姓名	学号	Github 账号	分工	成绩
高天琦	201592333	preciousky	组长	

开源软件基础课程报告

报 告 题 目　东北三省 AQI 空气质量指数简要分析

项 目 网 址　https://github.com/preciousky/PyOpenSource.github.io

完 成 日 期　2018 01 13

大连理工大学软件学院

目　　录

1　项目初始及软件定义

1.1　问题定义及需求分析

空气质量问题如今已经成为阻碍生活质量提高的一个因素，人们对于环境质量的要求越来越高，尤其是近五六年的雾霾天气更是进入了人们的关注焦点之中，每天出门前先看看今天的 AQI 指数成为越来越多人的习惯。软件主要利用网络上公开的数据，对各地区 AQI 指数进行统计并给出可视化展示，给出空气质量随时间，地理位置的变化规律及变化范围，给人们的生活出行活动以更多参考建议。

软件主要关注东北三省地区的 AQI 指数数据，以小时为数据获取周期，按时间存储 AQI 指数，生成实时更新的 AQI 指数的统计结果，在地图上定性展示空气质量的情况。

1.2　生命周期模型的确定

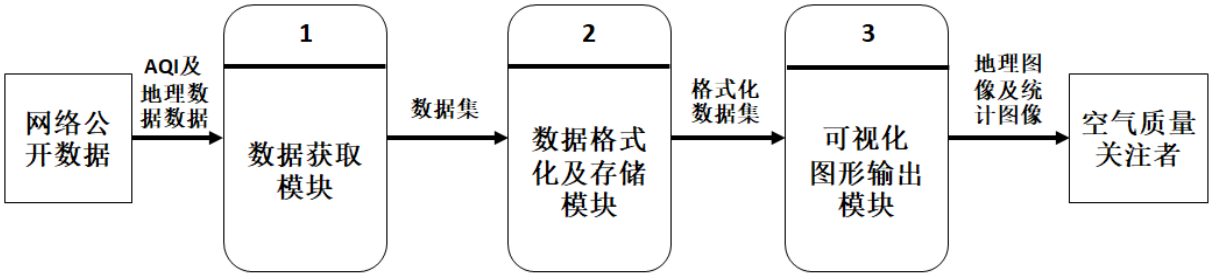
软件规模小，需求明确，开发过程短，采用瀑布模型作为软件生命周期模型。

2　概要设计

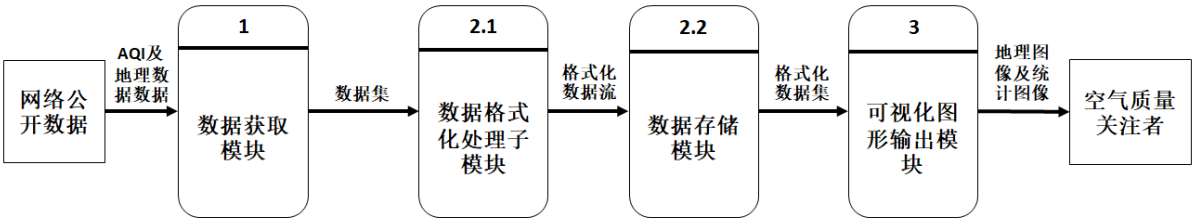
2.1　数据流分析

软件利用网络公开数据生成可视化图像并存储数据，从数据流的角度分析，设计各级数据流图，设计结果如下：

（图 2.1.1）



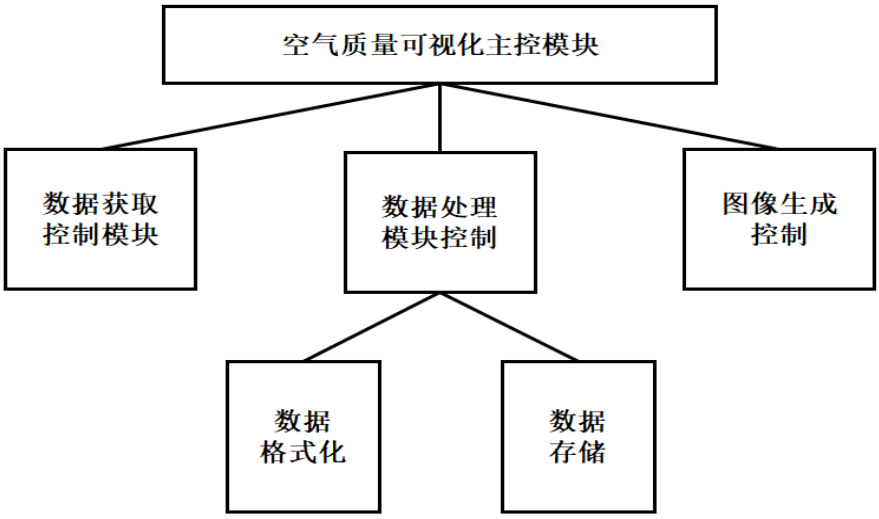
（图 2.1.2）



（图 2.1.3）

2.2 软件结构设计

根据面向数据流的分析过程，应用转换流的转化分析技术，将数据流图划分出逻辑中心，通过求精分析进而得到软件结构图，如下图所示：



（图 2.2.1）

2.3 技术实现分析：

软件应用 REQUEST, PLT, OSMNX, CSV 等 Python 库，基于 OSM 开源项目数据和 http://www.air-level.com/air 公开数据实现对 AQI 指数的可视化和统计分析。

3 详细设计

3.1 AQI 数据获取模块

使用 REQUEST 库，调取网页 http://www.air-level.com/air 的数据，获得东三省各个城市的 AQI 数据，存储为 CSV 格式文件，具体包括城市名称，AQI 指数，污染等级，数据时间，格式如下图：

city	aqi	class	date	time(XX:00)
沈阳	112	轻度污染	2018_1_2	12
大连	56	良	2018_1_2	12
鞍山	36	优	2018_1_2	12
抚顺	74	良	2018_1_2	12
本溪	103	轻度污染	2018_1_2	12
