系统概要设计说明书

目录

[1.需求说明 4](#_Toc437430275)

[1.1 需求概述 4](#_Toc437430276)

[1.1.1 软件平台部分开发目的 4](#_Toc437430277)

[1.1.2 关键字提取模块开发目的 4](#_Toc437430278)

[1.1.3 目标用户群 4](#_Toc437430279)

[1.1.4 软件定位 4](#_Toc437430280)

[1.2 功能列表 4](#_Toc437430281)

[1.2.1网络生成 4](#_Toc437430282)

[1.2.2 网络读取与保存 4](#_Toc437430283)

[1.2.3 网络显示 4](#_Toc437430284)

[1.2.4 各种参数计算功能 5](#_Toc437430285)

[1.2.5 系统功能 5](#_Toc437430286)

[1.2.6 关键词划分模块 5](#_Toc437430287)

[1.3 任务优先级 5](#_Toc437430288)

[1.3.1 P1：底层框架搭建 5](#_Toc437430289)

[1.3.2 P2：网络构建功能 5](#_Toc437430290)

[1.3.3 P3：网络结构图绘制 5](#_Toc437430291)

[1.3.4 P4：网络参数计算 5](#_Toc437430292)

[1.3.5 P5：其他外围功能 6](#_Toc437430293)

[1.4 其他约束 6](#_Toc437430294)

[2.总体设计 6](#_Toc437430295)

[2.1 构建与运行环境 6](#_Toc437430296)

[2.2 总体设计原则 6](#_Toc437430297)

[2.3模块划分 6](#_Toc437430298)

[2.4 用例设计 7](#_Toc437430299)

[3.功能分析 7](#_Toc437430300)

[3.1 网络创建 7](#_Toc437430301)

[3.1.1 流程描述 7](#_Toc437430302)

[3.1.2 UML序列图 8](#_Toc437430303)

[3.2 导入与导出 8](#_Toc437430304)

[3.2.1 网络文件导入流程描述 8](#_Toc437430305)

[3.2.2 UML序列图 9](#_Toc437430306)

[3.2.2 网络文件导出流程描述 9](#_Toc437430307)

[3.2.3 UML序列图 10](#_Toc437430308)

[3.3关键词网络构建 10](#_Toc437430309)

[3.3.1 流程描述 10](#_Toc437430310)

[3.3.2 UML序列图 10](#_Toc437430311)

[3.4 关键词提取 10](#_Toc437430312)

[3.4.1 流程描述 10](#_Toc437430313)

[3.4.2 UML序列图 10](#_Toc437430314)

[3.5 用户设置 10](#_Toc437430315)

[3.5.1 流程描述 10](#_Toc437430316)

[3.5.2 UML序列图 11](#_Toc437430317)

[4. 数据结构设计 11](#_Toc437430318)

[4.1 主要数据结构设计 11](#_Toc437430319)

[4.1.1 连边 11](#_Toc437430320)

[4.1.2 节点 12](#_Toc437430321)

[4.1.3 网络 12](#_Toc437430322)

[4.2 输入输出文件 13](#_Toc437430323)

[4.2.1 XML交换文件 13](#_Toc437430324)

[4.2.2 其他输入输出文件 14](#_Toc437430325)

[4.3 数据库设计 16](#_Toc437430326)

[4.3.1 E-R关系 16](#_Toc437430327)

[4.3.2 数据表设计 16](#_Toc437430328)

[5.软件架构设计 16](#_Toc437430329)

[5.1 软件架构概要 16](#_Toc437430330)

[5.2 系统内核设计 18](#_Toc437430331)

[5.2.1 内核架构说明 18](#_Toc437430332)

[5.2.2 内核架构问题与演化 18](#_Toc437430333)

[5.2.3 内核架构设计 19](#_Toc437430334)

[5.3 网络创建子系统 21](#_Toc437430335)

[5.4 文件读写子系统 22](#_Toc437430336)

