图形用户界面生成工具的设计与实现

学　　院　软件学院

专　　业　软件工程

年　　级　**2006**

姓　　名　陈春光

指导教师　张坤龙

2010 年 **6** 月 **15** 日

摘　　要

伴随着计算机技术的发展和普及，越来越多的非计算机专业出身的人员投入到专用软件的设计开发的工作中。由于并不具备计算机专业出身的专业知识，这些人需要一种简单的方式来进行界面的设计和实现，图形用户界面生成工具完成的就是这样的一个功能。

论文中所设计和实现的图形用户界面生成工具是由显示框架模块、工具栏模块、XML 转换模块、生成界面模块、数据存储模块和显示控件模块六个模块构成。

显示框架模块负责整个界面的布局，确定图形界面和控件界面的大小等等。

工具栏模块用于展示所有的工具选项，给用户展示工具控件，完成在用户完成拖拽操作后自动生成的代码。

XML 转换模块用于在后台将需要转换成 XML 代码的界面进行转换的过程中。

生成界面模块是在用户进行拖拽过程后生成界面的必需模块。

数据存储模块可以将模型处理后的数据保存到指定的文件，用于接下来的程序调用。

显示控件模块显示节点的界面，解析出工具栏中的控件。

关键词：用户界面；人机交互；脚本；xml；Windows Forms 编程

ABSTRACT

Following the development and the popularization of computer technology,there are more and more non-computer specialized people who invest into the special-purpose software's design development work. As they don't have enough knowledge about computer technology,they need a simple tool to design and realize the surface.The graphical user interface production tool has such functions.

The graphical user interface production tool mentioned in the article contain six modules. They are the demonstration frame module, the toolbar module, the XML transformation module, the production contact surface module, the data storage module and the demonstration controls module.

The demonstration frame module is responsible for the layout of the entire contact surface , the definition of graphical interface and the size of controls contact surface and so on.

The toolbar module is used in demonstrating all tool options, demonstrated the labor files for the user , completes the automatic production code after the user completes tows entrains the operation.

The XML transformation module is used in the processin the backstage transforming the surfaces that need to be transformed to the XML code.

The production contact surface module is needed after the user carries on tows entrains the produces to the contact surface.

The data storage module can store the data after the work of the models to the place assigned.So that the data can be used by the program following.

The demonstration controls module demonstrates the contact surface of the node and analyzes the controls.in the toolbar.

Key words： User Interface,Man-machine interaction,Script,Xml,
Windows Forms programming

目　　录

第一章　　绪论

1.1　项目背景

伴随着社会的发展和生产力的进步，电脑作为高科技产品的代表越来越进入到人们的生活中，无论是在普通的百姓的日常生活还是在社会生产的各个行业，电脑都越来越明显的发挥着作用，而伴着网络的飞速发展，电脑的作用也越来越突出，其逐步升级为人们生活的必需品，就连现在找工作的时候也是先要掌握电脑的操作，这样的技术已经成为了一种必需。随之飞速发展的便是软件的设计与开发实现，面对不同的行业，其使用的电脑软件的种类或者是侧重点都是不尽相同的，很多的软件都是面向特定的使用群体和使用范围，是有着不同的使用规范的，如一些财务上专用的软件、金融上的系统等等，这些都是软件发展的优势所在，也是其复杂程度的一个明显的体现，许多的专用软件就在这时应运而生，整个的系统是一个水利决策支持系统，其主要的功能是让一些水利方面的专家来使用，在他们并不是十分熟练地掌握计算机知识的情况下也可以方便而轻松地使用这套系统，来对某些特定水利形态的分析的基础上制定正确或者称之为合适的水利决策，供需要的水利方面的工程进行实施，在确定的过程中此软件起到了一个借鉴与建模的作用，对于这些水利专家的判断和决策有着十分重要的作用。

在软件的设计过程中，需要做的是一个用户图形界面生成器，其主要的作用是类似于 vs 中的界面生成器的那样的一个界面生成系统，设计中要实现一个工具箱的工具栏，用户可以拖拽工具箱中的工具进行编辑，生成自己需要的界面，用这个界面进行模拟和设计，从而达到好的效果，同时还要可以根据界面生成 XML 语言文档，在读取了满足其要求的 xml 文档之后还可以相应生成界面。

1.2　基本概念

1.2.1　用户界面

从软件设计的过程和方法来讲，软件设计总体可以分为两个主要的部分，首先是编码设计，也就是相当于后台的设计实现的编码；其次也就是作为软件设计的重要组成部分之一的用户界面（User Interface）设计，用户界面，又称 UI，在程序设计中起着至关重要的作用，但是其重要性却往往被人忽视，在设计的过程中，设计者常常关心的是编码设计，关心的只是功能是否完备，是否齐全，但是，界面设计的工作也占据着软件设计的相当大的比重，好的界面设计会使得软件的设计产生一个较大的提高，无论是在使用的便捷性方面还是在软件的普及度方面，界面设计都有着关键性甚至是决定性的作用，在功能都实现的类似的情况下，好的界面会带来更加便捷的操作性。

软件的产生也关系到人机之间的交互，只有当两方面交互得和谐友好的时候

这个软件才能够得到大多数人的认可和青睐，也就是说，一个软件设计的好坏的关键因素，就是其使用者的受众是否足够庞大，使用者是否感觉到软件的好用与方便，只有拥有大规模的使用者才是一个真正成功的软件设计，从这个方面来讲，用户界面设计就显得尤为的重要了。在新接触一个软件的时候，首先的第一印象是其界面看起来是否顺眼，然后便是使用起来是否方便，是否可以一目了然的对此软件的操作看得清清楚楚，可能有些人对界面的外表并不要求，但是很少有人会不在乎软件的操作的直观性，如果一个简单的功能却需要无比复杂的操作，这样的软件可能会让人很上火的。

用户界面的设计也要遵循几个原则，首先要注意的就是一致性，包括颜色的一致和功能操作的一致，一致的用户界面会使得使用者建立起关于应用程序工作流程的正确理解，而用户对应用程序工作流程的正确理解会使得对软件的掌握事半功倍。然后需要想到的是区域的组织排列等，要使得有一定相关联系的控件等在美观的前提下尽可能的放在一起，并要整齐排列，这样的话可以提高对用户的吸引力和使得用户的使用效率得到显著提高，此外还要注意字体、颜色对比度等等因素，这些因素都会对界面的成功与否起到关键的作用。

1.2.2 人机交互

提到人机交互，首先当然是说一下什么叫做人机交互，从网上查询的结果显示，所谓人机交互，是指通过计算机输入、输出设备，以有效的方式实现人与计算机对话的技术。它包括机器通过输出或显示设备给人提供大量有关信息及提示请示等，人通过输入设备给机器输入有关信息及提示请示等，人通过输入设备给机器输入有关信息，回答问题等。人机交互技术是计算机用户界面设计中的重要内容之一。它与认知学、人机工程学、心理学等学科领域有密切的联系。这是一个概念性的解释，人机交互，就是说用户在了解了某些机器或系统的要求之后按照自己的意愿对机器进行的操作，包括输入信息等等，这就是一个人操作机器的过程，也是操作的方法。在交互的过程中，人的作用是主观性的，人在充分满足了机器的约束后就可以按照自己的意愿进行设计和操作。

与用户界面相同，人机交互也有着优劣之分，按照网上查询的结果，机器是给人用的，无论什么时候人都是占据主导地位的一方，好的人机交互就是用户并不需要或者需要简单的培训就可以无阻碍的进行操作的过程，也就是说，自然人机交互模式是以直接操纵为主的、与命令语言特别是自然语言共存的人机交互形式，理想的人机交互模式就是“用户自由”，人们在使用的时候可以非常便捷地直接操作，而不是长时间的学习后才可以操作。当然很多时候这是很难做到的。

1.2.3 脚本语言

所谓脚本语言，是为了缩短传统的编写-编译-链接-运行过程而创建的计算机

编程语言。一个脚本通常是解释运行而非编译。脚本语言通常都有简单、易学、易用的特性，目的就是希望能让程序设计师快速完成程序的编写工作。

脚本语言的优点是，脚本语言不用编译就可以运行，非常便于修改，而编程序是一种经常性的活动，程序编完后总在不断的修改中，没必要搞的很隆重，还要编译。这样既节省了时间，也减少了程序的复杂和调试的繁琐程度，与系统语言如 C、C++等相比，它们有着一些区别，首先在类型定义上，系统语言需要强类型和静态类型定义，而脚本语言则要求不高，其使用较为松散的类型定义，没有类型声明，在运行时进行动态类型检查，除此之外还有前面说过的执行方面的区别，还包括体现在速度上的区别，脚本语言在易用性上得到了显著的提高，但也影响了它的执行速度，这也是很多人不喜欢脚本语言的原因之一。

脚本语言也分为几种，其中比较常用的有 perl、php、python、JavaScript 等等，这些语言都有着自己得天独厚的优势，在某些角度来看，它们都起着不可替代的作用。首先是 perl，其有着强大的字符串模式匹配，是最好的文本文件读取和生成语言，其有着较大的自由性，有着最大量的程序库，有着强大的数据库和其他形式的接口，这些都为 perl 提供了较大的使用空间，但是其也有个缺点是较为难懂，这也是很多人望而却步的理由。然后是 php，php 可以嵌入 html，可以较为容易地编写服务器端程序，而且其可以和 web 服务器及 MySQL 数据库相结合，不需要太多的人为的设置，方便了很多，还有个优点就是能够动态生成图像，这一系列的优点都使得其在现在的发展越来越迅速，得到了很多专业人士的青睐。再接下来是 python，这种语言是面向初学者的，较为易懂，但其有着丰富的语法，同时，它有着非常好的扩展性，python 程序可以与 C、Java 等程序很好的结合，它还适合编写较大的程序，所以当今使用 python 的人也是很多的。最后想要说的是 JavaScript 语言，在编写动态网页的时候常常用到它，无论是在验证的时候还是在一些网页有关的操作上，此语言都是得到最广泛的使用的，其方便性和快捷性都得到了承认。

