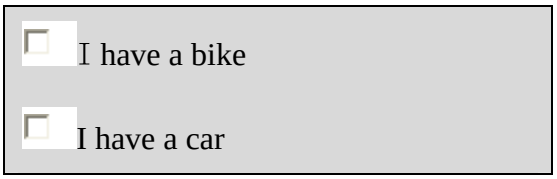

图2-7　复选框界面

表单的action属性和提交按钮:当用户点击提交按钮的时候，表单的内容会被提交到其他文件。表单的action属性定义了所要提交到的目的文件，该目的文件收到信息后通常进行相关的处理。代码如图2－8所示。

```
<form name="input" action="html_form_action.asp" method="get">
Username:
<input type="text" name="user">
<input type="submit" value="Submit">
</form>
```

图2-8　　提交按钮代码

在浏览器中显示如图2－9所示。

图2-9　　提交按钮界面

以上介绍了HTML 表单的基本知识,只有在实践中一步步探索才能逐渐掌握其精髓.在UIScript 系统的实现中,应用了其中的一些知识,在参数传递等环节中HTML Form提供了很大的帮助。

2.3　WPF 技术

GUI编程的发展历程中，伴随着Windows视窗操作系统出现的GDI技术，把GUI编程引入了一个新的阶段。GDI技术统治了近20年。随着应用要求的提高，Misrosoft提供了增强的GDI+技术。它在支持渐变功能、象素透明度等等方面进行显著的改进来弥补GDI技术在这种方面的不足。但是，运行GDI和GDI+时所使用的界面元素仍然由操作系统API提供。另一方面，Microsoft推出了新的开发技术平台.Net。在这个新的框架体系中，置入了新的Windows Forms和Web Forms两个 GUI编程应用，分别拥有传统的桌面应用程序和网页开发。而且.NET框架内置了大量常用的基本元素来提高界面编程的效率。所有这些变化都短短的几年

时间完成，这一切都预示着GUI编程正在经历一个快速发展的时期。虽然Windows Forms 和GDI+等已能满足几乎所有的应用，但随着硬件技术的飞速发展，用户对视觉体验的要求也更高。因此，开始出现一些用DirectX技术封装实现的高效GUI程序库。毫无疑问，应用DirectX技术实现的GUI将会更绚、更快，但它们的开发成本也是不可小视的。正是在这样的环境下，Microsoft的Windows Presentation Foundation技术应运而生[3]。

WPF作为.NET3.0的核心组件之一，她运用Direct3D技术提供了一套全新的GUI编程框架，并在此基础上提供了更加丰富的窗口元素。WPF打破了传统的GUI 实现式，分离了界面元素的外观和行为。例如，按钮的点击行为和她的外观无关，技术按钮的内容是一张图片，该按钮也能完美的响应点击事件。同时，WPF利用时下最广泛的XML技术，引入了一种新的语言Xaml。Xaml是一种声明式的界面描述语言，通过它我们可以直接用简单的标记代码来实现界面。最为重要的是，Xaml语言帮助我们实现了程序界面和应用逻辑的分离。我们可以用Xaml实现界面，而使用C#或者VB.NET实现应用逻辑，两者可以实现真正的无缝合成。WPF(Windows Presentation Foundation)在框架上彻底改变了它的前辈WinForm架构设计。如果说WinForm多少还保留着Win32或者MFC的结构，那么在WPF中我们看到的是完全不同的新的形式。以往的平台中，每个可视化控件只负责自己在屏幕上所占据的范围，这使得要实现特殊的效果变得很困难。WPF把整个窗口看成一个整体，窗口上的每个控件可以在屏幕的任何位置进行绘画，不再受控件本身范围的限制。最后一点，WPF中引入了XAML的标记语言,微软WPF设计软件Microsoft Expression Design就可以让美工设计出非常酷的WPF界面，但是这些设计的产生的结果文件是XAML文件[4]。

2.4 UIScript

论文定义了一种新的用户界面脚本语言——UIScript，其支持多种窗体控件的显示，包括包括文本输入框、复选框、单选框、选择框、TreeViwe 等控件。应用 UIScript,用户可以按归照一定的规则定义其所产生的 UI 文件和 SIM 文件。这两个文件是在 UIScript 的体系中非常重要。UI 文件定义之后才可心应用 UIViewer 解释器进行解析生成用户界面，可以说 UI 文件决定了整个界面的显示方式和文本传输方式。UI 文件编写的好坏直接决定着用户界面设计的好坏。

我们设计实现了 UI 文件的解释器——UIViewer，此程序用 C#语言编写，它按照 UIScript 的语法，对 UI 文件进行解释，对应生成 UI 文件中代码所要显示的用户界面，将系统需要的控件按照一定的布局显示在屏幕上。生成用户界面后，用户可以在此界面上进行程序所需参数的输入，输入完成后运行相应的程序处理数据，之后我们可以将处理过的数据保存在 SIM 文件中。

UI 文件保存的是用户界面显示信息和参数输入传递方式，及整个界面的布

局，而 SIM 文件保存的是系统所需输入的参数及经过系统处理后输出的参数。通过定义这两种不同类型的文件，我们可以将程序所需数据和程序的界面分隔开来，方便操作人员对整个系统的使用和管理。下面我们来看一下在水利泥沙防洪系统中是怎么应用 UIScript 来设计人机交互界面的。

首先定义 SIM 文件的语法，规则很简单,为方便起见,其定义规则与 XML 的语法定义规则相似.如下举例说明.

```
<Simulation>
  <Model>
    <MID name="123456789" />
  </Model>
  <Data>
    <Parameter name="input" value="C:\Documents and Settings\win\ 桌 面 \wrdssp\project\hermite\hermite.in.dfs0" />
    <Parameter name="output" value="C:\Documents and Settings\win\ 桌 面 \wrdssp\project\hermite\jfdslaf.dfs0" />
    <Parameter name="points" value="4" />
    <Parameter name="method" value="1" />
  </Data>
</Simulation>
```

这是一个典型的 SIM 文件，我们可以用文本编辑器打开并编辑它,编辑好后保存为一个以. SIM 为后缀的文件。我们可以将此文件分为文件序言 （prolog）和文件主体两个大的部分。这两大部分同在一个根结点之间,即 Simulation 为根结点。

根结点后的位于 MODEL 结点之间的是 SIM 文件固有的文件信息,那 MID ,MID 是一个工程中所有 SIM 文件的识别信息,每个 SIM 文件都有唯一的一个 MID 与这对应,这就方便了以后在查找执行过程中,只要输入 ID 号就能区分 SIM 文件.所有 SIM 文件的其余部分都是属于文件主体，SIM 文件的内容信息存放在此。我们可以看到，文件主体是由开始的〈Data〉和结束的〈/Data〉控制标记组成，这个称为 SIM 文件数据部分的“根元素”；〈Data〉是作为直属于根元素<simulation>下的“子元素”；在〈Data〉这对标记中存放的是 SIM 文件的数据部分。包括各个数据的存贮位置，如本例 Input 结点界面所获得的数据是存在 C:\Documents and Settings\win\桌面\ wrdssp\project\hermite 路径中。同时还给出了文件的拓展名，标志着用哪种程序可以打开编辑此文件。同理还有 Output 结点和 Parameter 结点的数据信息。