丹东	34	优	2018_1_2	12
锦州	92	良	2018_1_2	12
营口	50	优	2018_1_2	12
阜新	83	良	2018_1_2	12
辽阳	49	优	2018_1_2	12
盘锦	39	优	2018_1_2	12
铁岭	49	优	2018_1_2	12
朝阳	45	优	2018_1_2	12
葫芦岛	51	良	2018_1_2	12
哈尔滨	75	良	2018_1_2	12
齐齐哈尔	135	轻度污染	2018_1_2	12
牡丹江	46	优	2018_1_2	12
佳木斯	38	优	2018_1_2	12
大庆	82	良	2018_1_2	12
伊春	73	良	2018_1_2	12
鸡西	26	优	2018_1_2	12

（图 3.1.1）

3.2 地理数据获取模块

采用 QGIS 图形化系统，查看 OSM 地图数据结构，关注东北三省的图形接口，编写中文与接口定义的对照表用于之后的数据图形化处理的需求，数据结构如下图：

沈阳	shenyang	liaoning	China
大连	dalian	liaoning	China
鞍山	anshan	liaoning	China
抚顺	fushun	liaoning	China
本溪	benxi	liaoning	China
丹东	dandong	liaoning	China
锦州	jinzhou	liaoning	China
营口	yingkou	liaoning	China
阜新	fuxin	liaoning	China
辽阳	liaoyang	liaoning	China
盘锦	panjin	liaoning	China
铁岭	tieling	liaoning	China
朝阳	chaoyang	liaoning	China
葫芦岛	huludao	liaoning	China
哈尔滨	haerbin	heilongji	China
齐齐哈尔	qiqihaer	heilongji	China
牡丹江	mudanjiar	heilongji	China
佳木斯	jiamusi	heilongji	China
大庆	daqing	heilongji	China
伊春	yichun	heilongji	China
鸡西	jixi	heilongji	China

（图 3.2.1）

调用 OSMnx 库函数 ox.gdf_from_places() 获取地图数据，整理数据至 GeoDataFrame 结构列表

3.3 数据格式化模块

处理 AQI 指数数据，选取不同地理位置的代表地点，格式化数据为以城市为索引的数据格式，用于之后展示统计数据图形准备数据。整理后格式如下：

大连	56	58	58	59	67	62	65	66	70	70	67
大兴安岭地	30	24	23	22	27	42	52	75	120	132	116
长春	55	50	49	51	53	67	88	100	106	112	113