以上便是对几种脚本语言的简单阐述，几种语言可以说是各有千秋，都有着自己无可比拟的优势，在使用时使用者可以根据自己的需要挑选适合自己程序的脚本语言。

1.3 论文的研究内容

本论文研究的是一种用户界面生成系统，通过这个系统，用户可以自己设计并开发出符合自己需要的界面。

电脑作为高科技产品的代表越来越进入到人们的生活中，无论是在普通的百姓的日常生活还是在社会生产的各个行业，电脑都越来越明显的发挥着作用，而伴着网络的飞速发展，电脑的作用也越来越突出，其逐步升级为人们生活的必需品，就连现在找工作的时候也是先要掌握电脑的操作，这是一种趋势，也是一种

必需。面对不同的行业，其使用的电脑软件的种类或者是侧重点都是不尽相同的，很多的软件都是面向特定的使用群体和使用范围，是有着不同的使用规范的，如一些财务上专用的软件、金融上的系统等等，这些都是软件发展的优势所在，也是其复杂程度的一个明显的体现，许多的专用软件就在这时应运而生，我们的这个系统就是这样的一个水利决策支持系统中的一部分，主要的功能是让一些水利方面的专家来使用，在他们并不是十分熟练地掌握计算机知识的情况下也可以方便而轻松地使用这套系统，来对某些特定水利形态的分析的基础上制定正确或者称之为合适的水利决策，供需要的水利方面的工程进行实施，在确定的过程中此软件起到了一个借鉴与建模的作用。这些看起来并不十分起眼，但对于水利方面的专家及相关工程师来说却是有着不可比拟的优点，因为其计算机知识并不是那么足够的。

总而言之，这篇论文所研究的内容是对于一种专用软件的一个部分的实现，这是一个很有潜力和研究价值的方向，专用软件也必将在未来的日子里产生更大的发展和飞跃，因此，此论文的研究是很有意义的。

1.4 论文的组织结构

本论文共分为五章，分别是：

第一章 绪论，主要讲的是所实现的系统的相关背景和所需要了解的相关知识，包括用户界面、脚本语言等等，这些都是完成系统需要初步了解的内容，同时还简单介绍了论文的研究内容和意义。

第二章 相关技术，主要包括 C#语言程序设计、XML 的相关知识、Windows Forms 编程等一系列开发时可能用到的知识和需要掌握的技巧。

第三章 图形用户界面生成工具的设计，主要是阐述一下图形用户界面生成工具的设计的思想与过程等内容。

第四章 图形用户界面生成工具的实现，这一章主要是描述一下具体的实现各个功能的过程，其中包括一些截图等等。

第五章 结论，此章主要是对于毕设的一个总体的总结归纳以及对于前景的展望等。

第二章　　相关技术

2.1　C#语言程序设计

本次的程序实现使用的是 C#语言，作为一种较新的语言，C#语言是从 C 和 C++发展而来，它汲取了包括 C、C++、Java 在内的多种语言的精华，是一种简单、完备、类型安全和完全面向对象的高级程序设计语言。

它的设计目标就是在继承 C 和 C++强大功能的同时，兼有 RAD（Rapid Application Development，快速应用程序开发）语言的简易和高效。作为.NET 的核心编程语言，C#充分享受了公共语言运行时所提供的优势，能够与其他应用程序方便地集成和交互。C#语言有着自己的很多的优势和特点，这些特点都使得 C#在应用中有着不可替代的作用。

首先 C#的语法较为简洁，C#中没有指针，这就让程序变得不是那么的难以理解。同时，简单的语法也让它在掌握起来不想其他的语言那么困难，C 和 c++程序员在学习 C#的时候障碍也就不会很大，因为 c#具有面向对象的语言的所有的基本特性，包括封装、继承、多态性等等，这样程序员使用起来也就更加得心应手了。

其次，C#还注重与 Web 的紧密结合。由于有了 Web 服务框架的帮助，对程序员来说网络服务看起来就像是 C#的本地对象，程序员们能够利用他们已有的面向对象的知识与技巧开发 Web 服务，仅需要使用简单的 C#语言结构 C#组件将能够方便地为 Web 服务并允许它们通过 Internet 被运行在任何操作系统上的任何语言所调用，举个例子，XML 已经成为网络中数据结构传送的标准为了提高效率C#允许直接将XML 数据映射成为结构这样就可以有效地处理各种数据。第三 C#具备完全的安全性和错误处理的能力，这两项能力是衡量一种语言的优秀程度的重要依据，在程序员由于不小心或者是其他的各种原因出现类似忘记变量初始化、对自己无法改变的内存空间进行修改等这样那样的错误时，可能会产生一些难以预见的错误，C#的设计思想使得可以消除类似上面所说的常见的错误，通过 C#这一特性，程序员可以通过较之于其他语言更少的代码来完成相同的功能，这样的话编程人员的工作量就会大大减少，错误也会被避免。

提到变量，其是类型安全的 C#中不能使用未初始化的变量，对象的成员变量由编译器负责将其置为零，这减少了错误的概率。在兼容性方面，C#允许与 C 风格的需要传递指针参数的 API 进行交互操作，这些互操作性使得 C#方便而不失其安全性。

2.2　XML

要完成本次的用户界面生成器系统的设计实现，需要使用 XML 的相关技术，因为要使之能够转化到 xml，又要可以从 XML 文件转化到界面程序中。而提到

XML 相关的内容，其实有很多，下面就几项主要的简单阐述一下。

2.2.1 概念和应用

XML 是一种被设计用来传输和存储数据的语言形式，其全称为可扩展标记语言，英文称作 Extensible Markup Language，虽然其可被称之为语言，但是在通常情况下，其不具备常见语言的基本功能——被计算机识别运行，想要体现出 XML 想要得到的效果，就需要依靠另一种语言来解释它以得到最终的目的或结果。

XML 主要应用于文档型和数据型两种，这两种应用就是对 XML 的结构和定义的一个较为全面的诠释。其应用的具体主要在以下几个方面：

1、自定义的 XML 和 XSLT，然后将之相结合，生成 html 文档，这是很常见的文档型应用。XML 中存放的是整个文档的 XML 数据，其起到一个数据存储的作用。然后通过 XSLT 对 XML 进行转换和解析，在这个过程中，结合 XSLT 中的 html 的标签，最终转化生成 html 显示在浏览器上，这个应用的发展使得页面得到了进一步的前进，也促进了 html 的发展壮大。

2、XML 作为信息传递的载体。之所以称之为载体，是因为这些应用是以 XML 作为基本形态，但是其都已经发展出具有特定意义的格式形态，并不是完全按照 XML 的形态来进行的。在 web service 中，就是将数据包装成 XML 传递，但其中的 XML 有着自己的特定规格，也就是 soap，这充分说明了 XML 的信息载体的身份。当然，除了 web service 以外，AJAX 等其他的应用都使用到了 XML 的这个特性。

3、作为数据型的应用来说，XML 可以作为一个微型的数据库。XML 有着较多的 API 供用户选择来进行 XML 的存取和查询。但是要注意，XML 数据库与数据库系统有着不同，而在新版本的数据库系统中 XML 已经成为了一种数据类型，而完全以 XML 相关技术为基础的数据库系统也初见端倪。

4、应用程序的配置信息数据存储。在 J2EE 配置 web 服务器时需要使用到 web.XML 文件，在这个应用中，需要将需要的数据存入到 XML 中，XML 这时候起到一个存储数据的作用，然后将 XML 在应用程序运行载入，最终根据要求、根据不同的数据进行相应的操作。需要关注的是，在这里的配置信息的存储过程中，这些配置信息是较为静态的，其通常是不变化的。

除了这些的应用之外，XML 还可以用于一些其他文档如 word 和 excel 等生成 XML 格式等其他的一系列的应用，这些应用基本涵盖了 XML 的主要的用途。总而言之，XML 不像程序语言那么具体，需要从应用入手得出自己想要使用的用途。

2.2.2 xml 文档编写

Xml 文档写起来并不是十分复杂，但是 xml 是拥有一定的语法规则的，编写的过程也是有着一定的步骤的。

Xml 文档编写是有着各种检验的，首先是格式良好性的检验，其要求主要有：

（1）文档不可以为空。文档中没有了元素则此文档就失去了存在的意义。

（2）只能有一个根元素。以免引起歧义。

（3）元素之间的嵌套关系要正确。

（4）标签要对称。

（5）属性值用引号括起来。

（6）特殊字符必须用实体引用。

2.2.3 xml schema

XML Schema 是用来定义 XML 格式，构造的语言的一种。有 W3C 加以开发，标准定义。XML Schema 本身也是一种 XML 构造，它用来描述[哪个元素，在什么时候出现]，[该元素具有什么样的属性]等等，也就是说，XML Schema 是对 XML 的树形构造加以描述说明的一种语言。原本，使用 DTD 对 XML 的树形构造加以描述说明，但 DTD 存在严重的局限性，DTD 不能定义数据的类型，语法也与 XML 语言完全不一样，在使用的便利性，数据结构表达的严谨性上存在问题。

XML 作为数据载体，可以用来描述各种各样的数据。在系统开发中，可以使用 XML 在系统（或不同功能模块）之间传递数据，也可以使用 XML 作为配置文件，数据文件等，XML schema 文件描述了 XML 文档的结构、定义了可以出现在文档里的元素、定义了可以出现在文档里的属性、定义了哪些元素是子元素、定义了子元素的顺序、定义了子元素的数量、定义了一个元素应是否能包含文本，或应该是空的、定义了元素和属性的数据类型、定义了元素和属性的默认值和固定值，总而言之，XML Schema 是一个基于 XML 的语法或 schema 规范，用来定义 XML 文档的标记方式。XML Schema 是一个由 Microsoft 建议的 schema 规范，它与文档类型定义（DTD）相比具有很大的优势，而 DTD 是最初用来定义 XML 模型的 schema 规范。DTD 存在很多缺点，包括使用非 XML 语法，不支持数据类型定义，不具有扩展性等。例如，DTD 不允许把元素内容定义为另外一个元素，或字符串。XML Schema 从几个方面改善了 DTD，包括使用 XML 语法，支持数据类型定义和名域。

XML Schema 和名域 Schema 是一些规则的集合(也称为语法或者语汇)，其中包括了类型定义(简单和复杂类型)以及元素和属性声明。由于 XML 中可能存在不同的语汇来描述不同的元素和属性,因此需要使用名域(namespace)和前缀来

