BESTオープンソースの利用法

計算エンジン部分のソース(言語：Java)が公開されている。
(https://www.ibec.or.jp/best/application_opensource.html、建築計算部分は全て公開)

以下のソース利用法3つのうち、最も利用しやすいのは3番目といえる。

◑ソースを改造

巨大プログラムのためかなり困難

◑ユーザーインターフェースを新規作成

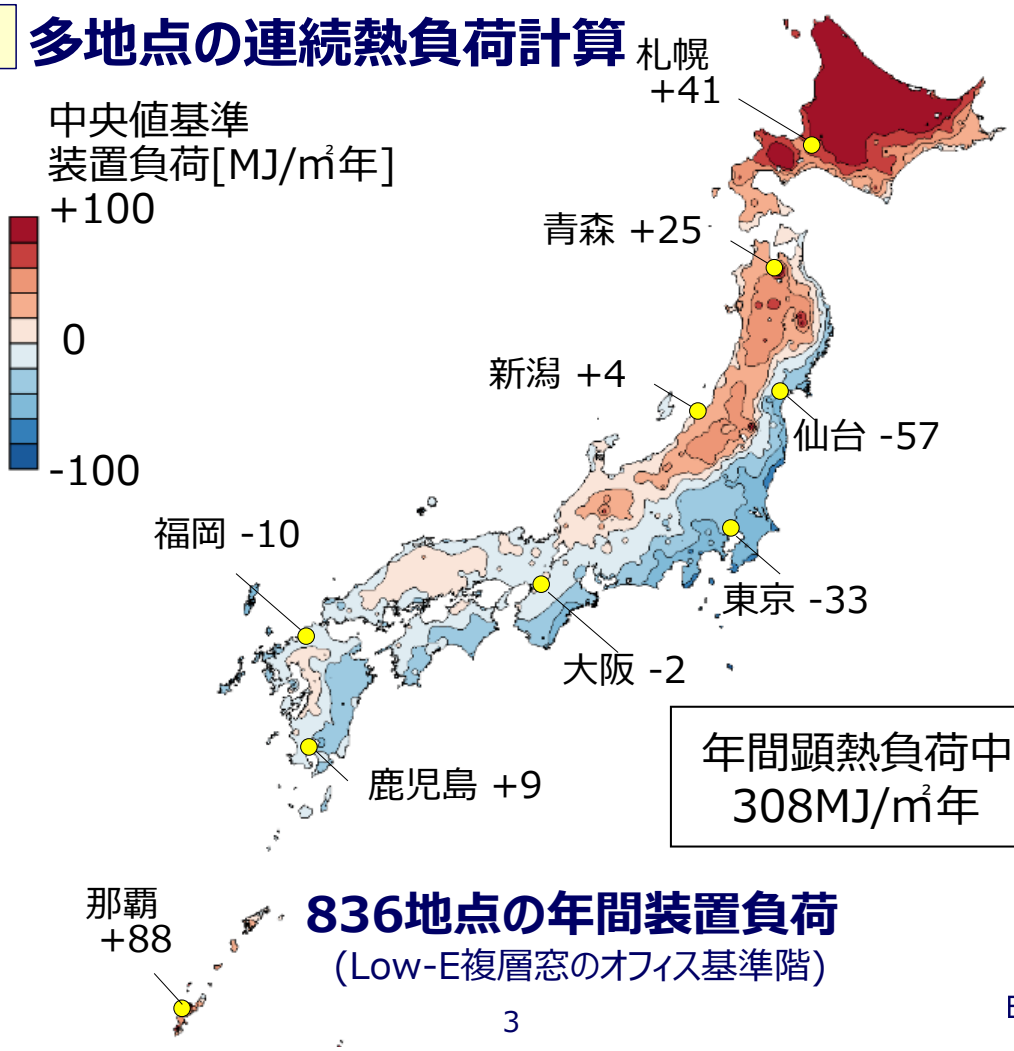
エンジン入力データの構成や詳細に関する資料は、ソースとともに公開されている。それを利用可能

◑連続実行による多ケースシミュレーション

・ソースは改造しない
・多ケース連続実行のプログラム作成
　入力データの変更・エンジン実行・出力データの抽出の繰り返し処理
・BEST-Pなどの既存UIで、基準ケースのエンジン入力データを作成
　これを変更しながら多ケースシミュレーションを行う