（图 3.3.1）

格式化并储存地图数据：由于.shp 数据格式的要求，对不同的数据格式应该分别存储，在已经获取的数据中很大可能存在着点，线，面的数据结构的复合，软件采用点来代表城市的地理信息，利用 isinstance(geometry, (Polygon, MultiPolygon))筛选出需要格式化为 point 数据格式的城市数据利用 bbox_west, bbox_east, bbox_south, bbox_north 等地理数据计算矩形外廓的几何中心来代表城市位置。在用点表述了城市地理位置之后，使用 ox.save_gdf_shapefile()函数将城市位置信息存储于硬盘。

最后，对城市的名称进行调整，将 OSM 数据，AQI 数据中的城市名采用统一标准描述并存储。

3.4 地图图像生成模块

根据 AQI 数据的极值情况，将其合理划分为不同的等级，并指定可视化需要的颜色，以使得之后的可视化效果有重点突出和比较效果。

利用 CSV 库函数取得存储的地理数据，创建 fig 图像及 ax 图像句柄，先将东北三省的轮廓数据描画在地图上，采用白色黑边。之后，读取城市数据，将点数据利用 buffer() 函数转化成二维多边形数据，利用 patch 机制将 city 位置信息加入到图形中去，设置一定的透明度。alpha 一般取 0.6 - 0.9

最后设置图形展现形式，设定图形边框信息，坐标信息，存储格式信息，内存管理信息等，完成图像的生成和存储。

3.5 统计图像生成模块

基于格式化为 city 为索引的数据结构，直接调用 CSV 库函数读取数据，并调用 PLT 库函数接收数据，指定图形格式，直接调用 OSMNX 库函数 save_and_show()展示并存储统计图像。

4 编码实现

4.1 REQUEST 库函数的数据获取

```
#读取城市列表数据
filename = 'cityList.csv'
with open(filename) as f:
    reader = csv.reader(f)
    places = list(reader)
#遍历城市信息，查找每一个城市的 AQI 指数并存储
for place in places:
    url = 'http://www.air-level.com/air/'+place[1]
    req = requests.get(url)
    req.encoding = 'UTF#8'

    soup = bs4.BeautifulSoup(req.text,'lxml')
    aqi_class = soup.find(attrs = {'class':'aqi-bg'}).text
    date_time = soup.find(attrs = {'class':'label label-info'}).text
      • • •  • • •
    datas.append([place[0],int(aqi_class[0]),aqi_class[1],year+'_'+month+'_'+day,time])
```

```
filename = 'aqiData'+ year + '_' + month + '_' + day + '_' + time +'.csv'
with open(filename, 'w', newline='') as f:
    writer = csv.writer(f)
    for row in datas:
        writer.writerow(row)
```

4.2 QGIS 和 OSM 的地理数据获取

```
places = []
place_names = []
#读取城市列表
filename = 'cityList.csv'
with open(filename) as f:
    reader = csv.reader(f)
    places = list(reader)
#遍历城市信息，生成请求数据集合结构
for place in places:
        place_names.append(place[1]+','+place[2]+','+place[3])
#发送请求获取城市地理位置信息
dongSanSheng = ox.gdf_from_places(place_names, gdf_name='dongSanSheng_city')
#遍历城市地理位置信息，格式化为 point 数据格式，再用 buffer()函数处理之后存储
for city_polygon,city_name,e,w,n,s in
  zip(dongSanSheng['geometry'],dongSanSheng['place_name'],dongSanSheng['bbox_east'],d
  ongSanSheng['bbox_west'],dongSanSheng['bbox_north'],dongSanSheng['bbox_south']):
    placeName = placeName[0].strip()
    #识别非 Point 类型的返回值数据
    if isinstance(city_polygon, (Polygon, MultiPolygon)):
        #利用矩形外廓边框几何中心作为城市坐标点数据
        x = (float(e) + float(w))/2
        y = (float(n) + float(s))/2
        city_point_df = GeoDataFrame([{'geometry' : Point(x, y).buffer(0.6)}])
        city_point_polygon = city_point_df['geometry'][0]
        dongSanSheng_city_byPoint =
            GeoDataFrame([{'place_name':placeName, 'geometry':city_point_polygon}])
        #存储城市地理位置数据
```

```
        objectName = placeName + '坐标圆 001'
        ox.save_gdf_shapefile(
            dongSanSheng_city_byPoint,filename=objectName, folder='osmnx_data/city/')
    #识别 point 类型数据，直接存贮在本地
     elif isinstance(city_polygon, Point):
        city_point_df = GeoDataFrame([{'geometry' : city_polygon.buffer(0.6)}])
        city_point_polygon = city_point_df['geometry'][0]
        dongSanSheng_city_byPoint =
            GeoDataFrame([{'place_name':placeName, 'geometry':city_point_polygon}])
        objectName = placeName + '坐标圆 001'
        ox.save_gdf_shapefile(
            dongSanSheng_city_byPoint,filename=objectName, folder='osmnx_data/city/')
     else: #否则报错：类型不匹配
        print('wrong datatype!' )
```