[5.5可视化子系统 22](#_Toc437430337)

[5.6 关键词提取子系统 23](#_Toc437430338)

[5.7 用户设置子系统 24](#_Toc437430339)

[5.8 故障处理模块设计 24](#_Toc437430340)

[5.9架构设计总结 25](#_Toc437430341)

[6.概要设计评审 27](#_Toc437430342)

# 1.需求说明

## 1.1 需求概述

### 1.1.1 软件平台部分开发目的

1. Matlab工具缺乏专业性，以及其代码的低可读性，低可维护性，低可复用性，低可用性。

2.一般用户缺乏计算机基础知识（尤其是Matlab程序编写），而无法使用计算机技术来研究大规模网络。

### 1.1.2 关键字提取模块开发目的

实现对海量文本信息的文本关键词的自动提取，减少人工标注所需的大量人力物力，为自然语言处理的相关研究提供基础。

### 1.1.3 目标用户群

从事自然语言处理研究（自动问答系统、自动摘要等）的相关人员或搜索引擎网站建设人员。

### 1.1.4 软件定位

复杂网络仿真平台属于小规模软件，且设计思想为“精确定位，客户定制”。软件产品族中每个版本都为不同客户专门裁剪定制，去除一切不需要的功能，做到规模小型化。

## 1.2 功能列表

### 1.2.1网络生成

1.支持生成BA无标度网络，ER随机图，小世界网络（WS，NW）；

### 1.2.2 网络读取与保存

1.能够读取.mat，.xml，sst和三元组格式的文件；

2. 能够将网络保存为.mat，.xml，sst和三元组格式的文件；

3. 网络读入后能够识别有向网络和加权网络；

### 1.2.3 网络显示

1. 支持网络可视化绘制（有向，加权）；

2. 支持网络参数，节点参数，连边参数显示；

3. 支持网络节点和连边可被高亮选中，节点可以移动；再次点中则高亮取消，节点锁定；

4. 支持参数绘制曲线图（对数分布），曲线图可以关闭；

5. 所有图形均可以保存为本地图片，支持bmp，jpg，emf，png， svg ，gif ，tif等格式；

### 1.2.4 各种参数计算功能

1.网络参数（平均度，平均距离，网络密度，网络直径）；

2.节点参数（聚类系数，环路系数，接近中心系数，介数，pagerank值，k-shell值，最短距离）；

### 1.2.5 系统功能

1. 网络显示中节点颜色可以由用户设置与修改；

2. 网络中节点的形状由用户设置与修改；

3. 网络显示密度和节点大小可以由用户设置与修改；

### 1.2.6 关键词划分模块

1. 参数计算：网络参数计算、节点参数计算

2. 文本网络构建：基础网络构建

3. 文本网络优化：词语相似度计算

4. 文本关键词提取

## 1.3 任务优先级

### 1.3.1 P1：底层框架搭建

1. 网络承载能力，构建基本数据结构存放网络数据；

### 1.3.2 P2：网络构建功能

1. 网络文件读取与保存；

2. 网络自定义生成器；

### 1.3.3 P3：网络结构图绘制

1. 软件数据库搭建，确定存储结构；

2. 网络结构图绘制和显示功能；

### 1.3.4 P4：网络参数计算

1. 网络参数，节点参数计算；

2. 网络参数，节点参数的显示功能；

### 1.3.5 P5：其他外围功能

1. 高级用户体验功能；

2. 代码性能优化；

# 2.总体设计

## 2.1 构建与运行环境

运行操作系统，Windows 7及以上版本, 对于早期版本需要安装.Net Framework 4.0 组件。

开发环境：Visual Studio 2010

开发语言：C#

## 2.2 总体设计原则

1、 架构应该尽可能简单，采用相同的原则和方法解决一类相近的问题。

2、开-闭原则：对扩展开放，对修改关闭。由于系统是为复杂网络领域各个研究方向开发的实验工具软件。必须具有较强的扩展能力，所设计的平台需要满足各种不同的需求。要做到在原有平台基础上新增模块，而不是通过修改原有平台代码来新增功能。