避免元素和属性声明之间的模糊性。

在 schema 元素的导言中可能包含三个可选的属性。例如，下面的语法使用的 schema 元素引用了三个最常使用的名域：前两个属性用 XML 名域来标识 W3C 中的两个 XML schema 规范。第一个 xmlns 属性包含了基本的 XML schema 元素，比如 element, attribute, complexType, group, simpleType 等。第二个 xmlns 属性定义了标准的 XML schema 属性类型例如 string, float, integer, 等。

XML 模式是指用来描述 XML 结构、约束等因素的语言，例如 XML Schema、XML DTD、XDR，SOX 等等。XML 格式则是 XML 文档本身所具有的格式。本文以 XML Schema 来代表 W3C 所推荐的 XML Schema 模式标准，而以"XML 模式"来代表所有的 XML 模式描述语言。

模式必须以某种格式来表示，XML Schema 的格式与 XML DTD 的格式有着非常明显的区别，XML Schema 事实上也是 XML 的一种应用，也就是说 XML Schema 的格式与 XML 的格式是完全相同的，而作为 SGML DTD 的一个子集，XML DTD 具有着与 XML 格式完全不同的格式。这种区别会给 XML Schema 的使用带来许多好处：XML 用户在使用 XML Schema 的时候，不需要为了理解 XML Schema 而重新学习，节省了时间；

由于 XML Schema 本身也是一种 XML，所以许多的 XML 编辑工具、API 开发包、XML 语法分析器可以直接的应用到 XML Schema，而不需要修改。

2.3 Windows Forms 编程

Windows Forms(Windows 窗体)是一个新的窗体包，它使得开发人员可以创建基于 Windows 的应用程序，来充分利用 Microsoft Windows 操作系统中丰富的用户界面特性。Windows Forms 是新的 Microsoft .NET Framework 的一部分，它使用了许多新技术，包括一个公共应用程序框架、受控的执行环境、集成的安全性和面向对象的设计原则。此外，Windows Forms 完全支持快速、容易地连接 XML 网络服务和在 ADO.NET 数据模型基础上创建丰富的、数据感知(data-aware)的应用程序。利用 Visual Studio 中新的共享开发环境，开发人员可以使用任何支持.NET 平台的语言，包括 Microsoft Visual Basic 和 C#创建 Windows Forms 应用程序。

可视化集继承是 Windows Forms 中的一个重要的新特性，它将提高开发人员的生产力，促进代码的重用。例如，一个组织可以定义一个包含诸如公司徽标，可能还包括一个公共工具栏等项目的标准的基本窗体。这个窗体可以通过继承由开发人员使用，并进行扩展以满足特定应用程序的需求，而同时在组织内使用公共的用户界面。基本窗体的创建者可以指定哪些元素可以被扩展，哪些元素必须按原样使用，这样可以确保窗体能够被适当地重用。

当开发人员设计 Windows Forms 应用程序的外观和感觉时，将拥有空前水平

的控制能力和生产力。菜单设计器(Menu Designer)、控件锚定(Control Anchoring)、控件入坞(Control Docking)和其他特性使得开发人员可以更精确地创建基于 Windows 的用户界面。

利用菜单设计器，开发人员可以既快速又容易地给窗体添加菜单，修改菜单，然后在不允许应用程序的情况下查看菜单的外观。利用控件锚定，窗体上的控件将更有效，从而使得窗体能够在用户调整窗体的大小时自动调整控件的大小。利用窗体入坞(Control Docking)特性，控件可以停靠在窗体的任何一侧，从而在布局方面提供了更大的灵活性。

在 Windows 程序中，菜单是一个必不可少的程序元素。通过使用菜单，可以把对程序的各种操作命令非常规范有效的表示给用户。一个 Windows 程序菜单一般包括一个主菜单（主菜单下面包含许多子菜单）及很多弹出式菜单。单击菜单项程序将执行相应的功能；另外在程序窗体的许多地方单击鼠标右键将会弹出一个针对性的快捷菜单（也可以称为弹出式菜单），单击将执行相应的功能，使得软件的应用变得更加简单，更加人性化。Microsoft 视窗系统及其应用软件站稳桌面系统及软件的霸主地位，友好的菜单应用立下不小的功劳。

与传统的 Windows 程序一样，Windows Forms 程序中的菜单也包括一般的菜单及弹出菜单两种。一般的菜单主要是指窗体的主菜单及子菜单。菜单从属特性外观及主要功能又可分类：命令式菜单、选择菜单、链接菜单及分隔菜单等等。大家一定使用过快速开发（RAD）语言工具（如 VB、Delphi 等）设计过菜单程序。使用这些开发工具，只需要把用鼠标把菜单控件拖放到窗体上，并通过简单的菜单外观及属性等设置即可完成非常友好的菜单程序开发。在.Net 中，C#是支持快速开发的，因此，通过使用相应的开发工具(如 Visual Studio .Net)也可以通过简单的鼠标拖放操作完成大部份的菜单程序设计。当然，由于 C#的强大的功能，使用.Net 提供的菜单类，也可以通过记事本等纯文本编辑工具就能写出具有良好用户界面的菜单程序。

第三章　　图形用户界面生成工具的设计

3.1　需求分析

图形界面是计算机系统的重要组成部分，其设计的成功与否直接关系到系统的可使用性和使用的效率，从商业方面来讲，它甚至可以影响软件系统的销售和应用的广泛性。有经验表明，高质量的用户界面的设计开发是一项周期较长且需要大量人力物力的工作，在过去，设计的时候要求程序员同时混合编写用户界面及功能两个方面，这使得程序过于复杂，也使得程序员的压力过大，从而可能忽略了对用户界面的关注与思索，使得其实用性和维护性都大大不能满足条件。当前计算机软件的开发已不再将用户界面和应用功能两部分混合编写，越来越倾向于二者分别编写。这是因为对不同的应用系统，用户界面部分在逻辑上和处理方法上具有高度的相似性，让软件开发人员花费大量的时间与精力去开发一个很类似又不具有通用性的用户界面程序显然是不可取的。要使用户界面的生成独立于应用功能的开发，具有强大的用户界面开发能力的工具是非常必要的，用户最好能够通过这样的工具方便地构造符合自己意愿的界面外观并能方便地对界面进行维护，本次毕设要完成的工作便是实现这样的一个系统。

3.1.1　图形界面描述语言

图形界面的设计这可以用一种专门的界面描述语言来对界面进行说明，此语言的形式可能会不只一种，其主要有菜单网络、状态转换图、事件语言以及面向对象语言等，无论使用哪种语言，其都是为了说明界面间的语法，即输入输出动作的合法顺序的一个简单的描述。下面是这几种描述语言的简单描述。

1、菜单网络。这是一种最简单的表示方法，表示支持菜单的层次或网络结构。

2、状态转换图。在由每个状态出发的边上标志出一个输入词码，它能引起转移到该边另一端的状态，使用这种方法，界面和应用程序的联系是通过很多全程变量实现的，所有的状态必须有清楚的边来反映所有可能出现的错误输入和公用命令。

3、事件语言。在这种语言中，输入词码被认为是事件，且这些事件将会被送到事件处理器中，处理器的作用是产生输出事件或调用子程序等。

4、面向对象语言。这个说法很熟悉，当前的主流的程序设计语言便是面向对象的，这里的面向对象语言与程序设计语言都是有着异曲同工之美的，其作用是提供一个面向对象的框架，而设计者是在此框架中编写界面程序的。这些都是类似于面向对象程序设计语言的。

3.1.2 图形界面生成工具

在前面曾经提到过，当前软件开发的设计思想已经倾向于图形界面与应用功能两部分分开实现，而不再是曾经的那种混合编写的方法。用户界面独立于应用程序，这使得具有强大的用户界面开发能力的图形用户界面生成工具就显得非常必要了。用户可以通过此工具方便的构造符合自己意愿的界面外观，从而会减少开发人员在界面生成方面工作的时间。除此之外，由于实现的系统的使用者是一些水利方面的专家，他们并不一定会掌握很多的计算机的知识，而通过图形用户界面生成工具，他们不需要专门掌握上述的图形用户界面描述语言，而是通过界面编辑器来直接操纵屏幕上的可视对象，生成用户界面原型，并能将结果立即显示在屏幕上，达到“所见即所得”的效果，这些也都是图形用户界面系统的优点和易用性的体现。

用户界面生成工具和功能实现之间如图 3-1 所示。

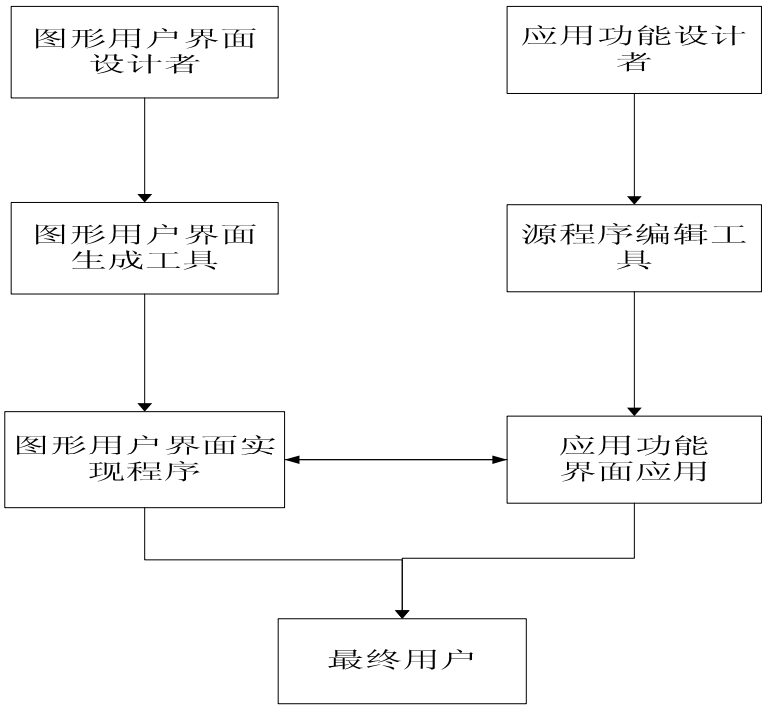


图 3-1　　界面与功能

总而言之，界面的设计和实现也是软件设计中必不可少甚至是至关重要的部分。

3.2 概要设计

首先是明确一下程序的需求，题目要求中是设计并实现一个用户界面生成系统，这个系统的主要功能是建造一个类似工具栏的控件库，用户在使用的时候运用拖拽的方法将控件放到新建的空白的画板上，然后按照要求生成 xml 文档或者