4.3 OSMNX 和 PLT 库函数的数据处理及图像生成

```
# 指定形成数据的时间范围，并遍历数据，画出图形
for i in range(12,22):
    fig, ax = plt.subplots(figsize=(6,6))
    #读取东北三省轮廓信息
    placedf = GeoDataFrame.from_file('osmnx_data\province\province.shp')
    #将东北三省的轮廓信息加入到图中
    for geometry in placedf['geometry']:
        if isinstance(geometry, (Polygon, MultiPolygon)):
            if isinstance(geometry, Polygon):
                geometry = MultiPolygon([geometry])
            for polygon in geometry:
                print(polygon)
                patch = PolygonPatch(
                                polygon, fc='#ffffff', ec='#555555', linewidth=1, alpha=1)
                ax.add_patch(patch)
        else:
            raise ValueError(' GeoDataFrame must be Polygons or MultiPolygons')
    #读取该趟循环中的 AQI 数据信息
    csv_reader = csv.reader(open('aqiData2018_1_2_'+ str(i) +'.csv', encoding='gb2312'))
```

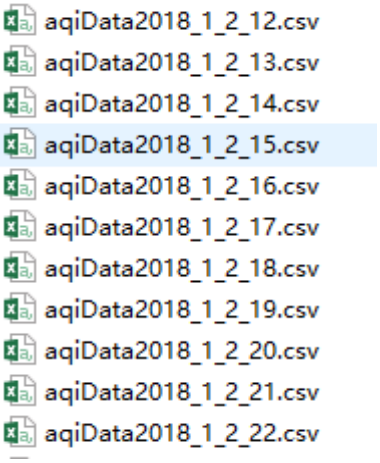
```
#指定相应的颜色，并将数据点加入到图形中
for city in csv_reader:
    if city[0] == 'city':
        continue
    city_name = city_name + '市'
    #指定颜色
    if int(city[1])<20:
        color = '#eeeeee'
    • • •  • • •
    elif int(city[1]) < 150:
        color = '#111111'
    else:
        color = '#000000'
    #读取点的位置信息
    city_point_df = GeoDataFrame.from_file(
        'osmnx_data/city/' + city_name+'坐标圆 001/' +city_name+'坐标圆 001.shp')
    polygon = city_point_df['geometry'][0]
    patch = PolygonPatch(polygon, fc=color, ec='#555555', linewidth=0, alpha=0.9)
    #加入点到图形中
    ax.add_patch(patch)
#调整输出图形的格式和内存设置
margin = 0.02
axis_off= True
west, south, east, north = placedf.unary_union.bounds
margin_ns = (north - south) * margin
margin_ew = (east - west) * margin
ax.set_ylim((south - margin_ns, north + margin_ns))
ax.set_xlim((west - margin_ew, east + margin_ew))
ax.set_aspect(aspect='equal', adjustable='box')
if axis_off:
    ax.axis('off')
#保存图形到硬盘
ox.save_and_show(fig,ax,save = True,filename = 'city_point_pic/'+ str(i), show = False,
close = True, file_format = 'png', dpi = 300 , axis_off = True)
```

