系统概述

1. 解决方案及设计思想
2. 实现方式概述

本系统基于Microsoft提供的MFC框架开发，并采用多文档(MDI)结构实现。

多文档(MDI)结构是一种可实现具有多子窗口(ChildFrame)的系统的结构。在本系统中，多子窗口可以用来与“多层”的数据流图相对应，这为系统最终实现“一个窗口对应于一层的数据流图”的效果提供了基本的可能性和保障。所以，采用多文档(MDI)结构开发本系统，可以满足数据流图本身的“多层”需求。

其次，MFC框架提供的文档视图模型可以很好地满足“可视化”这一需求。在MFC中，文档类Document结构负责存储窗口对应图形的数据，包括图元的坐标、类型、名称等；视图类View负责根据文档类中的数据，绘制出相应的图形及其中包含的图元。且在多文档(MDI)结构中，每一个子窗口均持有一个文档成员和一个视图成员，及子窗口、文档、视图之间是一一对应的关系。这就可以确定，在本系统在开发过程中，使用MFC框架开发可以非常有针对性地实现“一个窗口对应于一层的数据流图”这一关键需求。

为了支持以上实现方式，本系统采用集成开发环境Microsoft Visual Studio 2013开发， C++语言编写，并在小组合作中使用软件Github进行版本控制。

1. 类设计思想概述

一个好的多层数据流图可视化编辑系统应支持多张数据流图的创建、编辑、及管理等操作，而不是仅支持一张数据流图的以上操作。所以，本系统不仅需要将每张数据流图抽象为一个图形类Diagram，并且需要一个图形管理类（器）以管理这些图形，即DiagramEditor。

再次，由于一张数据流图由多个图元组成，每种图元有共同的特点，如图元坐标、名称等，也同时都具有各自不同的特点，如形状、操作等，互不相同。这就使得系统在开发时必须考虑到这一点，并为每种图元设计自身对应的类，且这些类均继承自一个共同的基类——图元类Element，以便于区分它们的相同点及不同点，且能够统一管理。具体的类间关系设计及接口设计将在“静态模型”部分说明。

除此之外，考虑到多种图元整体数量较多的相同操作及不同操作，为减轻图形管理类（器）DiagramEditor响应用户操作的压力，分离图元对象的使用和创建，提高抽象层次，本系统在开发过程中采用了工厂(Factory)模式，创建了控制图元操作的工具类Tool。并且，针对不同图元的不同创建过程、编辑过程、查找过程，工具类Tool作为基类派生出了多个派生类Tool，如RectangleTool、StreamTool、EditTool及LookupTool等。这些派生类分别控制于数据流图图元源(Source)的创建操作、数据流图图元流(Stream)的创建操作、图元的编辑操作及数据流路径的查询操作等。具体的类间关系设计及接口设计将在“静态模型”部分说明。

1. 系统框架概述
2. 窗口主框架MainFrame与图形管理类（器）DiagramEditor

在MFC中，一个主窗口(MainFrame)可包含多个MDI子窗口(ChildFrame)。一个MDI子窗口对应于一个文档类(CDocument)及一个视图类(CView)。也就是说，一个子窗口对应一个图形，一个视图对象对应一个图形，一个文档对象对应一个图形。作为图形管理类（器），由于DiagramEditor需要支持对多个图形的管理操作，所以它必须能够像一个主窗口管理多个MDI子窗口一样管理多个图形。由此我们想到，DiagramEditor与MainFrame的层次及特点都是类似的。所以，在本系统中，我们将整个系统层次最高的图形管理类（器）DiagramEditor的对象，作为主窗口MainFrame的一个成员。通过MainFrame与DiagramEditor的一对一关联，我们将MFC框架与本系统联系起来。

1. 文档视图与图形间的关联方式

在本系统中，为了使主窗口下的各MDI子窗口在每次刷新（包括用户手动点击标题栏刷新、最小化最大化刷新、拖动子窗口时各子窗口的刷新等）的时候能够正确地绘制出其自身对应的图形，我们在每个子窗口各自对应的文档类CDocument中存入了其自身对应的图形指针d(Diagram \*)。在每次子窗口刷新时，每个子窗口各自的视图类CView将通过GetDocument()方法，得到相应文档类CDocument中存储的对应图形，根据此图形进行绘图工作。一个概括性的绘图过程状态转换图如图 1所示。



图 1绘图过程状态转换图

1. 工厂模式的使用