3、YAGNI原则，如果不是马上需要，就不要去做。延迟设计决定，直到想好何时放入系统中。

4、不要害怕改变，没有什么是一成不变的。

5、面向接口编程，而不是面向实现。

## 2.3模块划分

本软件是由底层平台和上层应用共同组成的，其中底层平台负责基本IO，网络构建，界面显示等基本功能，而上层应用则提供关键词提取功能。具体模块划分如下图所示。

其中上部分分为Platform和Keyword两部分，他们各自完成自己的工作，只有在网络构建时才会产生数据交互。而他们对网络的基本操作都来自中间Interface层所提供的函数。

接口层的作用是将系统底层和上层分割开来。尽可能保持底层的封装性，同时给予上层应用足够的操作自由。

Core部分是软件的内核，提供一系列调用和接口给上层应用。该部分功能呢是软件的核心数据存放和处理，即对抽象的复杂网络实体进行操作。



图2.1 复杂网络仿真平台模块划分

# 3.功能分析

## 3.1 网络创建

### 3.1.1 流程描述

1、主窗体菜单中选择“文件”->“新建”，弹出新建网络对话框。如果当前存在一个使用中的网络，则提示用户保存；

2、在弹出的“新建网络”对话框中，用户首先选择自己需要创建的网络类型。随后系统会给出所需类型对应的构建参数，用户需要一一输入。系统对输入结果进行校验后，调用创建函数。

3、创建函数根据用户输入参数的不同，生成不同的创建算法对象，并使用用户参数创建对象实例。

4、创建算法对象根据算法生成网络，随后完成网络所需的初始化操作。

5、最后将网络实例返回给主窗体的ComplexNet对象。

6、主窗体中函数负责将网络绘制到主窗体绘图框中，并将网络参数导入各个参数显示框。

### 3.1.2 UML序列图



## 3.2 导入与导出

### 3.2.1 网络文件导入流程描述

1、用户点击主窗体菜单“文件”->”打开”，选中待导入的网络文件，单击确定导入。

2、主窗体判断当前是否有在使用中的网络，如果有提示用户保存。

3、主窗体创建网络实例，但暂不加入节点；最后调用网络Read函数，传入文件路径。

4、网络根据文件后缀名，初始化算法实例。

5、网络调用算法实例函数，从文件中读取数据，构成标准化XML数据。

6、算法实例返回XML数据，网络解析XML数据，生成节点和连边，并加入网络。

7、网络完成后续统计和处理工作，将自己返回给主调窗口。

8、主窗口得到生成完毕的网络后，在绘图窗口中绘制网络结构图。

### 3.2.2 UML序列图



### 3.2.2 网络文件导出流程描述

1、用户选中主窗体菜单“文件”->“保存”，在弹出的保存对话框中选择保存目录，并确定文件名和文件类型。

2、主窗体调用CNet的Wirte函数，传入待保存的完整文件名。

3、Cnet首先根据文件后缀名，生成对应的算法对象。

4、CNet将自己的XML核心数据传递给保存算法对象。

5、算法对象保存数据到文件，并返回结果。

### 3.2.3 UML序列图



## 3.3关键词网络构建

### 3.3.1 流程描述

输入数据：目标文本，停用词列表，无用词性列表

Step 1：分词，利用分词软件输出“词语+词性”组合列表，存入List<string> WordRes；

Step 2：分句，根据标点符号将目标文本分成句子，存入List<string> Sentence；

Step 3：构建节点，输入词语内容strWord，词性intType，根据Sentence确定词语位置intPositon，和所属句子索引intLineNum，构造节点存入List<kNode> NodeList中。