5 测试

5.1 单元测试

5.1.1 REQUEST 数据获取单元测试

分别在 12:00， 13:00，14:00，15:00，16:00，17:00，18:00，19:00，20:00，21:00，22:00 爬取 AQI 数据，预计形成格式化的数据存储，测试结果，成功获得格式化数据集合。



(图 5.1.1.1)

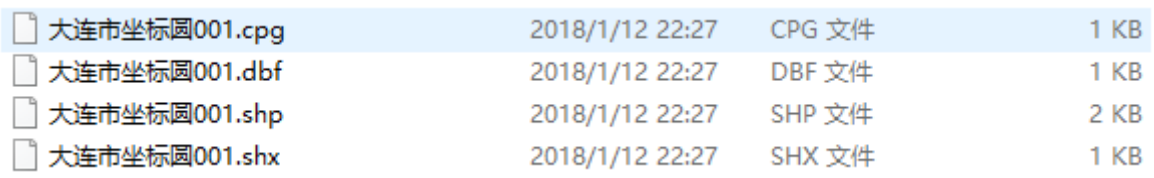
绥化	63	良	2018_1_2	17
黑河	60	良	2018_1_2	17
大兴安岭地	42	优	2018_1_2	17
长春	67	良	2018_1_2	17
吉林	33	优	2018_1_2	17
四平	52	良	2018_1_2	17
辽源	47	优	2018_1_2	17
通化	66	良	2018_1_2	17
白山	37	优	2018_1_2	17
白城	57	良	2018_1_2	17
松原	70	良	2018_1_2	17
延边州	36	优	2018_1_2	17

(图 5.1.1.2)

5.1.2 OSMNX 的 SHP 文件生成单元测试

在请求获得城市数据时应生成，指定格式的城市数据文件夹，测试结果，获得正确格式的文件夹。

(图 5.1.2.1)



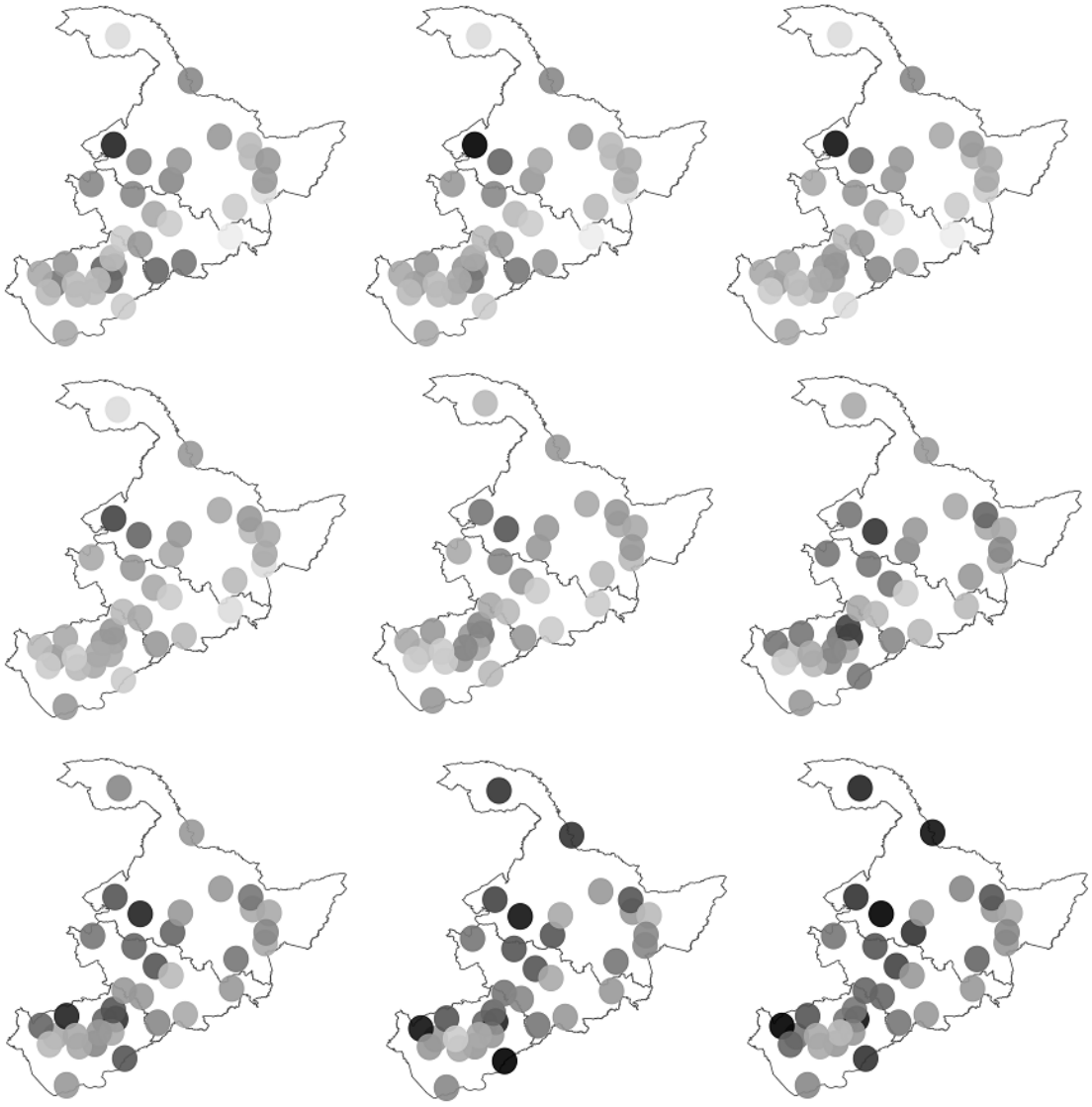
(图 5.1.2.2)