接着给出一个ＵＩ文件的定义规则，看下面的实例。

```
<UIscript>
    <T value="123">
        <Tree name="Input" text="Input" path="Input">
        </Tree>
        <Tree name="Parameters" text="Parameter" path="Parameters">
        </Tree>
        <Tree name="Output" text="Output" path="Output">
        </Tree>
    </T>
    <Input>
        <GroupBox name="inputGB" location="10,10" size="500,500" text="input" >
            <Label name="inlabel" location="50,50" size="80,25" text="infile :" />
            <TextBox name="input" location="135,45" size="300,20" text="" />
            <Button name="open" location="250,80" size="45,25" text="open"
link="input"/>
            <Button name="edit" location="300,80" size="45,25" text="edit" />
        </GroupBox>
    </Input>

    <Output>
        <GroupBox name="outputGB" location="10,10" size="500,500" text="output" >
            <Label name="outlabel" location="50,50" size="80,25" text="outfile :" />
            <TextBox name="output" location="135,45" size="300,20" text="" />
            <Button name="select" location="250,80" size="55,25" text="select"
link="output"/>
        </GroupBox>
    </Output>
</UIscript>
```

我们可以看到此 UI 文件以 UIScript 标记为结点，结点中包含两种主要信息。一是树形结点信息，此文件有 Input,Parameters,output 三个树形结点，这样在以后生成的界面中也将出现同名的三个树结点。二是文件给出了用户界面的具体形式，我们以第一个结点 Input 结点所对应的界面来说明。此界面首先有一个 GroupBox 控件，其名字为 inputGB，location 给出了控件在屏幕中所处的位置，size 给出了控件在界面中所覆盖的区域，文本框名为 input，在此界面中还存在一个标记，同样也给出了位置和大小，同时文件中也包含 TextBox 控件，用于进

行参数的输入，位于坐标 135,45 处，大小为 300X20 的长方形区域。最后整个界面还有两按钮，其名称分别为 open 和 edit。以坐标的形式给出了其在界面中的位置。

其它两个结点界面内容和 Input 结点定义相似。

编写好ＵＩ文件后，我们就可以用 UIViewer 对其进行解析。Input 界面解析后显示为图 2-10 所示。

图 2-10　　input 输入界面.

Output 结点界面经解析后如图 2-11 所示。

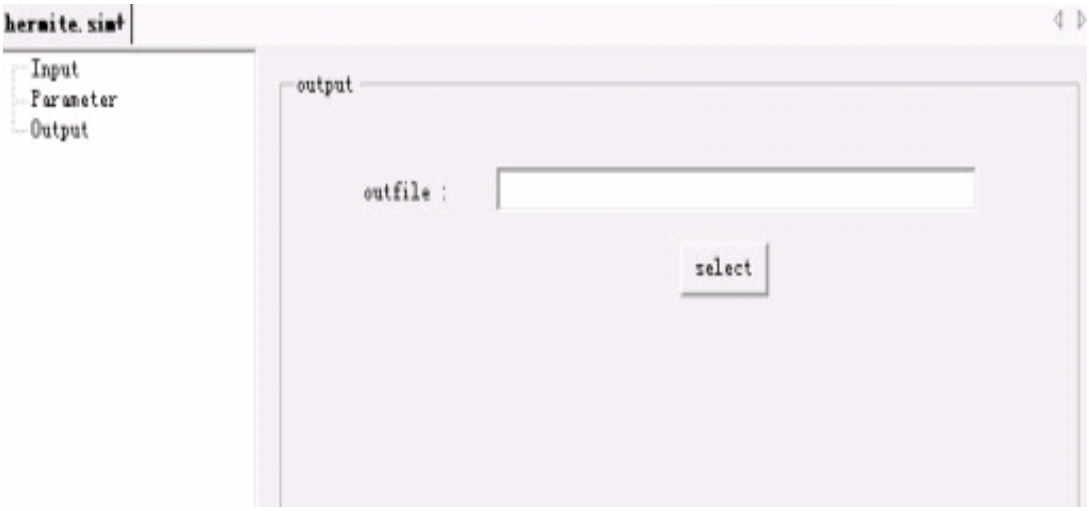

图 2-11　　output 输入界面

Parameter 结点界面解析后如图 2-12 所示。

图 2-3

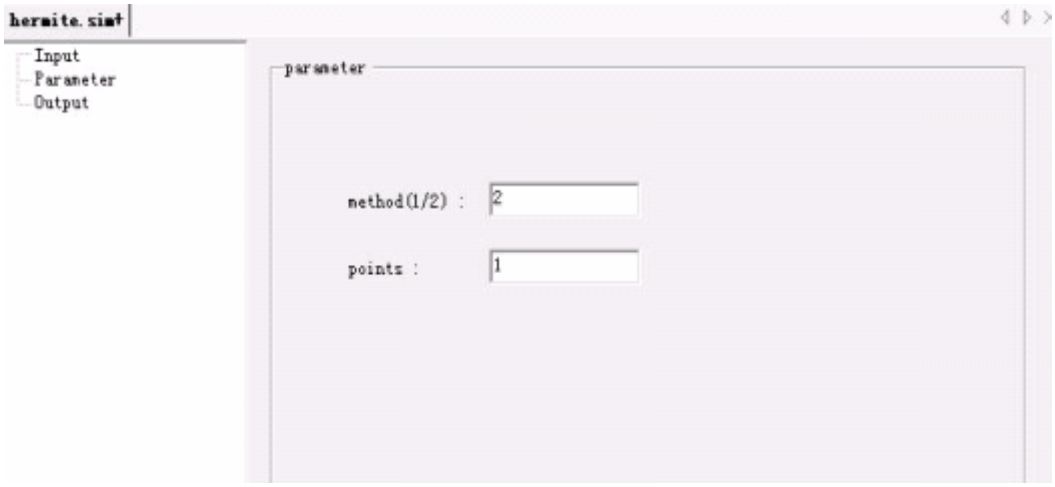

图 2-12　　parameter 输入界面

我们可以看到，通过编写 UI 文件和 SIM 文件，经过 UIViewer 解释器的解析，可以生成满足用户要求的用户界面。而且其编写方法比较简单，同 xml 一样，SIM 文件也是由一系列的标记组成，标记着头文件(MID)和数据，UI 文件中的标记是我们自定义的标记，具有明确的含义，我们可以对标记中的内容的含义作出说明。

从本例中我们可以看到应用 UIScript 开发用户界面是多么的快捷高效。

第三章　　UIViewer 解释器的设计

3.1　UIViewer 的需求

UIViewer 解释器分为 2 个部分，设计和解析。首先，为水利决策支持人员提供开发的平台。为他们提供可设计的界面和工具，设计人员可用这些工具在界面上编辑自己期望的模型。第二，它也可以用来将系统预定义的或者由水利决策支持人员设计编辑生成的 UI 文件解析成为系统所需要的或者设计人员所期望用户看到的形式，展示给用户，通过用户的输入获取相应的数据和参数，保存这些内容到指定的文件中，交予相关程序处理。

关于软件构架用例视图的说明。对于所选择的场景集和（或）作为迭代焦点的用例集，用例视图是很重要的输入。用例视图描述那些代表了某些重要的核心功能的场景集和/或用例集。它还要描述那些在构架方面的涉及范围很广（使用了许多构架元素）的场景集和/或用例集，或者那些强调或阐明了构架的某一具体的细微之处的场景集和/或用例集。