是生成界面文件，还有的需求就是要可以将 xml 文件导入到程序中，生成界面，这些都是需求上要求的功能。具体的要求是要按照图形界面的设计原则将界面设计出来，使得用户在使用的时候能够明晰的进行操作而不至于产生歧义，设计中借鉴 vs 中的工具箱界面的设计思路和整体风格，因为那个工具箱的使用已经被很多的用户所接受，其必然有着自己的优势和特色。用户界面的操作的流程示意图如图 3-2 所示。

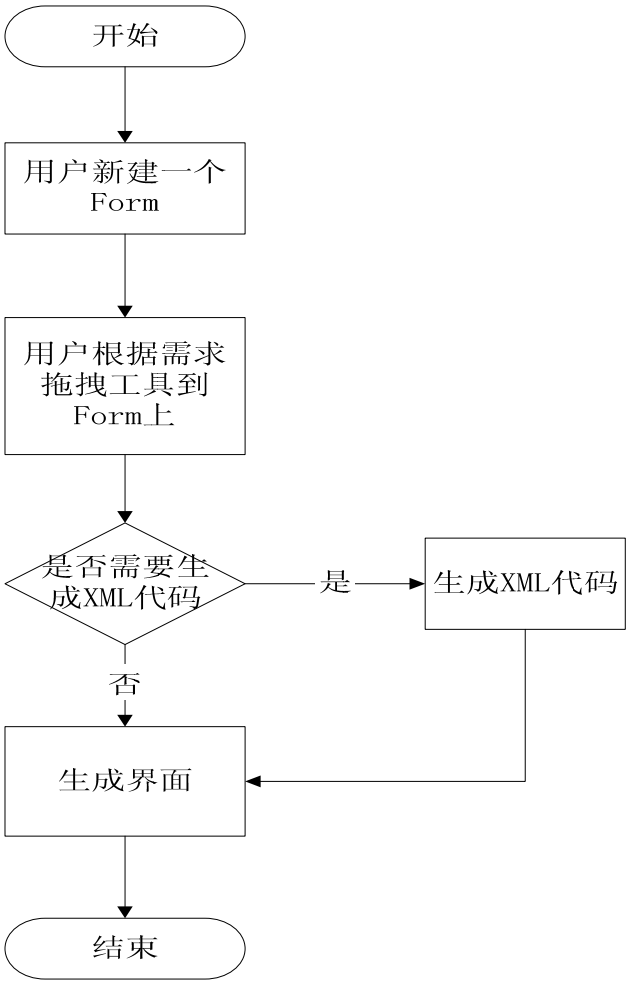


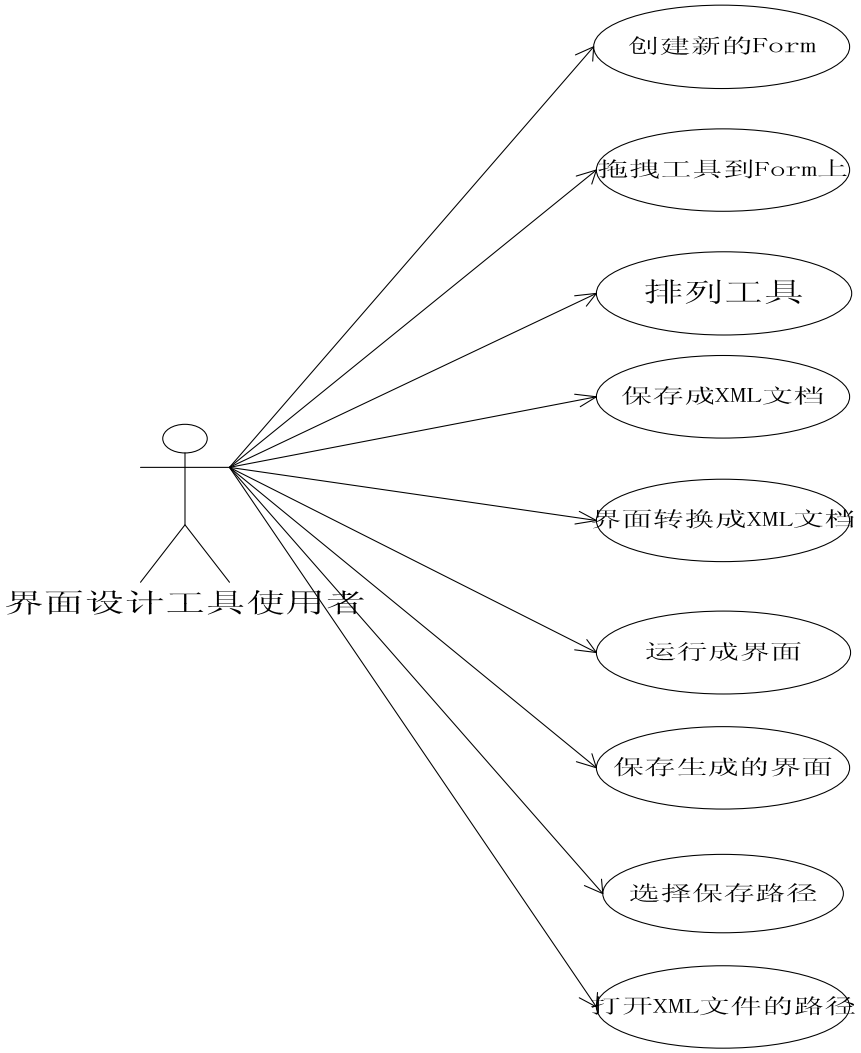
图 3-2　　界面操作流程

3.2.1　用例分析

用例分析是软件设计中一个不可或缺的部分，通过用例分析的过程可以看到软件的编写者和使用者都需要完成怎样的操作或者说是用法，这样的话就可能能够发现一些原本没有发现的问题，从而获得对软件开发的完全性和完善性的提高。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

首先，画出此界面设计工具使用者的用例图，使用者要做的是排列工具、生成界面、保存 XML 文档等等。图 3-3 给出使用者的用例图。



-

图 3-3　　界面设计工具使用者用例

其次，要画出界面设计工具设计者的用例图。其要做的是对界面的 XML 文档转换、保存等等工作。图 3-4 显示的是设计者的用例图。

3.2.2　运行流程

在图形界面设计工具运行的时候，操作是由用户来控制的，用户要完成的就是建立 Form、拖拽工具和排列各个工具、点击生成 XML 文档等等这些操作，而并不关心工具完成这些操作的过程，实际上我们在设计的时候的目的也是这样的，并不想要用户完全理解全过程，因为用户的计算机水平和知识可能并不那么高。

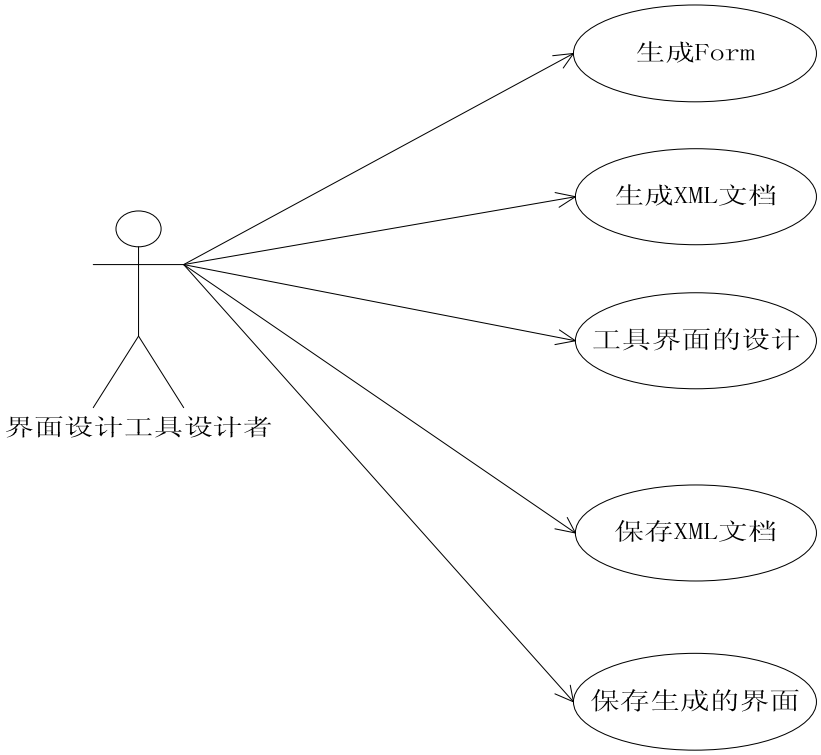


图 3-4　　界面设计工具设计者用例图

在设计图形界面生成工具的界面的时候，使用的是 Windows Forms 编程的方法来进行，同时还要结合 vs 平台中的图形界面的设计的功能，这个功能可以方便简洁的进行设计和完成界面的设计工作。在具体的设计中，运用 Windows Forms 编程的方法建立一个大的界面，然后通过一些特定的函数来完成菜单栏的设计，根据一般的菜单栏的设置，我们的图形界面生成工具的菜单栏应该包括文件、编辑、代码转换等按钮，这些按钮都是用户需要的。在文件按钮的下面，按照惯例应该有打开、新建、保存、另存为、退出等，而在编辑的按钮下弹出的应该有复制、粘贴、剪切、删除等等，用以对界面进行一些操作。而在代码转换的相关按钮中应该是主要有 XML 的转换相关操作按钮，这些都是需求必要的。

图 3-5 表示点击操作的简单流程示意图。

图 3-6 是编辑操作的简单流程示意图。

3.2.3　模块划分

整个图形界面生成工具具体可以划分为几个大的模块，这些模块之间的关系是相互呼应相互联系的，通过模块之间的调用和配合，对程序的运行起到一个完成和进行的动作。我将图形界面生成工具大致分成显示框架模块、工具栏模块、XML 转换模块、生成界面模块、数据存储模块、显示控件模块六个，这些的模块各有各的功能，实现的方式和结果都不同，相互适应相互调用之后达到相应的目的。每个模块的功能具体解释如下：