5.2 集成测试

图像生成模块测试：

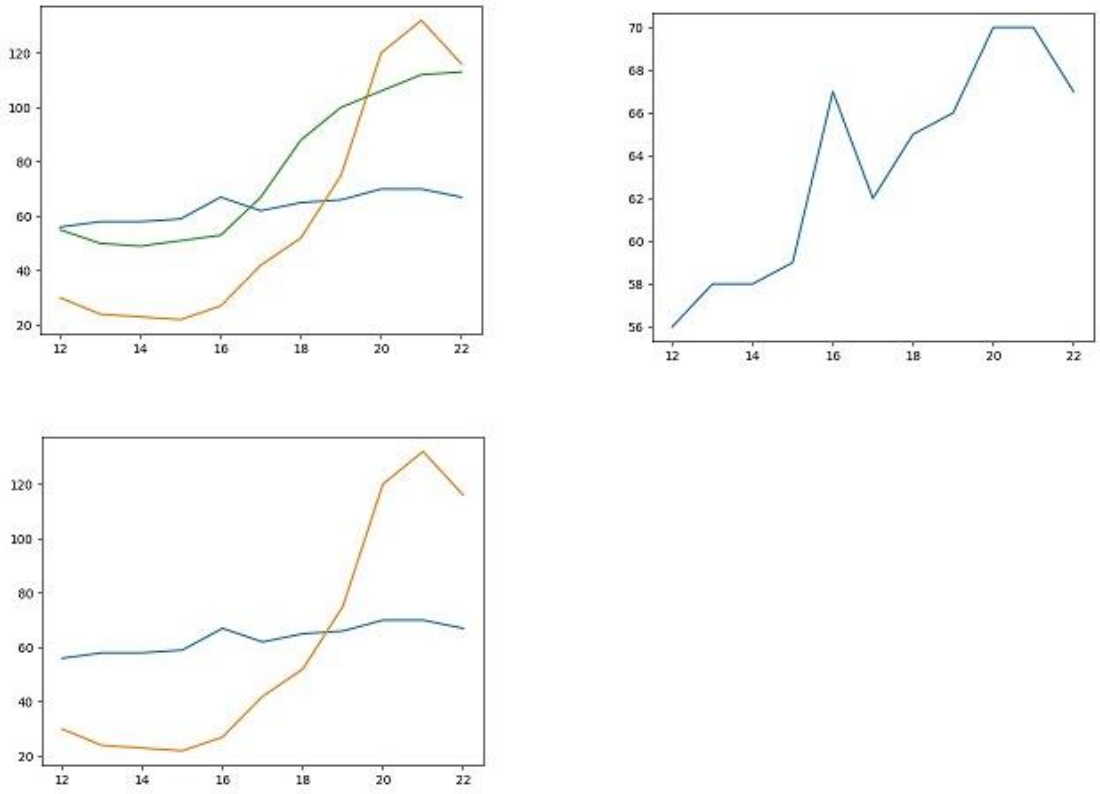
在捕捉到一定的数据量之后，启动图像的生成程序，查看统计图表的输出情况和地图可视化的输出情况是否符合规范。测试结果如下所示：