計算例

多地点の連続熱負荷計算



年間顕熱負荷中央値：
308MJ/㎡年

836地点の年間装置負荷
(Low-E複層窓のオフィス基準階)

the BEST Program
Building Energy Simulation Tool

建築エンジンを利用する
多ケースシミュレーション

計算例

多地点の年間熱負荷計算*1を連続実行し、各都市の出力ファイルから年積算負荷を抽出し別のファイルに書き出す。p.3のカラーマップは、オフィスビル基準階5ゾーンについて実行し、得られた国内836地点の年間装置負荷から作成したものである*2 。

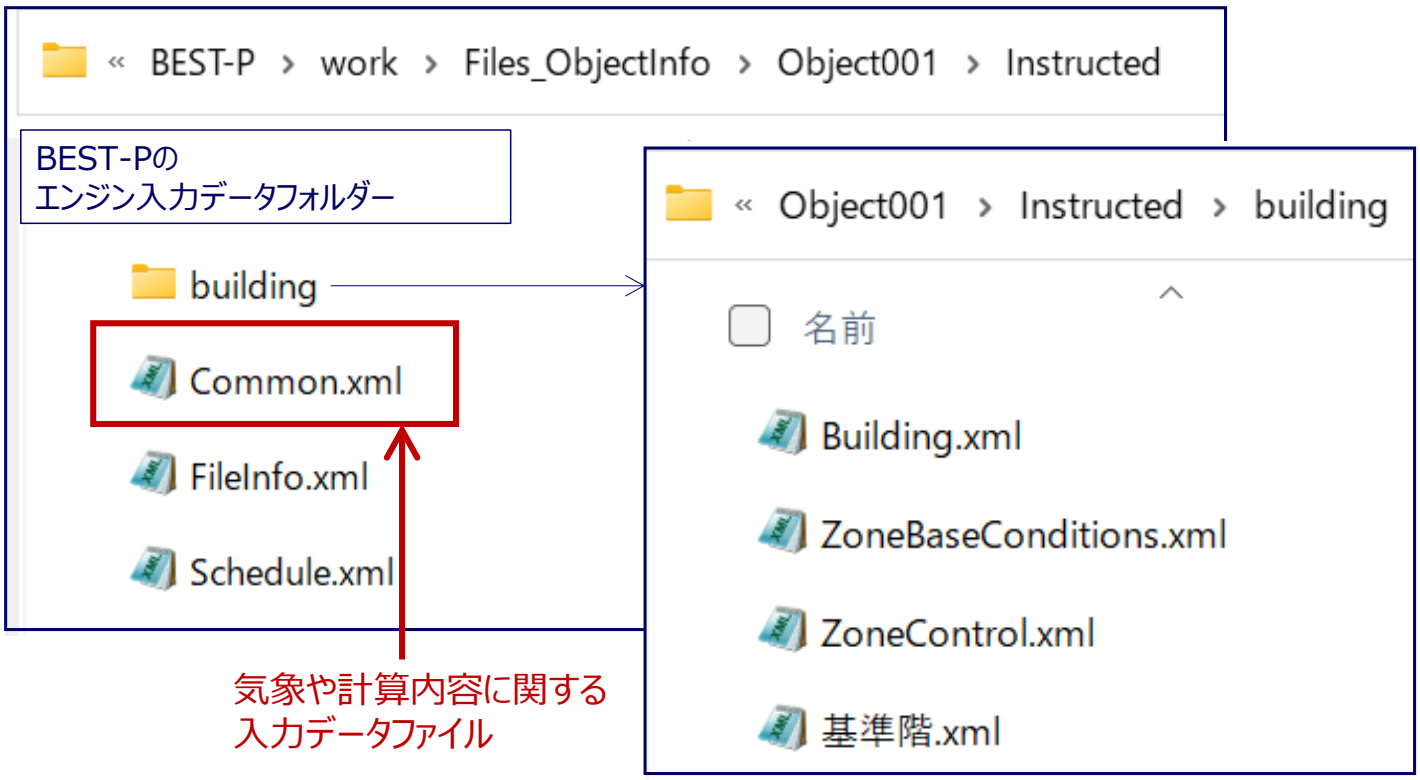
*1 年間熱負荷計算の前に最大熱負荷計算を実行して、装置容量を自動設定する。
*2 演算時間は30分程度。カラーマップ作製ツールは、ColorMap(MestDS、作者：秋田県立大 松本真一教授)