Step 4：扫描节点，如果命中停用词表和无用词性表的节点索引赋予-1。随后合并同义词节点，被兼并节点索引赋予-1，其kNode kOwner 引用指向兼并它的节点。最后，从前到后对所以不是-1的节点依次赋予索引，本索引为永久索引；

Step 5：根据连边规则，依据位置intPosition和所属句子索引intLineNum进行加边，对于被兼并节点，将边加载到其宗主节点上;

Step 6：新建关键词网络kNet，从 List<kNode> NodeList中取出索引>-1的节点，加入网络，完成后续处理，结束网络构建。

### 3.3.2 UML序列图

## 3.4 关键词提取

### 3.4.1 流程描述

TBC

### 3.4.2 UML序列图

## 3.5 用户设置

### 3.5.1 流程描述

1、用户选择主窗体中菜单“工具”->“设置”，系统弹出用户设置窗体。

2、用户设置窗体从数据库中读取当前的系统配置，并以“所见即所得”的方式呈现在窗体中。

3、用户调整配置窗体中的有关设置项，配置窗体不断刷新预览。

4、用户取消配置结果，则丢弃所有配置。用户选择应用和确定则将设置保存。

5、用户设置窗体将数据存入数据库，并通知主窗体根据当前新配置，刷新所有依赖元素。

### 3.5.2 UML序列图



# 4. 数据结构设计

## 4.1 主要数据结构设计

### 4.1.1 连边

由于需要支持有向图和加权图，连边中首先需要存放权重，其次也需要指明连边的起点和终点。因此连边类Edge主要结构如下：

Class Edge

{

int intSource; //起始节点

int intTarget; //目标节点

int intValue; //权重

}

### 4.1.2 节点

节点是组成网络的基本元素，每个节点必须要有自己独立的编号，随时可以用来索引。另外需要一组数据结构来存放连边，而连边也需要能随时通过目的节点的编号来索引。最后为了能在图像中绘制该节点，节点的中心点位置也需要保存。

本项目中，我们使用字典类Dictionary来存放节点中的连边，Dictionary类分为两部分，Key和Value，Key是用来查找输入的索引值，Value是返回值。二者分别对应着目的节点的编号和连边实例。也就是说，我们通过输入目的节点的编号，就能从Links中获取从当前节点开始，到目的节点终止的连边的Edge类实例。

Node类的基本数据结构如下所示：

Class Node

{

int intNumber; //节点编号

int intDegree; //节点度

Dictionary<int, Edge> Links; //连边列表

Point potLoc; //节点位置

}

### 4.1.3 网络

网络是提供各种复杂网络操作的基本对象，是上层应用直接操作的数据结构。由于复杂网络仿真平台面向单一平台加多应用方向的设计理念。就需要支持多种不同的上层应用功能。但是如果将各个网络研究方向的功能全都堆放在同一个网络类中，势必造成该类积重难返。

这里我们采用一个通用平台网络加多个应用网络的结构来解决这一个问题。其中通用平台网络负责文件IO，网络创建和基本参数计算等各个领域都需要用到的功能。而应用网络则负责各个领域不同的网络特性和功能。但有一点是相同的，就是作为复杂网络的基本网络元素——节点的组成方法。

如下所示就是每个网络，包括平台和应用网络都应该包含的数据结构。

class Net

{

int intNumber; //节点数量

List<Node> Network; //节点列表

}

主要包括的内容就是一个节点列表和节点数，这里节点列表采用List泛型容器来存放Node类型节点实例是利用了List容器方便删除、插入和索引的优点。由于每种网络都具有这种基本结构，那么从平台网络转化为应用网络就非常简单了。只需要由平台网络将基本结构通过XML输出，并由应用网络来读入即可生成各不相同的上层应用网络。

## 4.2 输入输出文件

### 4.2.1 XML交换文件

如前所述，我们使用XML文件作为系统内部数据交换的基本格式。从平台网络生成应用网络的时候需要使用XML文件对象进行数据传递。从普通文件读入网络的时候也是先生成XML对象，在交由网络类进行XML对象解析和网络构建。可以说XML就是整个复杂网络仿真平台中流动的血液。