UIViewer 插件的用例包括：

- 编写 UI 文件
- 解析 UI 文件内容
- 生成对应界面
- 保存输入数据
- 选择文件路径
- 打开指定文件
- 运行相应程序

完整的用例图如图 3-1 所示。

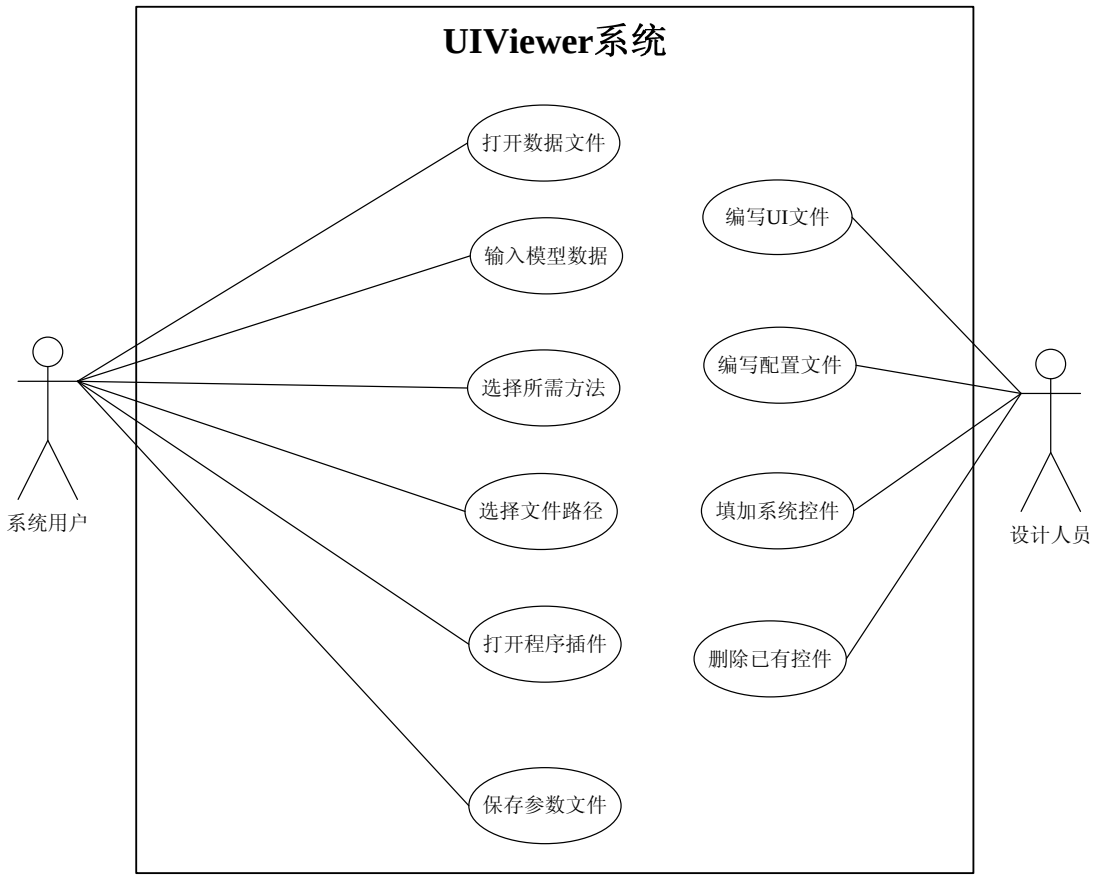

图 3-1　　UIViewer 解释器用例图

3.2　UIViewer 程序运行流程

用户选择要打开的.sim 文件，由外部系统处理，调用 UIScript 插件，UIScript 根据配置解析对应的 UI 文件，为使用者显示界面。可根据系统自定义的或者水利决策人员编写的配制文件，包括所要用到的 TreeView 的名称，相应的值，每个页面中用到的控件的名称和相应值等等，根据这些把他们解析成为用户期望看到的界面。其中左方为 TreeView 区域，负责显示树型结构，使用者可以单击各个节点，还可以展开带有子节点的节点，选中某个节点的时候就会在右方显示出对应的界面。也就是说，右方是用来显示每个节相应的用户界面的区域，在其上会有一些控件，例如 TextBox，Button 等等，这些控件就是用来与使用者交流，获得他们的输入信息。接着，保存输入数据。当使用者点击各个节点的时候，就可以根据标签的提示在控件中输入相应的数据，或者选择一些相关的参数。当用户确定已经输入了正确的内容之后，运行菜单项中的 RunSim，参数将被自动保存到指定的文件中，这个保存着运行参数和数据的文件将会被对应程序调用，计算出结果。

程序流程如图 3-2 所示。

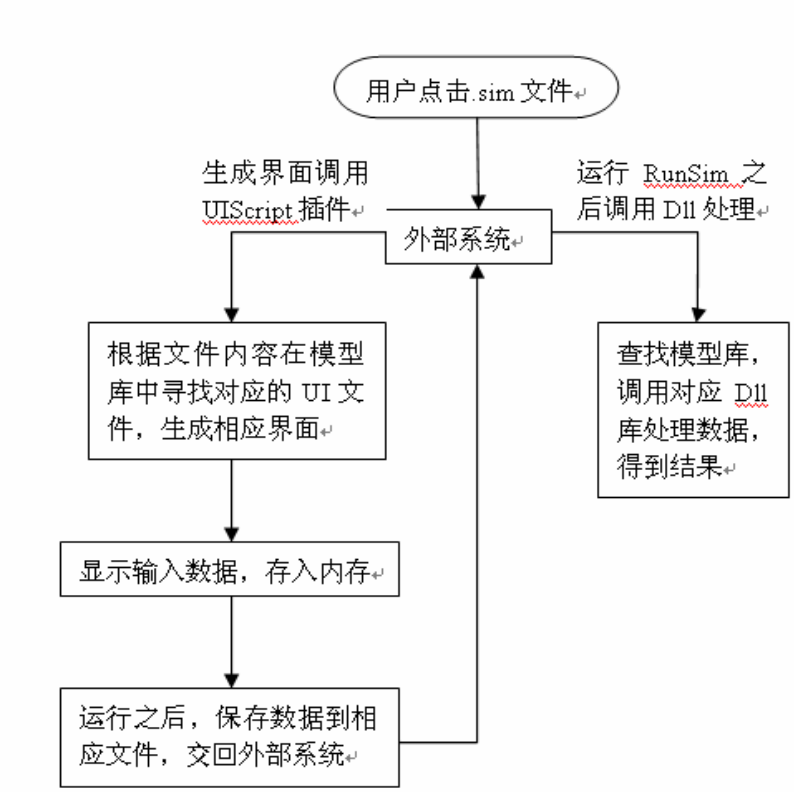

图 3-2　　程序运行流程图

3.3　UIViewer 的模块划分

UIViewer 在整个应用 UIScript 设计程序界面的过程中占有很重要的地位。它主要采取解析 UI 文件的方式，完成对整个系统间各个组成部分的通讯，传递参数等工作。由于其语法简单，结构清晰，所以操作起来相对比较容易。大大方便了系统间模块的交流。另一方面,在参数,命令传递的过程中,UIViewer 解释器可以使整个系统能采用图形化用户界面的方式,改变了以前在系统中用命令行传递的方法,使操作人员的工作效率大大提高,同时减少了其出错的概率,是决策支持系统中一项非常重要的创新.