以降は、多地点の熱負荷連続計算プログラムを新規に作成するために役に立つ以下の項目について説明する。

- **エンジン入力データの変更箇所**
- **出力データの抽出箇所**
- **新規作成プログラムで利用するオープンソースクラス**
- **サンプルプログラム**

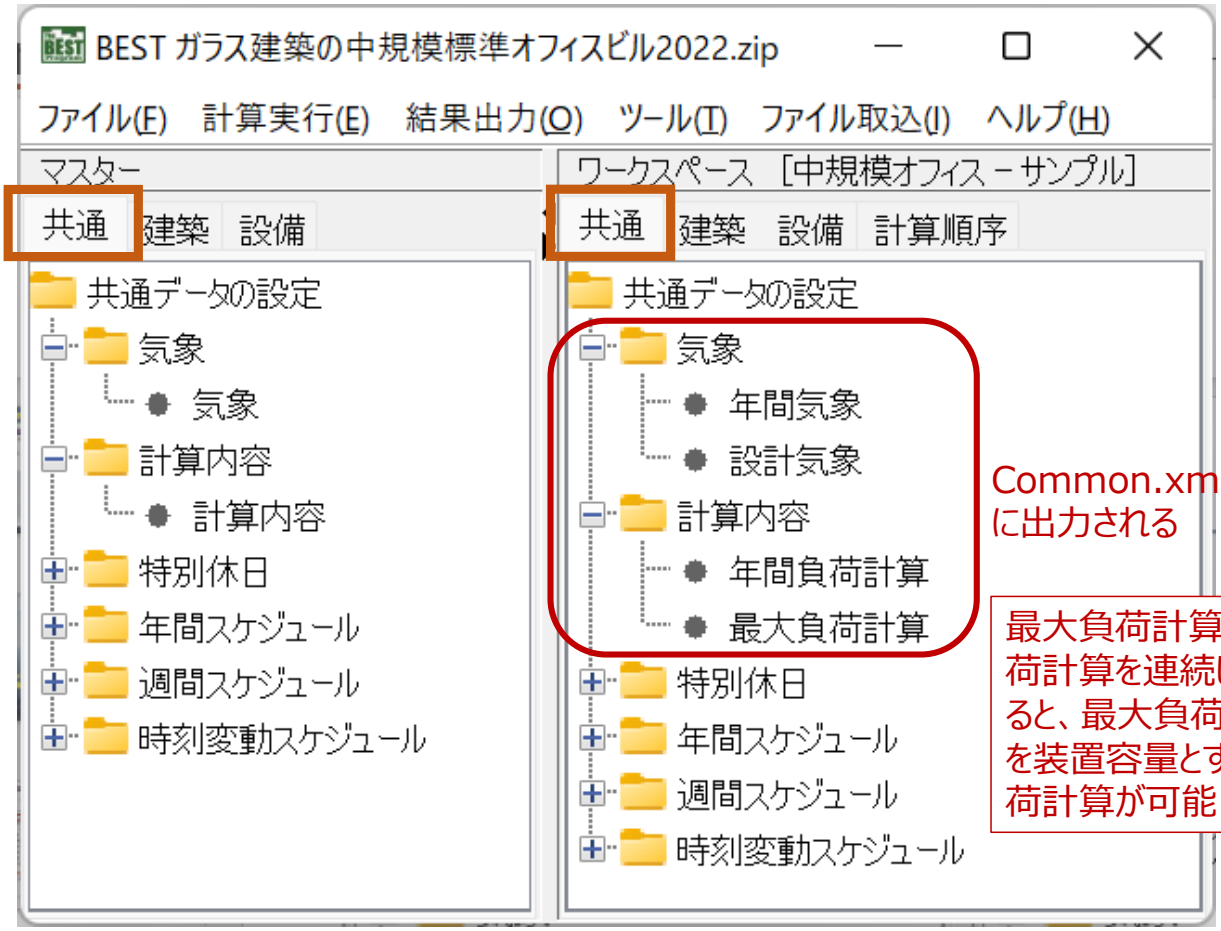
建築エンジンの入力データ

専門版(BEST-P)で作成したエンジン入力データを利用する



*各XMLファイルのフォーマット説明は、オープンソースの添付資料として公開されている。