XML文件所需要存储的数据结构非常简单，就是前述的网络基本元素。从下图中可以看出，网络的根节点是Network，其中包括两个属性节点数和连边数。

根节点包括若干个子节点，每个子节点都代表复杂网络中的一个“节点”对象。展开子节点可以发现，它由节点度Degree，节点位置Xpos，Ypos和连边列表Edges组成。其中连边列表的长度和节点度是对应的。

最里层是Edge对象的数据结构，每一个Edge由表示目标节点的属性Target和权重的文本值Value组成。整个XML文档就存放着构成网络所需的最基本元素。

同时为了标准化和校验的需要，我们使用了XSD文件生成了文件结构的标准描述，系统可以直接调用来对用户提供的XML网络文件进行校验。

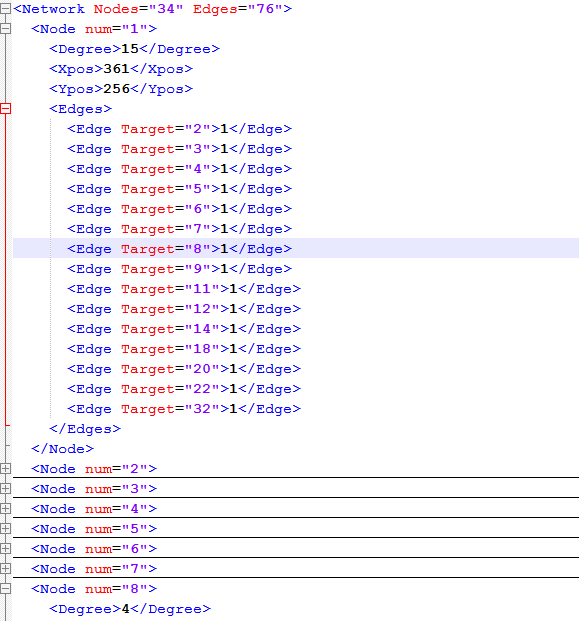


图4.1 XML文件格式示例

XSD文件 TBD

### 4.2.2 其他输入输出文件

由于系统前向兼容和第三方软件兼容的需要，我们同时还支持另外三种文件的输入和输出。

1. sst文件

sst文件是第一代复杂网络仿真平台的标准网络文件，但其存在一些不合理不方便的地方。在本系统中将仍然提供对该种网络文件的支持。

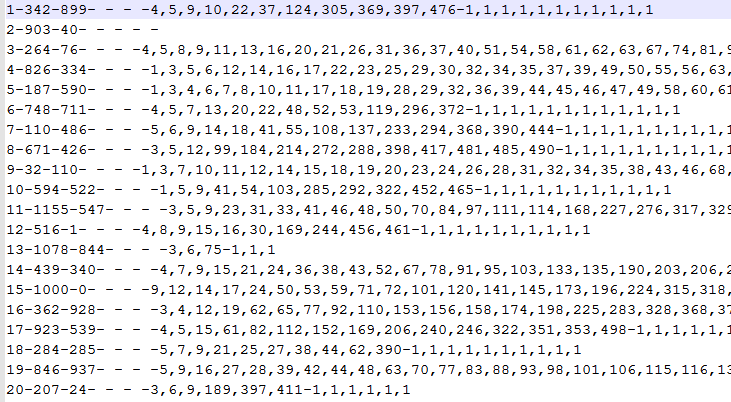


图4.3 sst文件格式示例

2. mat文件

mat文件是MATLAB的网络存储类型，主要存放的是复杂网络的邻接矩阵，它是一种广泛采用的数据存储文件。我们使用matlab中提供的文件结构说明，也同时实现了mat文件的读写。