UIViewer 解释器主要划分为七大模块，包括显示框架模块，树形结点生成模块，显示控件模块，调用外部程序模块，纠错模块，数据存贮模块，查找模块。每个模块的功能如下:

1.显示框架模块负责整个界面的布局，决定了树图界面和控件界面的大小。

2.树形结点生成模块负责用户界面左面的树形区域显示，由根结点和子结点构成。每个结点对应一个专用界面，结点可以有子结点和父结点，也可以展开和折叠等。

3.显示控件模块负责显示对应每个树形结点的界面，可以解析出 UI 文件中的文本输入框、复选框、单选框、选择框的控件。

4.调用外部程序模块主要负责在界面中调出处理模型数据的程序，用于编辑处理输入模型数据。

5.纠错模块负责体验输入数据是否符合模型所需数据的格式，如不符合弹出提示对话框，提示用户重新检查并重新输入数据。

6.数据存贮模块可以将模型处理后的数据保存到指定文件，用于系统中其它模型的输入。

7.查找模块可以通过 MID 查找指定的 UI 文件。

系统中各模块间的关系如图 3-3 所示。

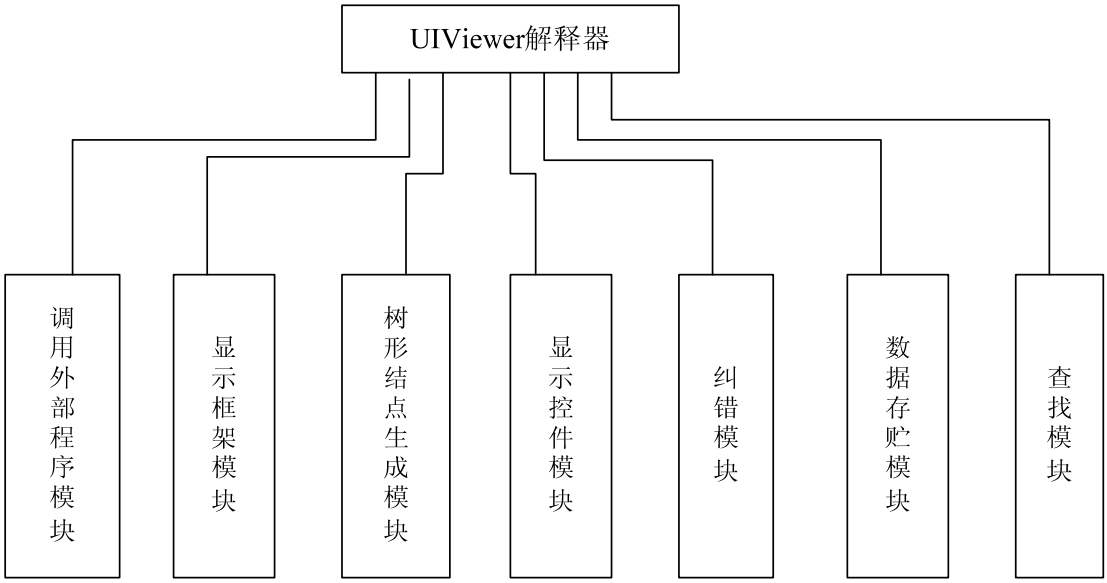

图 3-3　系统中模块划分

3.4　UIViewer 的界面设计

首先，选择需要打开的文件，针对本模块而言，可以打开的文件的后缀名为“.sim”是 Simulation 文件，在窗体右方的资源管理器中找到需要的“.sim”文件，双击打开它，就会显示出对应的界面。

左方的 TreeView 的节点为“Input”，“Parameter”，“Output”。它们分别对应着各自的界面。右方显示的是每个节点所对应的界面。

“Input”为一个文本框，两个按钮名字为“open”、“edit”。“open”按钮调出文件打开对话框，可以选择相应的文件，选择的文件的路径将会显示在文本框中，这时候文本框中的内容是不能被修改的。另一个按钮“edit”，是在用“open”选中需要的文件的路径之后应用合适的程序打开这个文件，对它进行编辑，保存编辑的结果。文本框中的内容最终会作为一个参数写到指定的文件中。这样就得到了要求的输入文件的路径。

“Parameter”页面包括两个标签和两个文本框，标签是用来使用户区分这两个文本框中要输入的内容的。 “method”标签提示用户在右边的文本框中输入所选

择的方法的代号，默认为 2；在“points”标签右边的文本框中要求输入的数据是在两点之间所要插入点的个数，默认为 1。这样就得到运行的参数。

“Output”页面包括一个文本框和一个按钮，基本上，功能与“Input”的相同，只是没有了“edit”的功能，所选择的是输出的文件的路径显示在文本框中。

当用户选择各个节点，输入相应的参数之后，在菜单项中找到“窗口”菜单，点击其下的“RunSim”，这时系统才会保存所输入的参数等信息。

界面设计的理念是尽量满足用户的需要，符合用户的使用习惯，方便设计人员和使用人员。在界面初始生成的时候，没有任何点击 Treenode 之前，即初始情况下，显示出的是第一个 Treenode 对应的界面。这样的设计会更加符合用户的习惯，方便熟悉使用本系统。用户可以在生成每个界面的时候，也就是在生成 XML 文件的时候，为每个控件加上默认的值，在显示的工程中用户设定的默认也会自动的显示在对应的控件中。Treenode 是被用户手动切换的，在每次切换之前都会保存用户在这个窗口中没个控件的内容，虽然用户暂时切换了 Treenode，但其实前面输入的内容已经保存在内存中了，并且当用户切换回前面打开过的 Treenode 时，曾经输入的内容还会显示回来，只有选择“RunSim”之后内存中的内容才会真正写入文件中，这样做既保证了运行的速度，又满足了需要。

3.5 UIViewer 的测试报告

用户点击.sim 文件之后，生成初始界面。测试用例：点击任何 treenode 都会显示正确的参数界面。点击 Input 的界面，点击 Parameter 的界面，点击 Output 的界面。