専門版(BEST-P)のデータ設定とCommonファイルの関係



Common.xmlファイル
に出力される

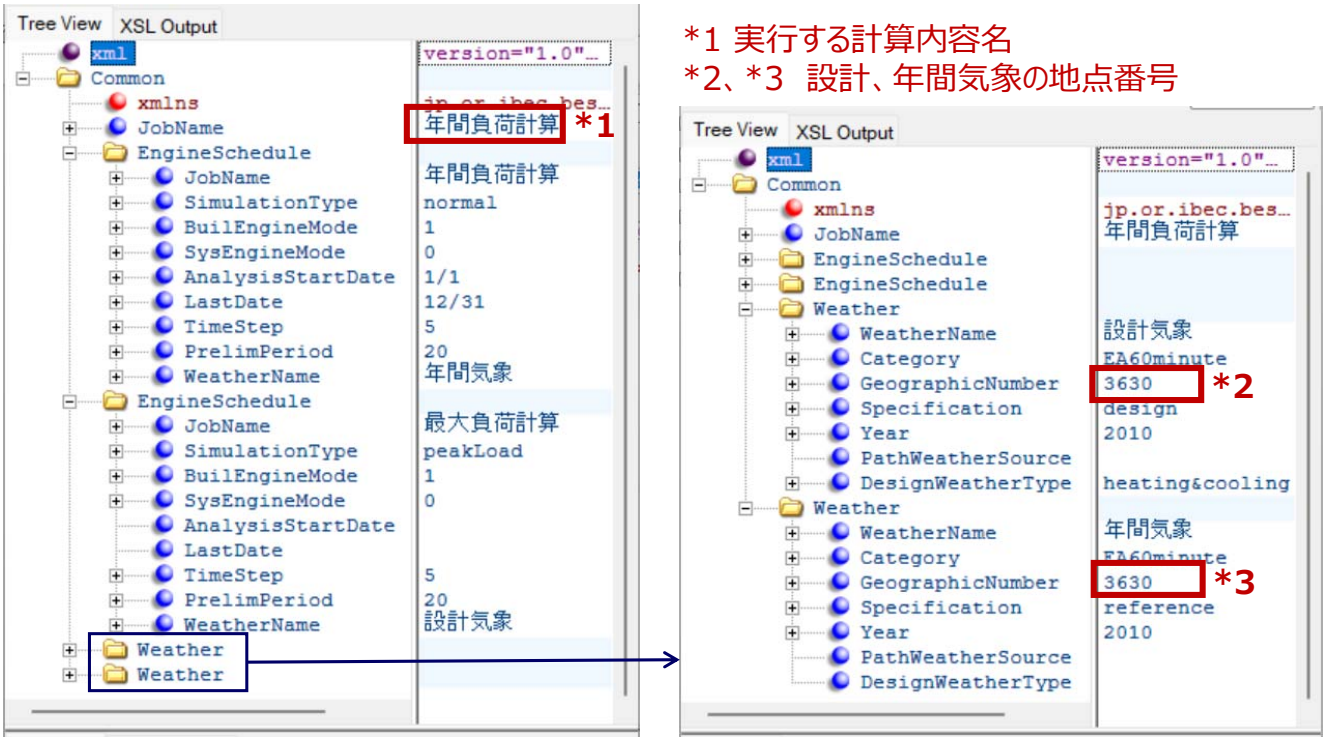
最大負荷計算と年間負荷計算を連続して実行すると、最大負荷計算結果を装置容量とする年間負荷計算が可能

BEST-P 共通画面

多地点連続計算のための入力データ変更

・地点番号(*2、*3)：10、20、…、8420 (欠番あり)　836ケース実行
　　地点番号と地点名は、建築操作マニュアルの付録参照
・ジョブ名(*1)：1地点につき、「最大負荷計算」、「年間負荷計算」の2ケース実行