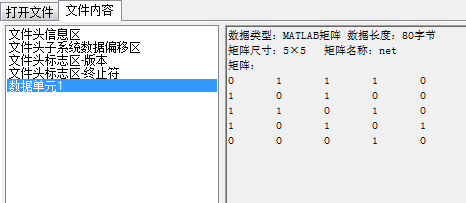


图4.4 mat文件格式示例

3. tri文件

tri文件格式即三元组格式，是一种在网络上经常使用的文件格式。通常是以文本文档的格式保存，但这里为了识别方便，统一标记为.tri后缀名的文件。如下图所示，整个文档由若干个三元组组成，每个三元组表示一条边的起点，终点，权重。

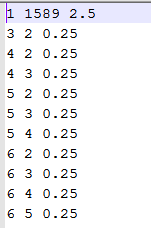


图4.5 tri文件格式示例

## 4.3 数据库设计

复杂网络仿真平台是面向实验研究的算法密集型系统，而非面向业务的数据密集型系统。所以系统数据库只是为了存放一些系统数据，进行统一管理。一般不涉及用户数据的存储，这样数据库内部的表也都独立存在，没有相互间的E-R关系。

### 4.3.1 用户样式表tabStyleSet



图 4.6 tabStyleSet表结构与字段属性

用户样式表用于存放复杂网络结构图绘制时所需要的各种参数，如前景色，背景色，字体大小等。系统会提供一系列预设样式供用户选择。当然，用户也可以创建自己的样式集，并通过配置界面，保存在数据库中，以备下次使用。

### 4.3.2 词性表tabWordType

TBC

### 4.3.3 停用词表 tabObsolete

TBC

# 5.软件架构设计

## 5.1 软件架构概要

复杂网络仿真平台的主要目的是将复杂网络的数据读入生成可视化的网络结构图，并在此基础上展开对网络的各种研究实验。如图5.1所示，用户输入网络文件，由平台读取生成网络，也可以在平台中输入参数直接生成网络。平台依据系统配置数据库中的数据对生成的网络进行配置并可视化地呈现给用户。

用户在现在有网络上，可以选择各种不同的应用功能对当前的复杂网络进行实验，实验完成后，平台会以图片或文本的方式提供实验结果，并方便用户导出保存。



图5.1 复杂网络仿真平台交互关系图

复杂网络仿真平台具有以下特点：

1）单一平台，以当前活动网络作为所有操作的核心。一切网络文件都会由平台读入并构建成网络实例，所生成的网络也都通过平台保存为文件。平台模块负责整个系统的文件读写，网络创建，可视化和基本参数计算功能。这些功能可以为大多数网络研究提供基础和重要参考；

2）多应用和可裁剪性。对于各种研究方向的特殊要求，设计开发不同的上层应用模块。他们拥有自己的目标网络，但这些网络的基本结构数据都来自平台模块。可以说是平台网络进行扩展后生成的特殊网络。由于这些应用只依赖于平台模块，不存在横向联系，它们可以根据用户的需求进行裁剪，做到软件的小型化和专业化，同时用户只需要为自己选购的模块付费即可。

3）算法密集型系统。系统本身的工作步骤并不复杂，但是对于底层，需要支持多种文件格式的输入和输出。对于上层应用，需要支持多种复杂网络研究方向的实验算法。对于复杂网络仿真平台而言，需要支持大量的算法，包括算法的扩展将成为设计的主要方向。

4）可视化。作为本软件的主要特性，可视化可以帮助研究者直观地从图形中观察网络的研究结果。在本软件中，从网络结构图到中间的仿真实验过程、仿真结果都可以以图片，图表等方式可视化地输出。

## 5.2 系统内核设计

### 5.2.1 内核架构说明

如第四节数据结构中所述，系统的主要数据结构是网络Net类，而在其内部又由若干节点Node和连边Edge构成。从宏观上来看，系统内核是由Net->Node->Edge三层结构组成的，大多数的操作和访问也是层层调用共同实现。

class Net

{

int intNode;

int intEdge;