在 Input 界面中，会有输入框。输入界面如图 3-4 所示。

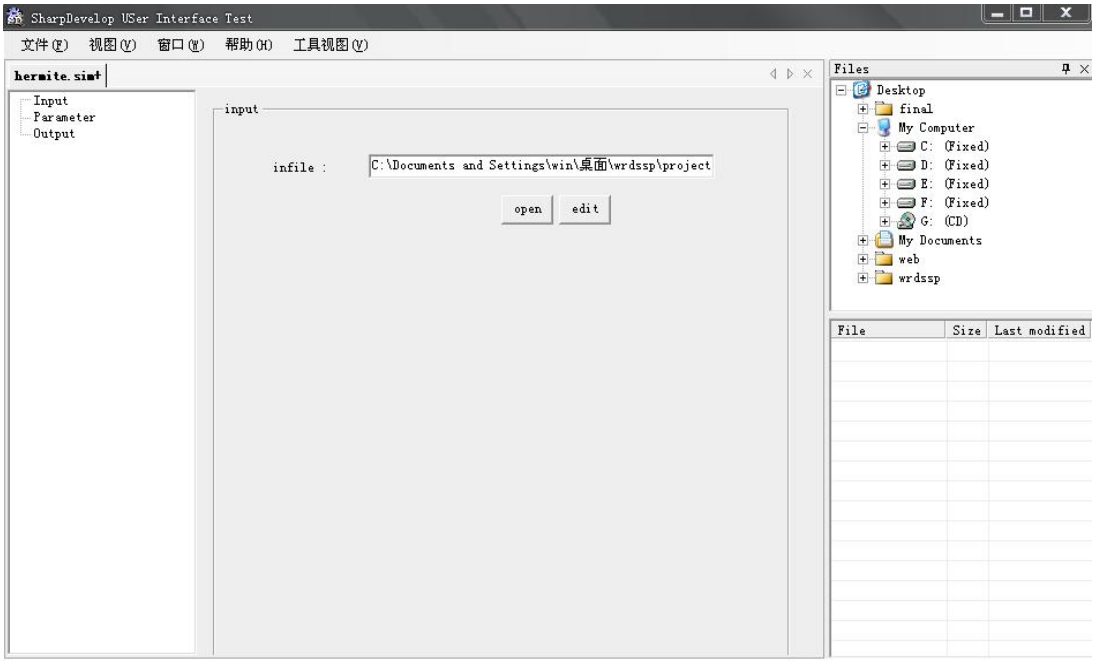

图 3-4　　Input 节点对应界面

OPEN 按钮打开界面如图 3-5 所示。

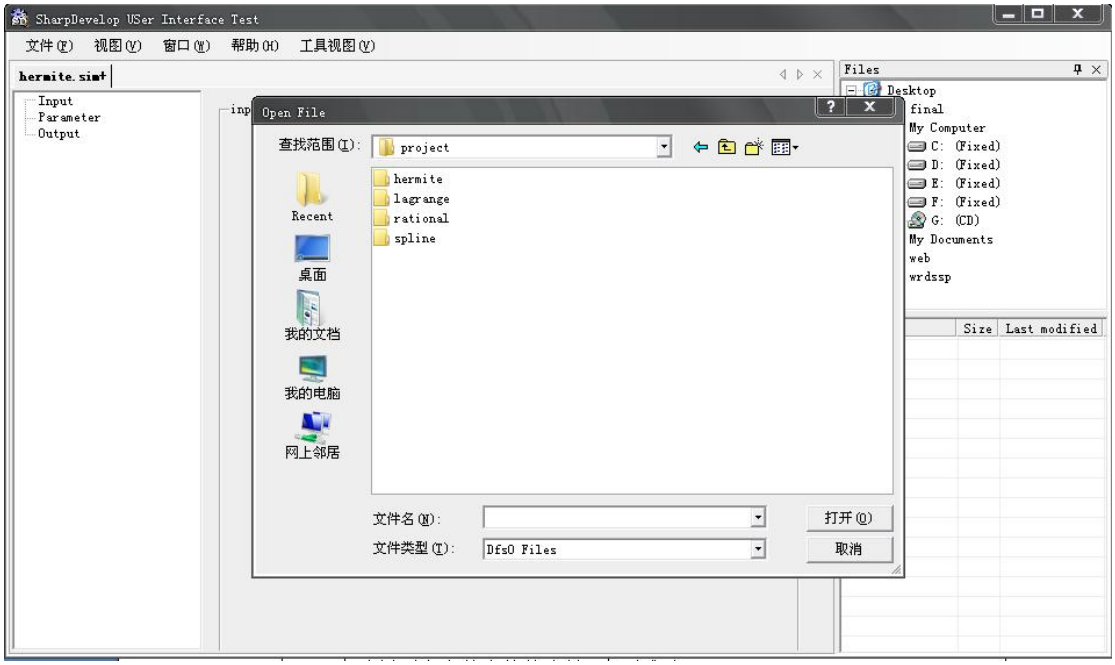

图 3-5　　OPEN 对应打开界面

第四章　　UIViewer 解释器的实现

4.1　对 UI 文件的解析

在实现系统的第一步，我们利用 C＃读取 UI(XML)文件，并用 ListView 组件按照数据结构给显示出来。要正确读取 XML 数据，首先要了解 XML 的结构[6]。根据系统的 XML 文件格式，可以知道此 XML 包含三个数据，下面就来解释如何读取这三条数据：

（1）装入 XML 文件，形成数据流： 通过创建一个"XmlDocument"对象，然后利用"load"方法，可以把 XML 文件装入，具体如下：

```
XmlDocument doc = new XmlDocument ( ) ;// 装入指定的 XML 文档
doc.Load ( "C:\\data.xml" ) ;
```

（2）读取 XML 文件，并显示出来：

要读取 XML 是通过创建"XmlNodeReader"对象来实现的，"XmlNodeReader"对象主要是用来可以读取 XML 的节点数据。在本文的程序中使用到"XmlNodeReader"的属性，譬如"NodeType"属性，用来判断读取得节点是什么类型。"Value"是节点的值。下面是读取 XML 文件并显示在 ListView 中的实现代码，其中 listview1 是已经创建 ListView 组件. 程序代码如图 4－1 所示。

```
while ( reader.Read ( ) )
{//判断当前读取得节点类型
switch ( reader.NodeType )
{
case XmlNodeType.Element :
s = reader.Name ;
break ;
case XmlNodeType.Text :
if ( s.Equals ( "Name" ) )
myItem = Listview1.Items.Add ( reader.Value ) ;
else
myItem.SubItems.Add ( reader.Value ) ;
break ;
}
}
```

图 4-1　　显示 XML 文件代码

在分析了解了上面的内容以后，可以得到用 C＃读取指定 UI(XML)文件的完

整代码.程序中读取 XML 文件可以有三种方式，如下。

1、XmlDocument 和 XPathNavigator —— 它们提供数据的内存视图和编辑工具，检查是否符合 XML 的要求，最重要的是它们提供了数据修改的方法；

2、XmlReader——数据的只读顺序访问；

3、PathDocument 和 XPathNavigator —— 在内存中对 XML 数据进行只读的、优化的随机访问。

在功能上，XmlReader 类和 XPathDocument 类有一些相似之处[7]，不过，XPathDocument 仍然必须将 XML 文档解析到内存中，因此对大型文档而言，它比 XmlReader 的速度要慢。

本文采用的是应用的是 XmlDocument,和 XPathNavigator 的访问模式。

4.2 TreeView 控件

4.2.1 TreeView 控件的定义

在实现 UIViewer 解释器中关键的步骤同时也是系统中最具特色的地方是如何使用只进、非缓存的 XmlTextReader 和 XmlTextWriter 类，把 XML 文件内容载入到 TreeView 控件中，以及如何把 TreeView 控件中的内容保存到 XML 文件中。

4.2.2 加载数据

本节的目的是说明演示如何把 XML 文件内容载入到 System.Windows.Forms.TreeView 控件中，以及如何把 TreeView 控件中的内容保存到 XML 文件中。 System.Xml 命名空间中 XmlTextReader 和 XmlTextWriter 类分别用于读取和生成 XML 文件[8]。

实现函数包含在一个名为 TreeViewSerializer 的类中。它有两项主要的任务：
1.保存 TreeView 中的内容到指定的 XML 文件中
2.从指定的 XML 文件中读入内容，填充到 TreeView 中,用于载入到 TreeView 中的 XML 文件的结构十分简单，下面是一个包含在所附演示项目中的 XML 示例文件的内容：
使用 XmlTextWriter 与 XmlTextReader 读取和保存 TreeView 控件到 XML 文件。
Code: <?xml version="1.0" encoding="utf-8"?>

```
<UIscript>
    <T value="123">
        <Tree name="Input" text="Input" path="Input">
        </Tree>
        <Tree name="Parameters" text="Parameter" path="Parameters">
        </Tree>
        <Tree name="Output" text="Output" path="Output">
```

```
        </Tree>
    </T>
    <Input>
        <GroupBox name="input" location="10,10" size="500,500" text="input" >
            <Label name="inlabel" location="50,50" size="80,25" text="infile :" />
            <TextBox name="infile" location="135,45" size="300,20" text="" />
            <Button name="open" location="250,80" size="45,25" text="open"
link="infile"/>
            <Button name="edit" location="300,80" size="45,25" text="edit" />
        </GroupBox>
    </Input>
    <Parameters>
        <GroupBox name="parameter" location="10,10" size="500,500"
text="parameter" >
            <Label name="para1" location="50,80" size="90,25"
text="method(1/2) :" />
            <Label name="para2" location="50,120" size="90,25" text="points :" />
            <TextBox name="method" location="145,75" size="100,20" text="2" />
            <TextBox name="points" location="145,115" size="100,20" text="1" />
        </GroupBox>
    </Parameters>
    <Output>
        <GroupBox name="output" location="10,10" size="500,500" text="output"
>
            <Label name="outlabel" location="50,50" size="80,25" text="outfile :"
/>
            <TextBox name="outfile" location="135,45" size="300,20" text="" />
            <Button name="select" location="250,80" size="55,25" text="select"
link="outfile"/>
        </GroupBox>
    </Output>
     </UIscript>
```