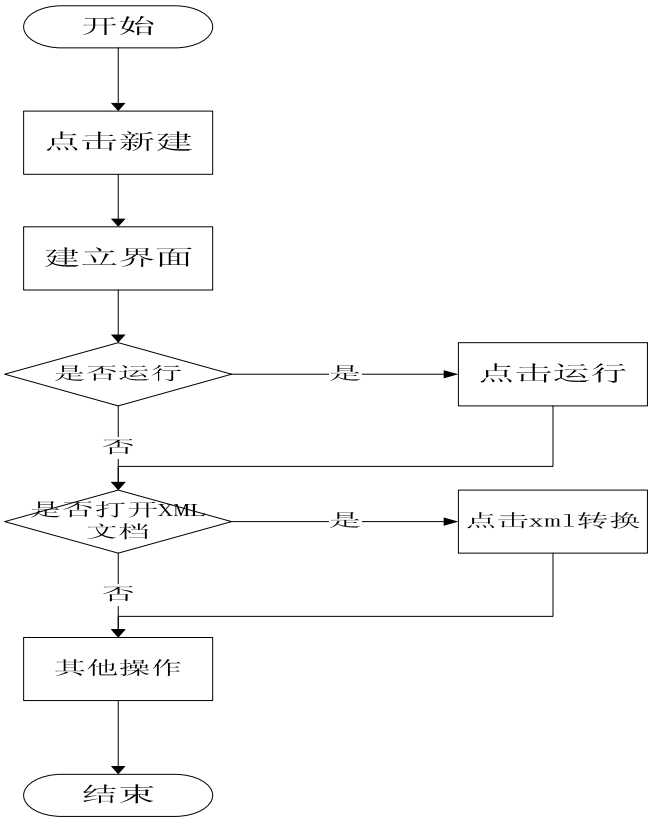


图 3-5　点击操作流程

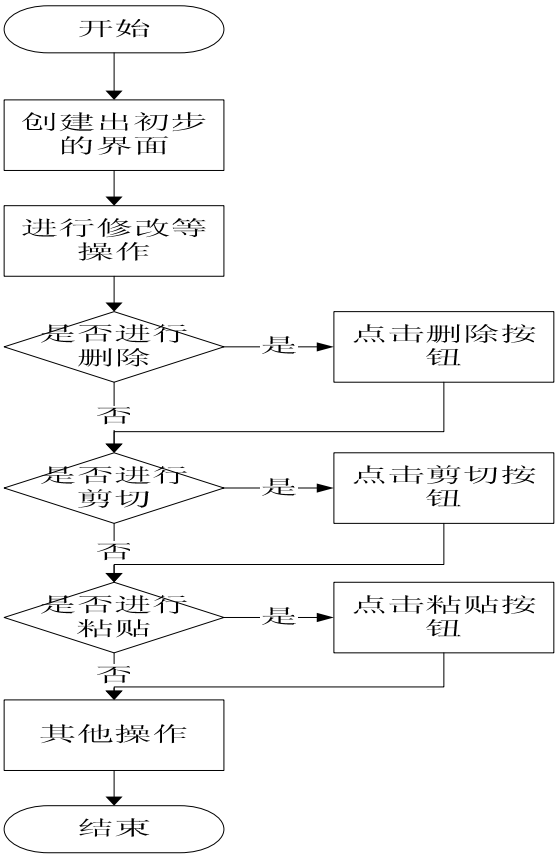


图 3-6　编辑按钮操作流程

1、显示框架模块。这个模块负责整个界面的布局，确定图形界面和控件界面的大小等等。

2、工具栏模块。这个模块是用于展示所有的工具选项，给用户展示工具控件，在其背景后完成在用户完成拖拽操作后自动生成的代码。

3、XML 转换模块。此模块用于在后台将需要转换成 XML 代码的界面进行转换的过程中，作用就是进行转换。

4、生成界面模块。此模块是在用户进行拖拽过程后生成界面的必需模块。

5、数据存储模块。这个模块可以将模型处理后的数据保存到指定的文件，用于接下来的程序调用。

6、显示控件模块。显示节点的界面，解析出工具栏中的控件。

图 3-7 表示各个模块间的关系：

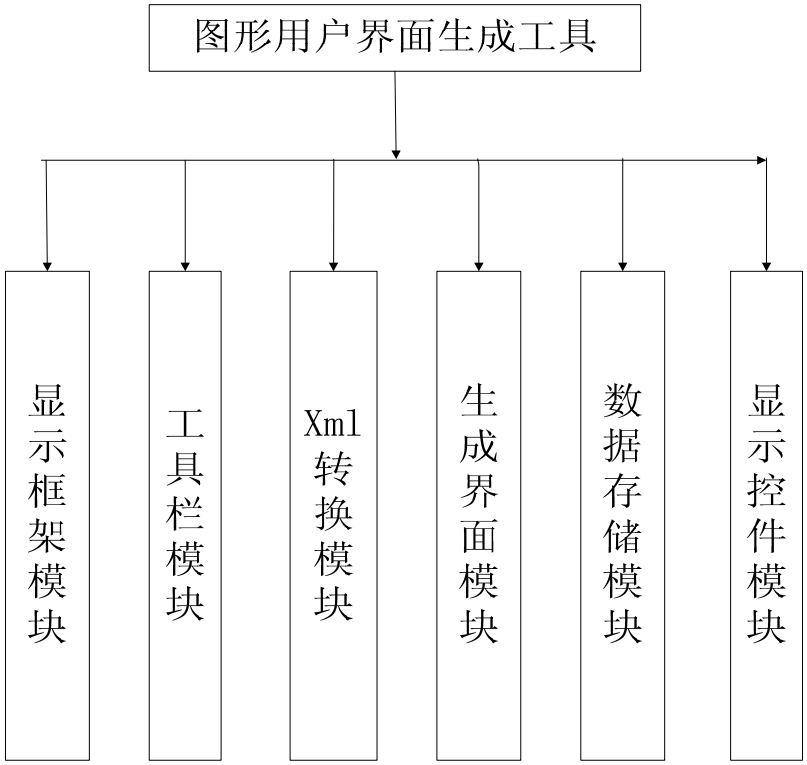


图 3-7　模块划分

经过上面的设计过程，接下来要进行的便是实现系统的过程了，在前面的设计的思想的指导之下，我开始按照设计的步骤一步步地进行编码实现工作，在实现的过程中我还适当地对有些地方的设计进行了一些或大或小的改变。

第四章　　图形用户界面生成工具的实现

整个图形界面生成工具的界面是由 Windows Forms 编程结合，以 vs 平台中的工具栏拖拽的界面为标准，那个界面已经被很多人所熟知，也使用的较为熟练了。

具体的实现中，菜单栏就是按照前面设计的那样，在用户使用的时候，按顺序来进行操作。图 4-1 给出三菜单栏的截图。

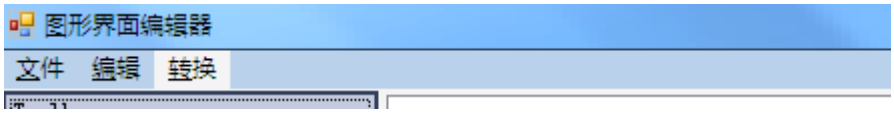


图 4-1　菜单栏截图

4.1　文件按钮

文件，出现各个需要的按钮点击产生效果。在文件按钮的下面实现的是六个不同的操作，分别是新建、类型、打开、另存为、运行和退出，分别的作用是：

1、新建：用于新创建一个 Form 来使得用户可以在上面进行工具控件的拖拽等。

2、类型：在新建之前点击，选择一个模式进行，选择之后才可以最后决定是运行出界面还是得出 XML 文档代码。

3、打开：用于找到一个 XML 文档，导入到程序中，产生界面的结果。

4、另存为：用于在 XML 模式下将 XML 文档存储到指定的路径。

5、运行：用于将运行模式下的界面存储到指定路径，并运行出结果。

6、退出：用于程序退出。

图 4-2 给出最终实现的过程的一个流程示意图。

在具体的实现过程中，首先使用 Windows Forms 编程建立界面，在后台的逻辑过程的编辑中，要分为 XML 转换模式和运行模式两种，两种的模式要调用的是两个不同的类，在这两个类中，分别用不同的函数和方法来进行编写，因为在 XML 转换模式下，需要进行转换的函数，而运行模式则是需要进行将界面的展示过程 。下面就利用一点代码来说明一下。

对于运行模式，其主要目的是为了将设计出的界面运行出结果并存储在一定的路径中。下面的这一小部分的函数用于载入一个空白的 Form：

```
DesignSurface ds = new DesignSurface();
ds.BeginLoad(typeof(Form));
IDesignerHost idh =
    (IDesignerHost)ds.GetService
    (typeof(IDesignerHost));
```