List<Node> Network;

}

class Node

{

int intNumber;

int intDegree;

Dictionary<int, Edge> Links;

}

class Edge

{

int intSource;

int intValue;

int intTarget;

}

### 5.2.2 内核架构问题与演化

虽然系统架构的基本结构如前所述，但是在具体设计时会遇到扩展性的问题。在一个类中直接引用另一个类的实例作为成员，会产生可扩展性不足的问题。例如，在本系统中关键词节点和一般网络节点不同，还需要增加词性、位置、词语内容等属性。如果将他们都加入当前Node类会导致Node类的膨胀。如果平台还需要支持其他不同应用，将这些应用所使用的特殊节点属性都加入Node类，会引发极大的问题。

第一次演化，Node类扩展，如图5.2所示。将Node类作为所有节点类型的基类，通过继承产生用于不同用途的节点cNode，dNode和kNode，他们分别完成不同的功能。同时在上层会有不同的网络类（cNet，kNet和DNet）作为承载这些节点的功能网络。这些网络通过调用各自节点的成员函数来完成网络整体的功能。



图5.2 内核第一次演化架构图

回到之前的系统设计概要，我们需要一个统一的平台和丰富多样的应用。这里通过继承产生了丰富多样的节点，完美地解决了丰富多样应用的问题。但统一的平台这一点被忽视了，也就是说上述的这些节点只能完成自己特定的功能，而不能拥有共同的功能。

比如cNet是平台使用的网络，能够读取文件，计算基本参数。而kNet只是关键词网络，能够提取关键词。但希望kNet能否计算基本参数的话就要在kNet中重新实现cNet的部分方法，并在上层应用中重新构建基本参数的计算和显示界面。这是软件开发者不希望发现的，一旦算法出现问题，不得不多处修改。系统功能越丰富，潜在的风险越难以控制。

究其根源，网络类和节点类一一对应的思维定势是问题的关键。cNet中只能包含cNode，kNet中只能包含kNode的这种类与类之间的强耦合，使得网络之间的功能不能共享。试想，如果关键词节点kNode能够存在于cNet和kNet中，不就可以同时使用cNet的参数计算功能和kNet的关键词提取功能了吗？但cNet中原本的cNode怎么办呢？

### 5.2.3 内核架构设计

对于上一节提到的问题，C#的接口提供了一个很好的解决方案。通过接口机制，可以将网络类和节点类进行解耦合，具体的方法就是在两者之间插入一个接口层。如图5.3所示，原本的网络层基本不变，但是构成网络的节点集合的类型从原来的cNode和kNode变成了IfPlatform和IfKeyWord。而cNode和kNode都可以实现接口IfPlatform的功能，这样就使得cNet类中既可以使用cNode节点也可使用kNode节点。而区分这两者，只需要根据读入文件的类型就可以了。

图5.3 架构第二次演化结果uml类图

同时kNode节点也实现了接口IfKeyWord的功能，这使得它可以在关键词网络中被使用。使用接口技术，可以保证以后如果需要使用其他的任何节点，只要继承所适配的网络接口，就能在对应的网络中使用。使得节点类和网络类之间的绑定关系得到了解耦合。

重新研究节点类kNode可以发现，kNode作为应用网络的节点同时又需要拥有平台的功能，其结构可以分为图5.4所展示的四部分。



图5.4 节点的一般性结构

首先是来自基类Node提供的各种变量，属性和方法，属于共享特性。其次是其根据自身功能所扩展的特性，属于特殊特性。第三是实现平台接口的各种方法，这使得它可以被平台网络cNet调用。最后是实现应用接口的方法，这些方法是关键词提取所单独使用到的方法。类似的，如果想给kNode增加功能，只要让其实现其他的功能接口，并在对应的网络中将它作为节点组件即可。