在声明 XML 后，所有的节点都被附加到 TreeView 的 标记上。TreeView 的标记项可以包含多个节点标记。节点标记也可以包含其他的节点标记。每个节点标记都包含下列三个属性：

1.Text

2.ImageIndex

3.Tag

我把 XML 载入到了 System.Windows.Forms.TreeNode 对象的上述三个属性中。你可以轻松地进行扩展，以包含其他属性。

XmlNodeTag, XmlNodeTextAtt, XmlNodeTagAtt 和 XmlNodeImageIndexAtt 是定义在 TreeViewSerializer 类中的常量[9]：

从 XML 文件载入内容到 TreeView 中载入任务由 DeserializeTreeView 方法实现，该方法使用了 XmlTextReader 类来解析 XML 文档，并填充到 TreeView 对象中。下面代码是 DeserializeTreeView 方法的定义。

```
public void DeserializeTreeView(TreeView treeView, string fileName)
{
XmlTextReader reader = null;
try
{
// 禁止 treeview 重绘，直至节点被添加完毕
treeView.BeginUpdate(); reader = new XmlTextReader(fileName);
TreeNode parentNode = null;
while (reader.Read())
{
if (reader.NodeType == XmlNodeType.Element)
{
if (reader.Name == XmlNodeTag)
{
TreeNode newNode = new TreeNode();
bool isEmptyElement = reader.IsEmptyElement;
// 读入节点属性
int attributeCount = reader.AttributeCount;
if (attributeCount > 0)
{
for (int i = 0; i < attributeCount; i++)
{
reader.MoveToAttribute(i);
SetAttributeValue(newNode,
reader.Name, reader.Value);
}
}
// 添加新的节点到父节点(Parent Node)或 TreeView
if(parentNode != null)
parentNode.Nodes.Add(newNode);
else


public void DeserializeTreeView(TreeView treeView, string fileName)
{
XmlTextReader reader = null;
try
{
// 禁止 treeview 重绘，直至节点被添加完毕
treeView.BeginUpdate(); reader = new XmlTextReader(fileName);
TreeNode parentNode = null;
```

```
while (reader.Read())
{
if (reader.NodeType == XmlNodeType.Element)
{
if (reader.Name == XmlNodeTag)
{
TreeNode newNode = new TreeNode();
bool isEmptyElement = reader.IsEmptyElement;

// 读入节点属性
int attributeCount = reader.AttributeCount;
if (attributeCount > 0)
{
for (int i = 0; i < attributeCount; i++)
{
reader.MoveToAttribute(i);
SetAttributeValue(newNode,
reader.Name, reader.Value);
}
}
// 添加新的节点到父节点(Parent Node)或 TreeView
if(parentNode != null)
parentNode.Nodes.Add(newNode);
else

public void DeserializeTreeView(TreeView treeView, string fileName)
{
XmlTextReader reader = null;
try
{
// 禁止 treeview 重绘，直至节点被添加完毕
treeView.BeginUpdate(); reader = new XmlTextReader(fileName);
TreeNode parentNode = null;
while (reader.Read())
{
if (reader.NodeType == XmlNodeType.Element)
{
if (reader.Name == XmlNodeTag)
{
TreeNode newNode = new TreeNode();
bool isEmptyElement = reader.IsEmptyElement;

// 读入节点属性
int attributeCount = reader.AttributeCount;
```

```
if (attributeCount > 0)
{
for (int i = 0; i < attributeCount; i++)
{
reader.MoveToAttribute(i);
SetAttributeValue(newNode,
reader.Name, reader.Value);
}
}
// 添加新的节点到父节点(Parent Node)或 TreeView
if(parentNode != null)
parentNode.Nodes.Add(newNode);
else
treeView.Nodes.Add(newNode);
// 如果不为空，则设置当前节点为父节点(ParentNode)
if (!isEmptyElement)
{
parentNode = newNode;
}
}
}
// 如果遇到结束标记，提升父节点
else if (reader.NodeType == XmlNodeType.EndElement)
{
if (reader.Name == XmlNodeTag)
{
parentNode = parentNode.Parent;
}
}
else if (reader.NodeType == XmlNodeType.XmlDeclaration)
{
//忽略 Xml 声明部分
}
else if (reader.NodeType == XmlNodeType.None)
{
return;
}
else if (reader.NodeType == XmlNodeType.Text)
{
parentNode.Nodes.Add(reader.Value);
}
}
}
finally
```

```
{
// 在所有节点被添加后，允许重绘 treeview
treeView.EndUpdate(); reader.Close(); }
}
```

4.2.3 生成用户界面树形结点

当 UIViewer 解析 UI 文件时，适当的动作依赖于节点类型(NodeType)做出。如果节点类型(NodeType)为(元素)Element, 则建立一个新的 TreeNode ，并且使用 XML 的节点属性设置它的属性。在非空元素的情况下，父节点(ParentNode)设置为新的 TreeNode[10]， 以便于它的子节点能被载入。如果遇到一个末尾元素标记(EndElement)，父节点(ParentNode)设置为当前父节点的父节点，以标识当前节点的所有子节点都已经被载入。

为了设置 TreeNode 的 Text,Tag 和 ImageIndex 属性，调用 SetAttributeValue 方法，定义如图 4-2 所示。

```
///被 Deserialize 方法调用，以 XML 的节点属性，设置 TreeNode 的属性
private void SetAttributeValue(TreeNode node,
string propertyName, string value)
{
if (propertyName == XmlNodeTextAtt)
{
node.Text = value;
}
else if (propertyName == XmlNodeImageIndexAtt)
{
node.ImageIndex = int.Parse(value);
}
else if (propertyName == XmlNodeTagAtt)
{
node.Tag = value;
}
}
```

下面是此方法的定义：

```
    public void SerializeTreeView(TreeView treeView, string fileName)
{
XmlTextWriter textWriter = new XmlTextWriter(fileName,
System.Text.Encoding.ASCII);
// 写入 XML 声明标记
textWriter.WriteStartDocument();
//textWriter.WriteRaw("\r\n");
// 写入包含所有节点标记的主标记
textWriter.WriteStartElement("TreeView");
```

```
// 保存节点，递归方法
SaveNodes(treeView.Nodes, textWriter);
textWriter.WriteEndElement();
textWriter.Close();
}
```

图 4-2　　设置 TreeNode 属性代码

SerializeTreeView 方法：

1.创建 XmlTextWriter 类的实例，传递提供的文件名参数.

2.XML 声明 (<?xml version="1.0" encoding="us-ascii" ?>) 被写入到流中.

3.TreeView 标记被写入流中.

4.SaveNodes 方法被调用。把 TreeView 控件中的 TreeNode 集合传递给此方法，此方法递归调用自身，以保存 TreeView 中的所有节点到流中。此方法的定义在下面。

5.TreeView 的结束标记被写入.

6.最后关闭流.