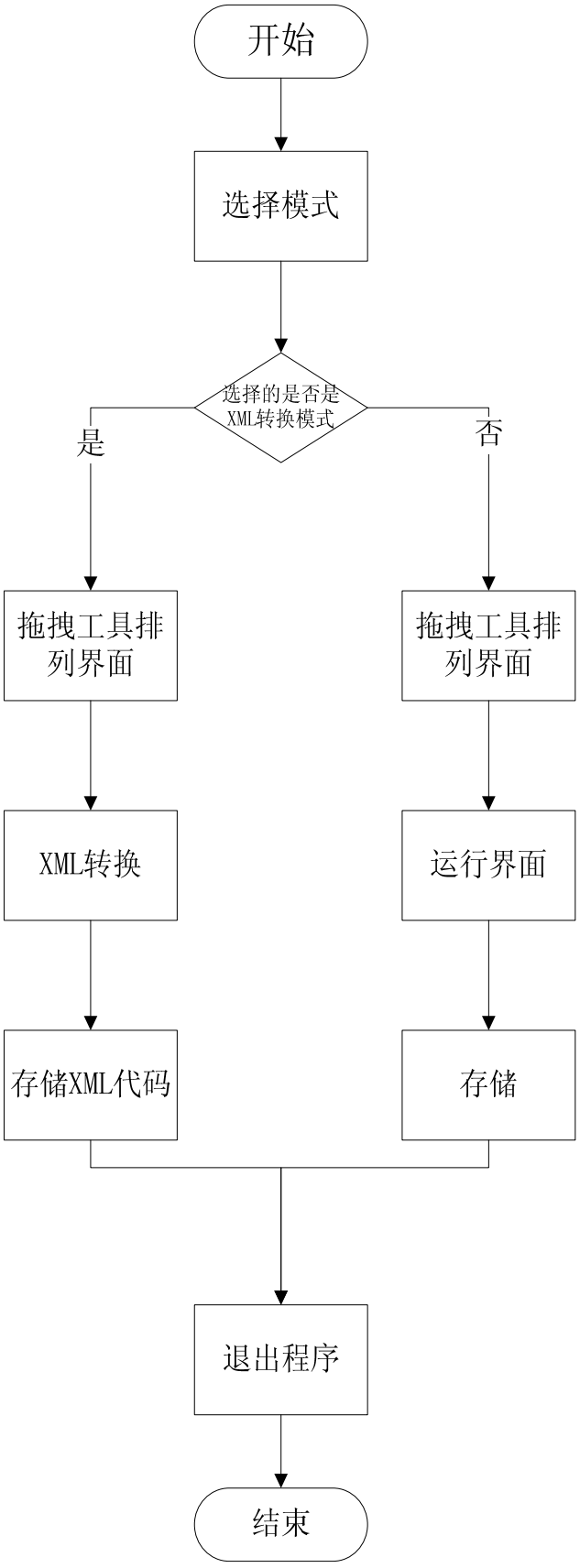


图 4-2　菜单栏操作流程示意图

```
idh.RootComponent.Site.Name = "Form1";
cg = new CodeGen();
ccu = cg.GetCodeCompileUnit(idh);
AssemblyName[] names =
    Assembly.GetExecutingAssembly().
    GetReferencedAssemblies();
for (int i = 0; i < names.Length; i++)
{
    Assembly assembly = Assembly.Load(names[i]);
    ccu.ReferencedAssemblies.
       Add(assembly.Location);
}
codeCompileUnit = ccu;
return ccu;
```

这一小段的代码中，使用了系统自带的几个函数和 DesignSurface 等其他的类函数进行一个 Form 的新建过程，这个 Form 的作用就是用户将工具放在其上的，建立的过程并不十分复杂。

这个运行模式与 XML 转换模式最主要的特点就是其可以生成可执行的文件，实现这个功能的主要的代码如下：

若可进行执行操作，则：

```
SaveFileDialog dlg = new SaveFileDialog();
dlg.DefaultExt = "exe";
dlg.Filter = "Executables|*.exe";
if (dlg.ShowDialog() == DialogResult.OK)
{
   executable = dlg.FileName;
}
```

否则的话，则进行下面的运行：

```
CompilerParameters cp = new
CompilerParameters();
AssemblyName[] assemblyNames =
   Assembly.GetEntryAssembly().
      GetReferencedAssemblies();
foreach (AssemblyName an in assemblyNames)
{
```

```
            Assembly assembly = Assembly.Load(an);
            cp.ReferencedAssemblies.
                Add(assembly.Location);
        }
        cp.GenerateExecutable = true;
        cp.OutputAssembly = executable;
        cp.MainClass = "DesignerHostSample." +
            this.LoaderHost.RootComponent.Site.Name;
        CSharpCodeProvider cc = new
            CSharpCodeProvider();
        CompilerResults cr =
            cc.CompileAssemblyFromDom(cp,codeCompileUnit);
```

图4-3给出用这个模式进行操作的截图。

图4-3　运行模式结果

在运行前要进行可执行文件的保存行为，图 4-4 是保存路径的截图。

在 XML 转换模式下的运行和操作过程和前面有所不同。

给出一小点代码，这些代码是为了读入 XML 文件，将其中的数据等放到一定的存储介质中用于以后使用。

```
        StreamReader sr = new StreamReader(fileName);
        string cleandown = sr.ReadToEnd();
        cleandown = "<DOCUMENT_ELEMENT>" + cleandown +
```

```
    "</DOCUMENT_ELEMENT>";
XmlDocument doc = new XmlDocument();
doc.LoadXml(cleandown);
 foreach (XmlNode node in
    doc.DocumentElement.ChildNodes)
 {
    if (baseClass == null)
    {
        baseClass =
        node.Attributes["name"].Value;
    }
    if (node.Name.Equals("Object"))
    {
        ReadObject(node, errors);
    }
}
     document = doc;
```

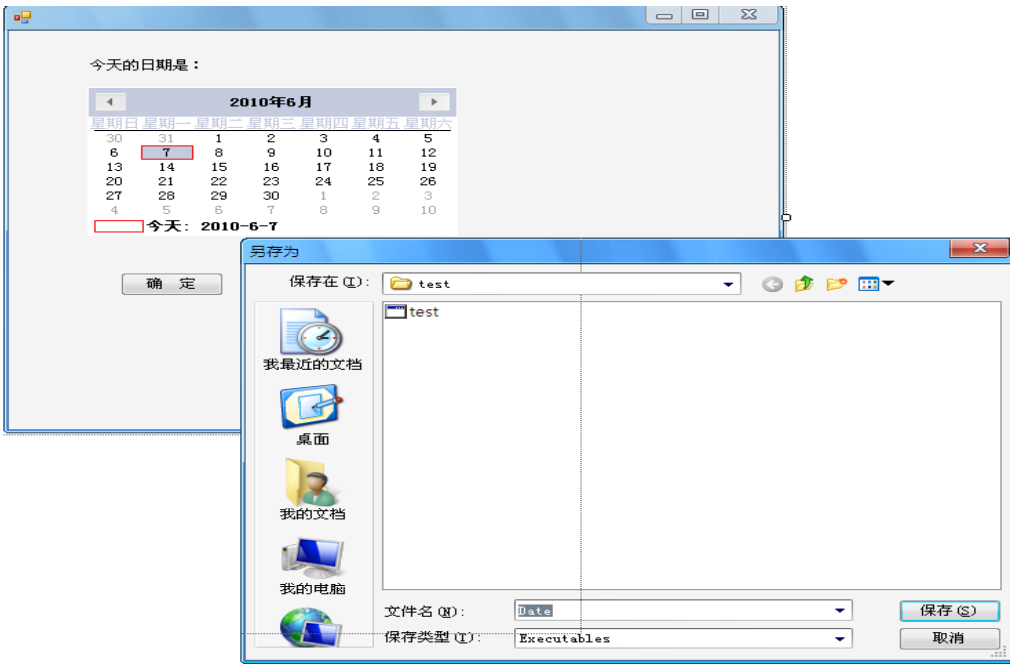


图 4-4　运行模式保存路径

接下来的函数是 GetCode，这个函数用来获取 XML 的代码，传入相应的参数，这个函数被调用用以完成代码的转换过程。

```
StringWriter sw;
```

```
sw = new StringWriter();
XmlTextWriter xtw = new XmlTextWriter(sw);
xtw.Formatting = Formatting.Indented;
xmlDocument.WriteTo(xtw);
string cleanup =
    sw.ToString().Replace
        ("<DOCUMENT_ELEMENT>", "");
cleanup = cleanup.Replace
        ("</DOCUMENT_ELEMENT>", "");
sw.Close();
```

图 4-5 给出操作的截图。

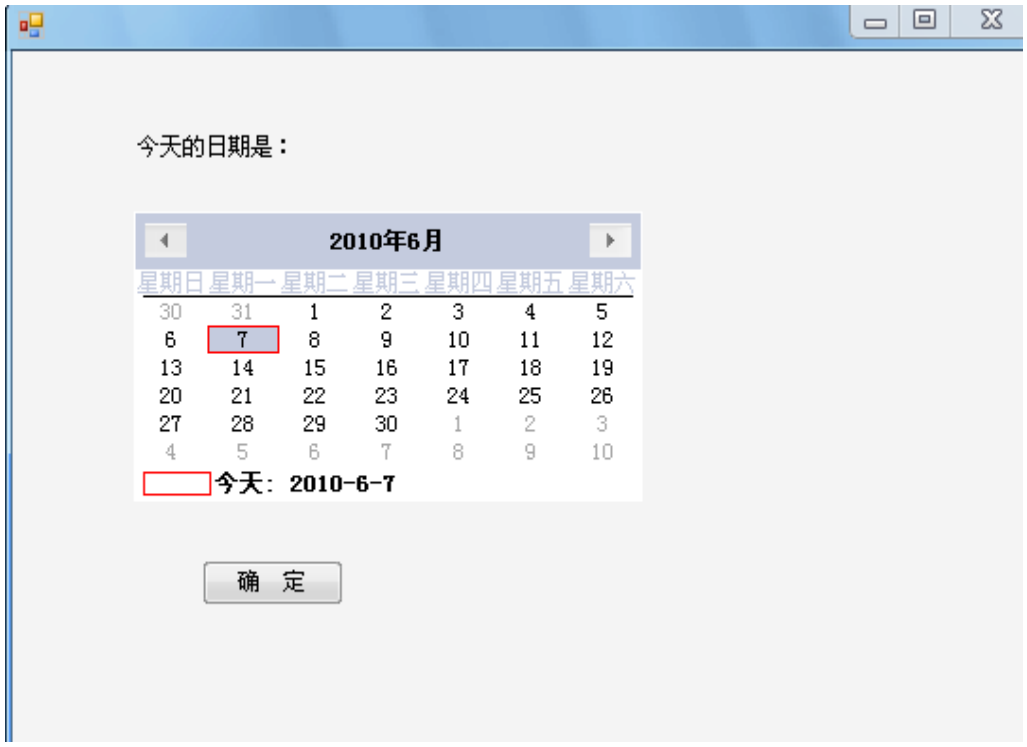


图 4-5　XML 转换模式设计

上面的例子是与前面的运行模式基本相同的，为的是做一个简单的比较。图 4-6 给出经过 XML 转换的结果，此结果是显示在新的标签中的。

在运行完成为 XML 代码时，还可以另存为 XML 文档文件，在存储时也是需要选择保存路径的，图 4-7 给出选择存储路径时的截图。

```
<Object type="System.Windows.Forms.Form, System.Windows.Forms, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a
  <Object type="System.Windows.Forms.Button, System.Windows.Forms, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b7
    <Property name="TabIndex">2</Property>
    <Property name="Name">Button1</Property>
    <Property name="Size">75, 23</Property>
    <Property name="UseVisualStyleBackColor">True</Property>
    <Property name="Text">确　定</Property>
      <Property name="Location">100, 254</Property>
      <Property name="DataBindings">
        <Property name="DefaultDataSourceUpdateMode">OnValidation</Property>
      </Property>
    </Object>
    <Object type="System.Windows.Forms.MonthCalendar, System.Windows.Forms, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c5619
      <Property name="DataBindings">
        <Property name="DefaultDataSourceUpdateMode">OnValidation</Property>
      </Property>
      <Property name="Name">MonthCalendar1</Property>
      <Property name="Location">64, 80</Property>
      <Property name="TabIndex">1</Property>
    </Object>
    <Object type="System.Windows.Forms.Label, System.Windows.Forms, Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"
      <Property name="TabIndex">0</Property>
      <Property name="Size">100, 23</Property>
      <Property name="Text">今天的日期是：</Property>
      <Property name="Location">64, 42</Property>
      <Property name="DataBindings">
        <Property name="DefaultDataSourceUpdateMode">OnValidation</Property>
      </Property>
      <Property name="Name">Label1</Property>
    </Object>
    <Property name="Name">Form1</Property>
    <Property name="DataBindings">
      <Property name="DefaultDataSourceUpdateMode">OnValidation</Property>
    </Property>
    <Property name="ClientSize">539, 402</Property>
</Object>
```