2018 年 1 月 12 日 13:00 到 21:00 东北三省各城市 AQI 指数变化情况：



（图 5.2.1）

大连（蓝色），大兴安岭（红色），长春（绿色）2018 年 1 月 12 日下午 AQI 变化情况：



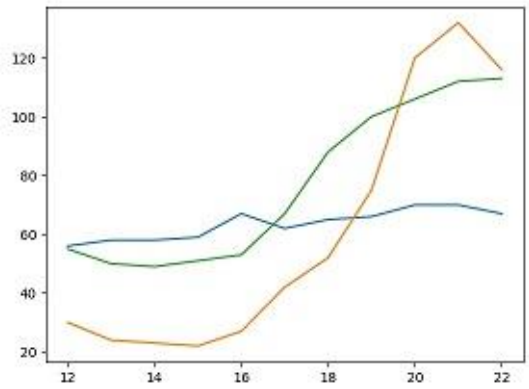
（图 5.2.2）

6　项目结束

6.1　过程模型总结

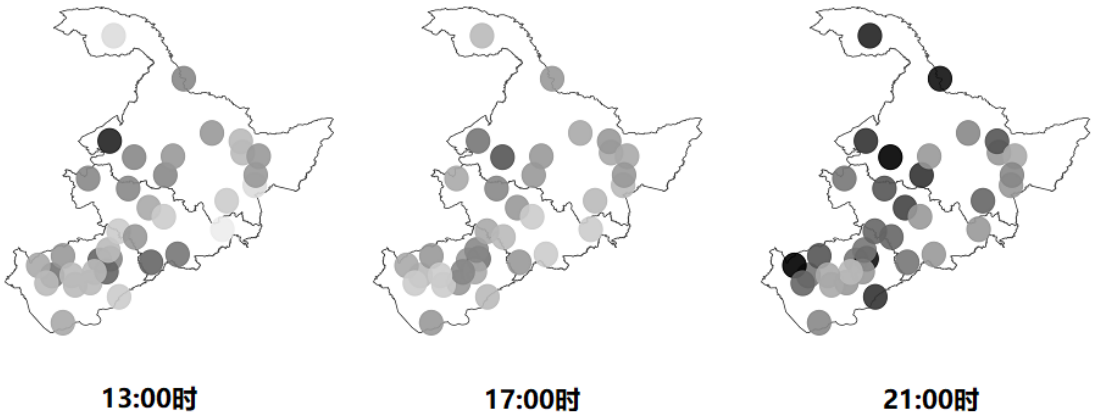
成果中采用瀑布模型进行开发，在完成了概要设计的基础之上进行编码和测试，一定程度上保证了软件系统开发的完整性，软件系统的目标达成度保证有所提高，但是，由于对相关技术的了解不是非常全面，在详细设计阶段没有对所有的情况进行考虑，导致详细设计没有对实现编码起到应有的作用。应进一步加强。

6.2 分析结果概述



（图 6.2.1）

从数据可视化的结果来看，大连等沿海城市 AQI 的变化比较稳定，而越靠近内陆，AQI 的变化显现出越大的波动性，另外，以大兴安岭为例，黑龙江北部地区的 AQI 呈现出极小值小，极大值大的规律。



（图 6.2.2）

由东三省整个下午时段的 AQI 数据来看，在下午 17:00 左右，全区域的 AQI 指数出现了明显的上扬趋势，推测是由于下班高峰期交通增多，更重要的是，北方冬天的燃煤取暖在夜间会出现峰值，而夜间的空气流通较不通畅，冷空气下压从而导致了夜间 AQI 指数的上升。