SaveNodes 方法的定义如图 4-3 所示。

```
private void SaveNodes(TreeNodeCollection nodesCollection, XmlTextWriter
textWriter)
{
for(int i = 0; i < nodesCollection.Count; i++)
{
TreeNode node = nodesCollection[i];
textWriter.WriteStartElement(XmlNodeTag);
textWriter.WriteAttributeString(XmlNodeTextAtt, node.Text);
textWriter.WriteAttributeString(XmlNodeImageIndexAtt,node.ImageIndex.ToString()
);
if(node.Tag != null)
textWriter.WriteAttributeString(XmlNodeTagAtt, node.Tag.ToString());
// 此处添加其他节点属性到序列
if (node.Nodes.Count > 0)
{
SaveNodes(node.Nodes, textWriter);
}
textWriter.WriteEndElement();
}
}
```

图 4-3　　Savenode 方法代码

SaveNodes 方法通过 TreeNode 集合进行循环，并且写入 节点标记(node tag) 和它的属性。如果一个 节点 有子节点，此方法将被递归调用。在方法返回后，已经写入子节点到流中，节点结束标记也被写入。

4.2.4 TreeView 组件应用

TreeView 组件是由多个类来定义的，TreeView 组件[11]是由命名空间"System.Windows.Forms"中的"TreeView"类来定义的，而其中的节点(即 Node)，是由命名空间"System.Windows.Forms"中的"TreeNode"来定义的。所以当在程序中创建一个 TreeView 对象，其实只是创建了一个可以放置节点的"容器"。而在这个容器中加入一个节点，其实就是加入了从"TreeNode"类中创建的一个节点对象；同样删除一个节点，也就是删除一个"TreeNode"节点对象。

在 UIScript 中实现树图的功能,必须应用 TreeView 组件,其有三种基本操作：加入子节点、加入兄弟节点和删除节点。下面说明其在本系统中的功能实现。

（1）加入子节点：

所谓子节点，就是处于选定节点的下一级节点。加入子节点的具体过程是：首先要在 TreeView 组件中定位要加入的子节点的位置，然后创建一个节点对象，然后利用 TreeVeiw 类中对节点的加入方法（即：Add ( )方法），加入此节点对象。下面就是在 treeView1 组件中加入一个子节点的具体代码：

```
  //首先判断是否选定组件中的位置
if ( treeView1.SelectedNode == null )
{
MessageBox.Show ( "请选择一个节点" , "提示信息" , MessageBoxButtons.OK , MessageBoxIcon.Information ) ;
}
else
{
//创建一个节点对象，并初始化
TreeNode tmp ;
tmp = new TreeNode ( "节点名称" ) ;
//在 TreeView 组件中加入子节点
treeView1.SelectedNode.Nodes.Add ( tmp ) ;
treeView1.SelectedNode = tmp ;
treeView1.ExpandAll ( ) ;
}
```

（2）加入兄弟节点：

所谓兄弟节点，就是在选定的节点的平级的节点。加入兄弟节点的方法和加

入子节点的方法基本一致，只是在最后的实现方法上有着略微的区别。加入兄弟节点的具体步骤，首先也是要确定要加入的兄弟节点所处的位置，接着定义一个节点对象，最后调用 TreeView 类中对兄弟节点加入的方法，加入此节点对象。加入兄弟节点和加入子节点的最大区别就在于这最后一步。希望读者能够注意。下面是在 TreeView 组件加入一个兄弟节点的具体代码：

```
  //首先判断是否选定组件中节点的位置
if ( treeView1.SelectedNode == null )
{
MessageBox.Show ( "请选择一个节点" , "提示信息" , MessageBoxButtons.OK ,
MessageBoxIcon.Information ) ;
}
else
{
//创建一个节点对象，并初始化
TreeNode tmp ;
tmp = new TreeNode ( textBox1.Text ) ;
//在 TreeView 组件中加入兄弟节点
treeView1.SelectedNode.Parent.Nodes.Add ( tmp ) ;
treeView1.ExpandAll ( ) ;
}
```

（3）删除节点：

删除节点就是删除 TreeView 组件中选定的节点，删除节点可以是子节点，也可以是兄弟节点，但无论节点的性质如何，必须保证要删除的节点没有下一级节点，否则必须先删除此节点中的所有下一级节点，然后再删除此节点。删除节点比起上面的二个操作要显得略微简单，具体方法是：首先判断要删除的节点是否存在下一级节点，如果不存在，就调用 TreeView 类中的 Remove ( )方法，就可以删除节点了。下面是删除 TreeView 组件中节点的具体代码：

```
  //判断选定的节点是否存在下一级节点
if ( treeView1.SelectedNode.Nodes.Count == 0 )
//删除节点
treeView1.SelectedNode.Remove ( ) ;
else
MessageBox.Show ( "请先删除此节点中的子节点！" , "提示信息" ,
MessageBoxButtons.OK , MessageBoxIcon.Information ) ;
```

（4）展开、折叠结点等操作

在系统中，我们也可以展开所有节点，展开指定的节点、和折叠所有节点。下面就来具体介绍一下：

1　展开所有节点：要展开 TreeView 组件中的所有节点，首先就要把选定的节点指针定位在 TreeView 组件的根节点上，然后调用选定组件的 ExpandAll 方法就可以了，下面是具体代码：

```
//定位根节点
treeView1.SelectedNode = treeView1.Nodes [ 0 ] ;
//展开组件中的所有节点
treeView1.SelectedNode.ExpandAll ( ) ;
```

2　展开选定节点的下一级节点：由于只是展开下一级节点，所以就没有必要用 ExpandAll ( )方法了。展开下一级节点只需要调用 Expand ( )方法就可以了，下面是具体的实现类。

```
treeView1.SelectedNode.Expand ( ) ;
```

3　折叠所有节点：折叠所有节点和展开所有节点是一组互操作，具体实现的思路也大致相同，折叠所有节点也是首先要把选定的节点指针定位在根节点上，然后调用选定组件的 Collapse ( )就可以了，下面是具体的实现代码：

```
//定位根节点
treeView1.SelectedNode = treeView1.Nodes [ 0 ] ;
//折叠组件中所有节点
treeView1.SelectedNode.Collapse ( ) ;
```

4.3　UIViewer 解释器的插件化

一、在水利泥沙决策支持系统中,我们所做的 UIViewer 解释器是做为该系统的一个插件而应用的,所以,必须将其做成 DLL 文件才能为整个决策支持系统所用，另一方面 DLL[12]的产生使得我们的应用程序在可维护性、代码的重复使用等方面都有了很大的提高。所以本节将就这方面的问题来进行简单的介绍。

Visual C++、VB 等编程语言来编写成的 DLL 文件，在编译完成过以后，产生 DLL 文件已经是一个可以直接供计算机使用的二进制文件。但用 Visual C#编译器生成的受管代码（managed code）虽然也是二进制文件，但不是可以直接供计算机使用的原始代码（机器语言代码）。他实质上是一种中间语言（IL）代码，这种 IL 代码要转变成可以供计算机直接使用的原始代码，就需要经过"下一代窗口服务"(Next Generation Windows Services,简写为 NGWS) runtime 的即时编译器（即 JIT）进行编译。

经过以上比较，我们可以看出，用 Visual C#生成的 DLL 文件已经和以前的 DLL 文件有了本质上的区别。用 Visual C#生成的 DLL 文件在程序设计中更多的

表现为一种类（Class）或者类库（Class Library）。本文就试着通过一个具体程序的例子，按照下面步骤来具体介绍：

（1）.创建一个 DLL 源代码。

（2）.编译此 DLL 源代码，生成 DLL 文件。

（3）.用此 DLL 来创建一个简单的客户端程序。

二、下面创建一个 DLL 源代码（UIViewer.cs）

由于用 Visual C#创建的 DLL，此 DLL 是不需要执行的界面，所以在 DLL 文件就没有必要定义 Main ( )函数[13]，来作为应用程序执行的入口。Dll.cs 的源程序代码如图 4-4 所示。

```
Dll.cs：
namespace Dll file://定义了名称空间，在调用 DLL 的时要导入此名称空间。
    public class Show file://定义了一个类，在程序中就要来继承此类。
  {
    public string Messages ( )
    file://定义了一个方法，此方法的作用就是返回下面字符串。
    {
      return "使用 C#做的 DLL 文件！";
    }
 }
```