图 4-6　XML 转换代码图

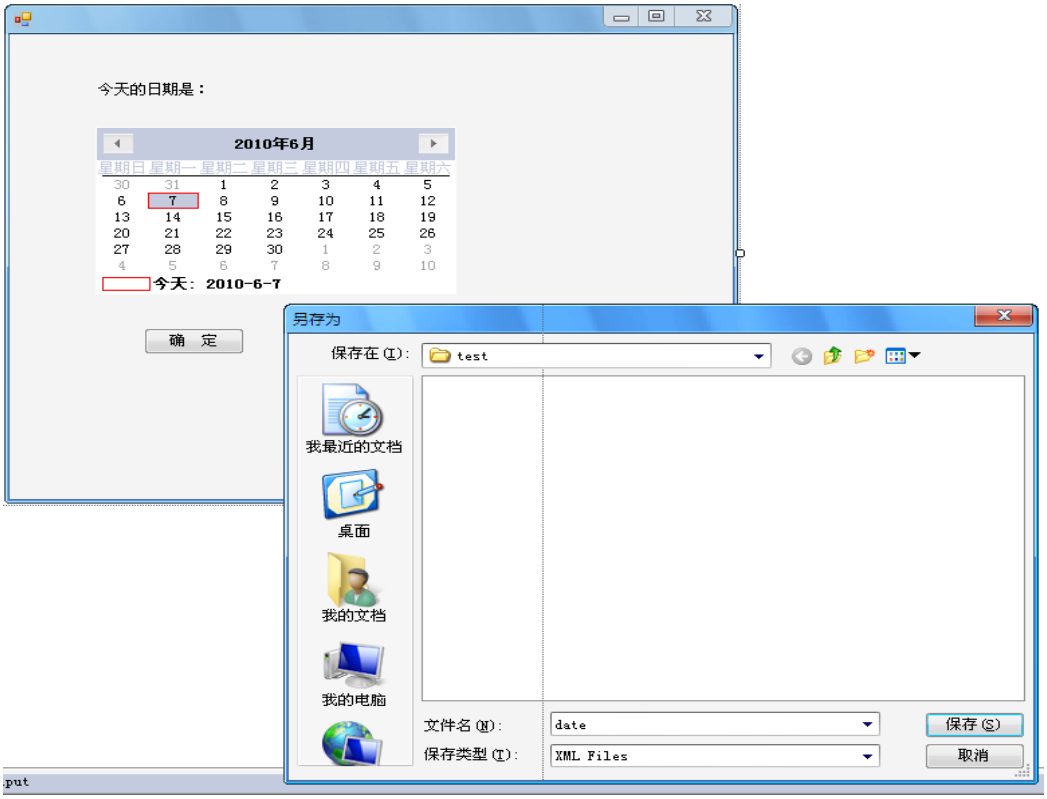


图 4-7　XML 转换模式保存路径

在文件按钮的下面还有一个打开按钮，其作用是选择打开已经存在的 XML 代码，生成界面的图形，图 4-8 显示的是打开的界面。

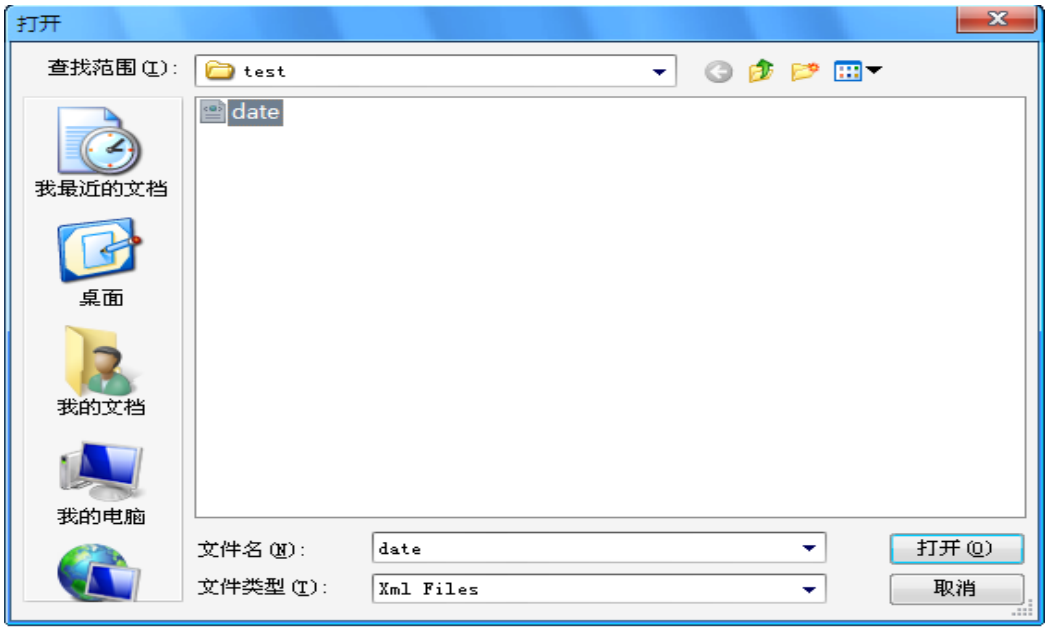


图 4-8　打开 XML 文档

图 4-9 给出打开 XML 文档的结果。

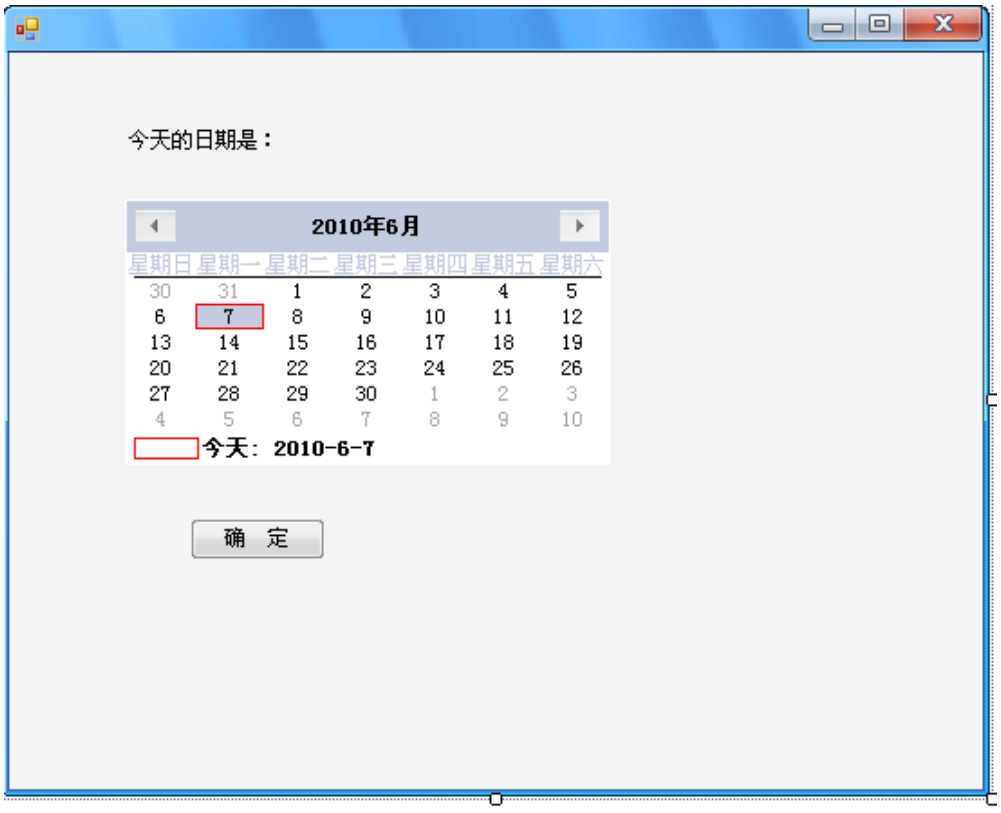


图 4-9　XML 文档打开结果

图 4-10 给出得出的 xml 文档。

```
- <Object type="System.Windows.Forms.Form, System.Windows.Forms,
    Version=2.0.0.0, Culture=neutral, PublicKeyToken=b77a5c561934e089"
    name="Form1" children="Controls">
  - <Object type="System.Windows.Forms.Button,
      System.Windows.Forms, Version=2.0.0.0, Culture=neutral,
      PublicKeyToken=b77a5c561934e089" name="Button1"
      children="Controls">
      <Property name="TabIndex">2</Property>
      <Property name="Name">Button1</Property>
      <Property name="Size">75, 23</Property>
      <Property name="UseVisualStyleBackColor">True</Property>
      <Property name="Text">确  定</Property>
      <Property name="Location">100, 254</Property>
    - <Property name="DataBindings">
        <Property
          name="DefaultDataSourceUpdateMode">OnValidation</Property>
      </Property>
    </Object>
  - <Object type="System.Windows.Forms.MonthCalendar,
      System.Windows.Forms, Version=2.0.0.0, Culture=neutral,
      PublicKeyToken=b77a5c561934e089" name="MonthCalendar1"
      children="Controls">
    - <Property name="DataBindings">
        <Property
          name="DefaultDataSourceUpdateMode">OnValidation</Property>
      </Property>
      <Property name="Name">MonthCalendar1</Property>
      <Property name="Location">64, 80</Property>
      <Property name="TabIndex">1</Property>
    </Object>
  - <Object type="System.Windows.Forms.Label, System.Windows.Forms,
      Version=2.0.0.0, Culture=neutral,
      PublicKeyToken=b77a5c561934e089" name="Label1"
      children="Controls">
      <Property name="TabIndex">0</Property>
      <Property name="Size">100, 23</Property>
      <Property name="Text">今天的日期是：</Property>
      <Property name="Location">64, 42</Property>
    - <Property name="DataBindings">
        <Property
          name="DefaultDataSourceUpdateMode">OnValidation</Property>
      </Property>
      <Property name="Name">Label1</Property>
    </Object>
    <Property name="Name">Form1</Property>
  - <Property name="DataBindings">
      <Property
```