## 5.3 网络创建子系统

网络创建需要支持ER随机图，WS小世界网络，NW小世界网络和BA无标度网络等四种网络的生成。在未来，不排除需要扩展更多的生成算法。假设每个生成算法都需要三个方法，那么类中所包含的方法将不断激增，使得管理成本不断上升。

通过仔细分析各个生成算法可以发现，他们基本上都是由网络构建，网络初始化和节点放置三个步骤组成的。作为基本流程不变的情况，这里考虑使用strategy模式。使用接口将所有生成算法所共同需要的函数提取出来，构成Construct()，Initilized()，PlaceNodes()三个接口函数。而每个Strategy将各自实现这三个接口方法。对于cNet类而言，只要根据用户选择的网络类型创建对应的算法实例，随后直接调用接口里的方法来创建网络就可以了。具体的uml类结构图如图5.5所示。



图5.5 网络创建子系统uml类图

## 5.4 文件读写子系统

与网络创建子系统类似，文件读写也需要支持多种不同的文件格式算法。那么使用Strategy模式也将给设计使用带来便利。如图5.6所示，cNet包含一个IO对象IPhandle，它在文件读入时确定具体是由哪个类来组成。可以是以下三种算法中的任意一种。无论如何，他们都可以实现IfIOStrategy所要求的接口方法。

图5.6 文件IO系统uml类图

## 5.5可视化子系统

算法多样的问题同样体现在绘图系统中，系统中存在默认的算法，也包含一系列供用户选择的预置算法。接口对象NetPainter就是为了解决多样性的方法而存在的。但与之前的创建和文件读写不同之处在于，绘图算法只能通过用户配置数据库中的样式集来实现修改，而不能由用户输入文件来确定。因此在图5.7的下半部分可以看到，这两个具体的算法类都包含StyleSet对象，他们是初始化的时候从系统数据库中读出的配置样式集。根据用户的配置，来决定具体生成哪种算法类实例。

另外，由于绘图的参数比较多，还专门设置了PaintParameter类来负责参数的传递，保证cNet类的封装性。

图5.7 绘图子系统uml类图

## 5.6 关键词提取子系统

TBD

## 5.7 用户设置子系统

TBD

## 5.8 故障处理模块设计

TBD



图5.8 故障处理子系统uml类图

## 5.9架构设计总结



图5.9 系统uml总结构图

经过之前几节的描述，可以得出如图5.9所示的复杂网络仿真平台总结构图。从上到下由应用层，网络层，接口层，节点层和基本元素层五层组成。图5.10是这五层的分层示意图。



图5.10 分层示意图

应用层：具体实现各个功能的上层窗体函数。平台包含一个默认的ComplexNet对象（Cnet类型），其他网络则由二次开发程序员自行设计。平台支持：网络生成，文件读写，参数计算，网络显示，在主窗体中使用。应用支持：各不相同，例如关键词提取使用kNet网络，从cNet中拷贝构造，在关键词构造窗体中使用。

网络层：由各种功能节点组成的功能网络，如平台的Cnet，关键词提取kNet。但本层更重要的是包含各功能接口实现的算法对象。一个功能网络由适当的功能节点加上数个功能组件共同构成。

接口层：为节点层和网络层提供接口，网络层只看得到接口层，并不知道其具体包含哪种节点。比如kNode同时继承平台接口和关键词接口，平台网络cNet只能使用平台接口中的函数，而关键词网络kNet只能使用关键词接口中的方法。

节点层：存放各种继承自Node的派生节点，如平台使用的Cnode，关键词的kNode等。他们都是Node的派生类，同时为了需要在平台网络中使用的必须继承实现平台接口。另外为了完成应用网络中的功能，也需要继承实现各自不同的应用接口（默认平台节点cNode只继承平台接口，而无应用接口）。

基本元素层：Node基类和Edge类，是组成网络的最小元素，本身不包含附加信息只保存网络文件中最必要的信息。

# 6.概要设计评审