图 4-4　　做成ＤＬＬ文件的代码框架

通过此 DLL 的源程序可以看出，此 DLL 表现为一个小型的类库，这是因为在此 DLL 中封装了名字叫 DLL 的名称空间，在此名称空间中又定义了一个 Show 类，在此类中有一个方法就是 Messages。虽然定义的内容相对少了些，但却相当完全。

三、编译此 DLL 源代码，生成 DLL 文件

要把 DLL 源代码编译成 DLL 文件，就需要配置好编译器 Csc.exe 的参数和开关。我们知道编译器 Csc.exe 可以把源代码编译成四种不同的文件，分别是控制应用程序、代码库、windows 应用程序、模块程序。编译命令具体如下[14]：

```
csc /target:exe UIViewer.cs //  创建一个 UIViewer.exe 控制程序
csc /target:winexe UIViewer t.cs file://创建 UIViewer.exe 的 windows 程序 c
csc /target:library myProject.cs file://创建一个 UIViewer.dll 代码库
csc /target:module UIViewer.cs file://创建一个 UIViewer.dll 模块
```

还有最后一步,输入以下命令,即完成任务.

```
Csc /r:system.dll /t:library /out: UIViewer.dll dll.cs
```

至此,我们就完成了将 UIViewer 系统程序转换成 UIViewer.dll 文件,在水利决策支持系统中可以直接调用了.需要在此说明的是用 Visual C#或者其他.Net 程序开发语言制作的 DLL 和以前的 DLL 有了实质上的区别。它已经不是严格意义上的动态连接库了，而是一个类或者类库[15]。

第五章　　结论

5.1　总结

论文主要定义了一种界面开发语言——UIScript，同时给出了应用UIScript开发用户界面时所需编写的两种格式的文件——SIM文件和UI文件。详细定义了UI文件的编写方法及遵循的语法规则，并明确了其是用于界面开发生成用户界面的文件。同时也详细的介绍了SIM文件的编写规范和用途。UI文件保存的是用户界面显示信息和参数输入传递方式，及整个界面的布局，而SIM文件保存的是系统所需输入的参数及经过系统处理后输出的参数。通过定义这两种不同类型的文件，我们可以将程序所需数据和程序的界面分隔开来，方便操作人员对整个系统的使用和管理。论文给出了在水利泥沙防洪系统中应用UIScript开发用户界面的示例。生动的说明了应用UIScript开发用户界面所带来的种种好处。

论文同时设计实现了UI文件的解释器——UIViewer，其是负责将UI文件解析成用户期望看到的图形化界面，用户可以在此界面上进行参数的输入输出，保存更改，系统命令的传递。其功能相当于一个编译器，将系统定义的UIScritp语言编写的UI文件解释执行，生成用户界面，完成系统和操作人员的沟通交流。

通过完成以上两方面的工作，我们就可以用 UIScript 开发软件系统的用户界面。在水科院开发的水利决策支持系统的用户界面开发中，我们用 UIScript 代替了传统的界面开发手段。实践表明，其出色的完成了系统间各个模块的通讯，传递参数、命令等工作。而且在开发界面的过程中由于其语法简单，结构清晰，所以开发起来相对比较容易. 另一方面，采用了图形化用户界面的方式，改变了以前在系统中用命令行传递的方法，使操作人员的工作效率大大提高，同时减少了其出错的概率。

5.2　展望

UIScript在用户界面开发中的研究和应用很多都还处于不成熟的探索阶段。在探索的过程中，越来越多的问题暴露出来：

1. UIScript设计生成界面图形化程度不高。目前的很多本体仍处于简单的图形化阶段，只是为领域提供一种参数的传递,存贮,更改等简单操作。这无法满足计算机系统的复杂需求。

2. UIscript开发设计工具的缺乏。领域的变化、应用背景的变化也都可能引起一种开发语言的的重用甚至重建。另外，缺乏一套完整的系统评价和维护方法，也就是缺少开发UISscript的工具，这种工具应该不但可以负责UIViewer解释器程序的开发，更可以进行有效的插件管理、评价和维护。最新的C#语言虽然提供了一定的对UIViewer开发和维护的支持，但仍然很不完善，需要更多的研究者来关注这一新的角本界面语言的研究方向。

虽然UIScript在用户界面开发的应用中有种种以上的不利因素，但我们不能忽略用其开发的用户界面使系统各模块之间的通讯更加迅速。同时同传统的界面开发语言相比，大大降低了用户界面开发的难度，使用户界面开发所需的时间也大大缩短。尽管其处于不成熟的阶段,但通过我们的不懈努力，一定会克服前进中的种种困难，道路是曲折的。前途是光明的，UIScript一定会有一个光明的应用前景。

参考文献

[1] 布托．用户界面设计指南[M]．机械工业出版社，2008．34－57．

[2] 哥瑞菲斯．HTML之路[D]．机械工业出版社，2008．22－24．

[3] Awad,Ala.Achieving high performance on Web-based J2EE application servers[D]. Canada：Carleton University，2007.

[4] Albert Elcock,Phillip Laplante.Testing software without requirements: using development artifacts to develop test cases[J]．Innovations in Systems and Software Engineering，2006，2(3)：137-145.

[5] 斯梅切尔．C#和.NET 2.0 实战、平台、语言与框架[M]．人民邮电出版社，2008．112－197．

[6] 顾进广，陈莘萌．基于语义的XML信息集成技术[M]．武汉大学出版社，2007．155－197.

[7] 吴洁．XML应用教程（第2版）[M]．清华大学出版社，2007．55－107．

[8] 左伟明．即用即查XML数据标记语言参考手册[M]．人民邮电出版社，2007．215－230．

[9] 依夫杰．C# 2005 &.NET 3.0编程[M]．清华大学出版社，2004．33－54．

[10] 雅可布斯．XML基础教程[M]．人民邮电出版社，2007．34－36．

[11] 内格尔．C＃高级编程(第4版)[M]．清华大学出版社，2006．44－48．

[12] 罗斌．Visual C#2005编程技巧大全[M]．水利水电出版社，2007．65－98．

[13] 马丁．敏捷软件开发（C#版）[D]．人民邮电出版社，2008．85－88．

[14] 夏普．Visual C# 2005从入门到精通[M]．清华大学出版社，2006．23－58．

[15] 特罗尔森．C#与.NET 3.0 高级程序设计（特别版）[M]．人民邮电出版社，2008．34－36．

致　　谢

值此论文完稿之际，谨向所有帮助过我的老师，同学和亲友表示我最诚挚的谢意。

首先我要衷心的感谢我的导师张坤龙。本文的选题，研究工作以及论文的写作都是在张老师的悉心指导和帮助下才得已完成的。论文的顺利完成，得益于张老师对我的严格训练和谆谆教诲。张老师丰富的专业知识，独到的见解，严谨的治学态度，创新的学术思想，追求完美和一丝不苟的敬业精神深深的激起了我对知识的渴求，不但对我的学业帮助巨大，更将使我终身受益。其次特别感谢我的师弟邹岑同学，在论文写作期间给了我很大的帮助。给我提了很多建设性的意见，使得论文得已最终完成。

四年的大学生活，学院的老师，同学无论在生活还是学习上都给予了我巨大的帮助。这里特别要感谢学院的贾津老师，团委的潘峰老师对我的无私帮助让我能够克服万难完成论文，在这里我真心的感谢并祝福他们。

最后，我要把一份特别的感谢献给我的家人，为了我的学业，他们做出了巨大牺牲，他们无微不至的关怀和期望是我一直不断向前的动力之源。对于这些，我真是觉得无以回报，只有在将来的学习和工作中加倍努力，不辜负家人对我的期望。