图 4-10　保存的 xml 代码

在这些文件按钮的相关的操作中有的是表面上的界面操作，而另一些则是后台的逻辑操作，这些操作都是作为一个图形界面编辑系统所必需的，也是用户需要掌握的操作。

4.2 编辑按钮

在编辑按钮下存在着五个操作，分别是剪切、复制、粘贴、删除、全选，这些操作都是在编辑过程中很常见的操作。

在实现这些按钮的过程中使用的是 GlobalInvoke 函数的一些用法，在程序中，直接调用此函数，只需要传入 command 的具体参数，如 Cut、Copy 等等就可以进行相应的操作了。

给出一点操作的过程以说明具体的实现。

```
if (this.CurrentDocumentsHostControl == null)
    return;
IMenuCommandService ims =
    this.CurrentDocumentsHostControl.
        HostSurface.GetService
        (typeof(IMenuCommandService)) as
        IMenuCommandService;
try
{
    switch (text)
    {
        case "&剪切":
          ims.GlobalInvoke(StandardCommands.Cut);
          break;
        case "&复制":
          ims.GlobalInvoke(StandardCommands.Copy);
          break;
        case "&粘贴":
          ims.GlobalInvoke(StandardCommands.Paste);
          break;
        case "&删除":
          ims.GlobalInvoke(StandardCommands.
```

```
                Delete);
            break;
        case "&全选":
            ims.GlobalInvoke(StandardCommands.
                SelectAll);
            break;
        default:
            break;
    }
}
catch
{
    this.OutputWindow.RichTextBox.Text += "操作出现
      如下错误: " + text.Replace("&", "");
}
```

4.3 转换按钮

在这个程序中，有转换成为 XML 代码的需要，因此在转换按钮之下有一个选项“XML”，用这个选项来显示产生的 XML 代码，用户可以通过 XML 代码的形式来查看保存用户自己设计的用户界面，XML 作为当代较为常用的数据保存形式正在越来越被很多专业人士所接受，因此很多的用户都会使用 XML 代码形式进行操作，所以产生了这个转换的必要。

界面上的 XML 操作很简单，只需要点击一下转换下弹出的 XML 按钮即可，但是其后台的转换过程却远不是想象中那么简单。

首先用 XmlDocument 类创建一个实例，先设置一个根结点，其是使用 XmlDocument 中的 CreateElement 方法来进行的，然后检查用户是否进行了拖拽的行为，这里如果有拖拽行为的话前面曾设置过的一个 Container 中的内容的数量会发生变化，这样的话就会看出已经发生了用户的操作，在发现用户拖拽控件后，系统将这些控件看做是一个个节点，使用 XmlNode 中的 AppendChild 方法将这些节点都放到前面的根结点下面，这样下来就会产生一个用户拖拽控件操作产生的一个 xml document。

在产生了 document 以后，需要完成的操作是将 document 中的内容读取出来并显示。在这个过程中，首先用 XmlTextWriter 方法建立一个 XmlTextWriter 实例，这个方法是 XmlTextWriter 类的一个构造函数，然后通过 XmlDocument 类中

的 WriteTo 方法将 document 中的内容写入到 XmlTextWriter 实例中，并将相应的需要 xml 代码的“< >”特定格式的地方进行替换，从而实现产生 xml 代码文档的目的。

读入一个已存在的 xml 文档并显示在标签中同样是程序的功能之一。

程序中主要是建立了一个叫做“ReadFile”的函数来进行文档读入的，函数首先建立一个数据集合，用于读取在 xml 文件中存储的 xml 数据，然后运用系统中的 StreamReader 类中的一个 StreamReader 函数进行创建一个实例，这个函数调用时的参数是读取的文件的路径及最终名称，这些读入的数据保存在自己定义的一个容器中，然后将这些数据通过 XmlDocument 类中的 LoadXml 函数传入到一个新的 xml 文档中，接着从头到尾遍历整个的 xml 节点结构，把每一个节点的信息都读取出来，根据每个节点的 xml 代码，找寻到对应的控件信息，显示在新的标签中。

4.4 工具栏

工具栏的实现主要是用一些 Windows Forms 编程中的一些控件使用和一些函数的调用来完成的，在得到工具栏的工具控件的列表时使用的是 Type 类中的 typeof 方法，获取在 System.Windows.Forms 中的 PropertyGrid、Label、LinkLabel 等等控件，将这些控件按照一定的顺序排列起来，放到一个数组中，在后面将数组中的控件按照顺序再显示到工具栏的控件里面，通过调用 Forms 中的函数，就可以做到将控件放到相应的位置以供使用了。实现拖拽使用的是 System.Windows.Forms 中的 Control 类里面的 DoDragDrop 函数，传入的参数是要拖动的数据，以及允许拖动的产生的结果，本程序主要的结果是复制，也就是将工具栏中的工具控件复制到目的地，从而达到拖拽工具到用户所需位置的目的，此函数返回的就是拖拽产生的最终结果。

第五章　　结论

5.1　总结

图形用户界面是计算机系统的重要组成部分，它直接的关系到计算机系统的可使用性和效率。当前计算机技术的发展使得软件的开发已经越来越向着界面和功能的分开编写发展，用户界面的生成独立于应用功能的开发，这就使得具有图形用户界面开发功能的工具显得越来越有必要了。本文中所论述的图形用户界面开发工具正是这样的一种允许用户根据自己的要求对界面进行排版设计的工具。

该工具使得用户可以简单的使用鼠标的拖拽来将工具控件放到面板上并进行布局排列，这样的操作比较的容易掌握，这样很多对计算机知识并不是十分精通的用户也可以在短时间内完成对操作的学习，其使用的普遍性和灵活性都是很高的。与此同时，它还可以满足很多的用户将界面转换为 xml 文件的需求，可以将 xml 文档转换为界面的形式，这样的功能使得其对使用者的吸引也有了大幅度的提升。

5.2　展望

当前，很多的功能强大、设计成熟的图形用户界面生成工具不断出现在计算机用户的眼前，这些工具能够方便快速的生成理想的图形用户界面，这样的功能为计算机用户带来了极大的便利，也减少了很多程序员的工作，他们可以将更多的精力放在功能的编写上。

相信在将来，会有功能更为强大、设计更为成熟的工具出现在用户的面前，通过这些工具应用程序的开发也必将走向一个新的效率更高的阶段。

参考文献

[1]米切尔森．C# Primer Plus[M]．北京：人民邮电出版社，2002.7．124-210．

[2]Anders Hejlsberg．Scott Wiltamuth．Peter Golde．C#编程语言详解[M]．北京：电子工业出版社，2004.9．26-64．

[3]邓苏，张维明，黄宏斌．决策支持系统[M]．北京：电子工业出版社，2009.1．59-74．

[4]李东，蔡剑．决策支持系统与知识管理系统[M]．北京：中国人民大学出版社，2005.9．78-102．

[5]Shneiderman,B．用户界面设计：有效的人机交互策略（第4版）[M]．北京：电子工业出版社，2006.1．65-98．

[6]孟祥旭．人机交互技术-原理与应用[M]．北京：清华大学出版社，2004.9．65-112．

[7]David Richard，Kalkstein DeLoveh，William Sempf．Visual Studio .NET 高效编程[M]．北京：清华大学出版社，2002.11．135-148．

[8]王华杰，张帆，戴伯勇．Visual Studio.NET 程序设计教程[M]．北京：中国铁道出版社，2003.7．59-79．

[9]Nonogaki，Hajime and Ueda．A construction of the twenty-first centry human interface，Proc[M]．New York：CHI' 91 Human Factors in Computer Systems，ACM，1991．103-142．

[10]Coombs,M.J. and Alty,J.L.．Computing Skills and the User Interface[M]，New York：Academic Press,1981．105-117．

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

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

[13]R.J.Torres.Practitioner．Handbook for User Interface Design and Development[M]．Pearson Education，2002．98-121．

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

[15]李垠．软件界面设计的基本原则[EB/OL]．http://blackmagnet.blog.sohu.com/67803113.Html,2007 ．

[16]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.

[17]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．

致　　谢

值此论文结稿之际，谨向所有给予我帮助的老师、同学、亲人和朋友表示衷心的感谢！

最要表示我诚挚谢意的是我的导师张坤龙老师，从一开始的选题到确定题目，从设计到实现，从程序的编写到论文的写作，这些都是在张老师的细心指导和热心帮助下完成的，很多时候张老师都牺牲休息时间不辞辛劳的为我说明，为我指出自己在哪里还需要大的改进，张老师的丰富的专业知识和严谨的治学态度深深激励了我，张老师的一丝不苟的敬业精神深深感染了我，不仅仅是在学业上，在为人处世和对待事务的态度上都使我受益匪浅，在论文的写作中张老师同样给了我很多宝贵的意见和指导，在这里再次对张老师说一声感谢。

同时要感谢我的同学们，尤其是我的宿舍的各位舍友们，在毕设的过程中，他们给了我很大的帮助，有些是在程序方面，有些是在生活方面，他们对我的帮助使得我按时完成了程序和论文的编写，感谢他们。

还有学院的领导和老师们，为我们的毕设尽可能创造出方便的条件，创造出尽可能舒适的环境，感谢各位领导老师！

最后要将一份特别的感谢献给我的家人，不仅仅是在毕设的这一段时间中，在我的学习生涯中他们都是我坚强的后盾，精神上默默支持，生活上无微不至的关怀，这些都是支持我向前的不竭动力，我唯有加倍努力，用实际行动回报家人对我的关怀和期望。

再次对所有帮助我的人给予我最诚挚的谢意！