*1 実行する計算内容名
*2、*3　設計、年間気象の地点番号

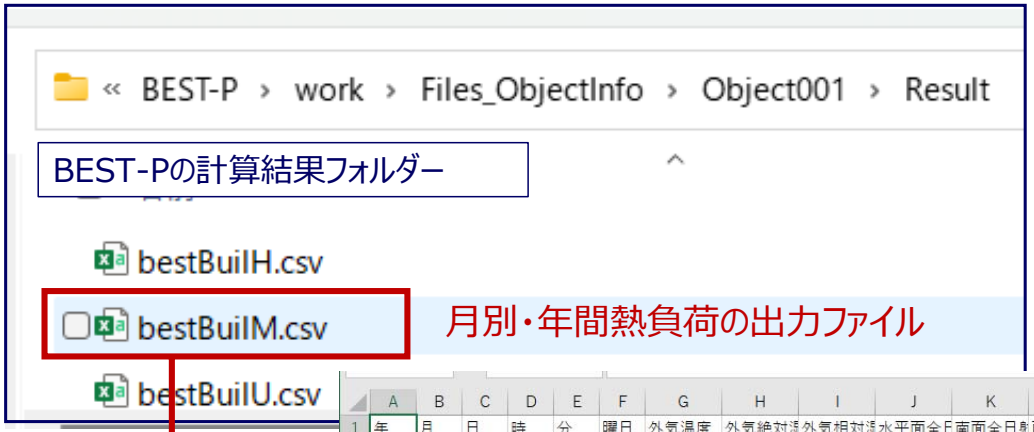


Common.xmlデータファイルの内容

BEST建築計算例

多地点連続計算のための出力データの抽出

ファイル「bestBuilM.csv」の年積算負荷出力行を抽出して別ファイルに出力



*出力データファイルと内容の詳細は、建築操作マニュアル参照

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
1	年	月	日	時	分	曜日	外気温度	外気絶対湿	外気相対湿	水平面全日	南面全日射	西面全日射	北面全日射	東面全日射	全ゾーン_(	全ゾーン_(
2	–	–	–	–	–	–	°C	g/g	%	MJ/m2	MJ/m2	MJ/m2	MJ/m2	MJ/m2	MJ/m2	MJ/m2
3							気象	気象	気象	気象	気象	気象	気象	気象	室負荷S_(	室負荷S_(
4	2006	1	99	99	99	99	6.13	0.0027	46.45	292.04	444.17	175.27	73.56	187.91	-3.3	10.8
5	2006	2	99	99	99	99	6.6	0.0029	47.12	304.54	335.28	172.37	81.78	184.72	-3.41	10.79
6	2006	3	99	99	99	99	9.86	0.0037	48.15	443.59	344.4	235.28	119.61	253.13	-0.98	14.56
7	2006	4	99	99	99	99	15.11	0.0057	53.89	520.85	287.56	280.05	139.27	274.05	-0.11	17.3
8	2006	5	99	99	99	99	18.38	0.0083	64.16	494.07	212.74	253.17	159.27	257.84	0	21.81
9	2006	6	99	99	99	99	23.13	0.0124	70.3	418.04	173.44	209.04	155.34	222.88	0	31.42
10	2006	7	99	99	99	99	27	0.0159	71.04	548.12	222.03	290.07	175.35	260.62	0	40.28
11	2006	8	99	99	99	99	28.04	0.0161	68.02	488.24	241.78	260.02	151.96	249.89	0	46.59
12	2006	9	99	99	99	99	24.38	0.0133	69.23	367.83	240.28	205.9	113.67	194.2	0	33.43
13	2006	10	99	99	99	99	18.74	0.0082	61.68	328.48	318.69	177.47	93.43	193.41	0	24.28
14	2006	11	99	99	99	99	13.44	0.0061	62.26	234.43	302.41	138.49	67.29	140.5	-0.19	15.88
15	2006	12	99	99	99	99	9.02	0.0037	60.41	259.31	432.79	173.59	64.76	158.05	-1.25	14.51
16	2006	13	99	99	99	99	16.71	0.0083	59.44	4699.55	3555.58	2570.71	1395.3	2577.2	-9.24	281.66

bestBuilM

全ゾーン・各ゾーンの年積算負荷の出力行

利用するオープンソースのプログラム

新規作成プログラムで、メソッドを利用するBESTプログラム

- 熱負荷計算用メインクラス

  jp.or.ibec.best.domain.building.test.control.**HVACsysBuilSimulator**

- Common.xmlファイルの入出力処理クラス

  UnMarshalUtil、MarshalUtilクラスで、XMLファイルのデータの読込み、書出しを行い、読込んだCommonファイルのデータをCommonクラスで管理する。EngineSchedule、Weatherクラスは、Commonファイルのデータのうち、計算内容、気象に関するデータを管理するクラス。

  jp.or.ibec.best.domain.building.xml.common.**Common*、EngineSchedule*、Weather***

  jp.or.ibec.best.domain.building.xml.util.**UnMarshalUtil、MarshalUtil**

  *を付したクラスは、Javaの機能により自動生成されたクラス。
  タグ名をもとに、読込んだ個々のデータを取り出したり、再設定したりできる。

サンプルプログラム

プログラム言語：Java

- メインクラス　**MainSample**

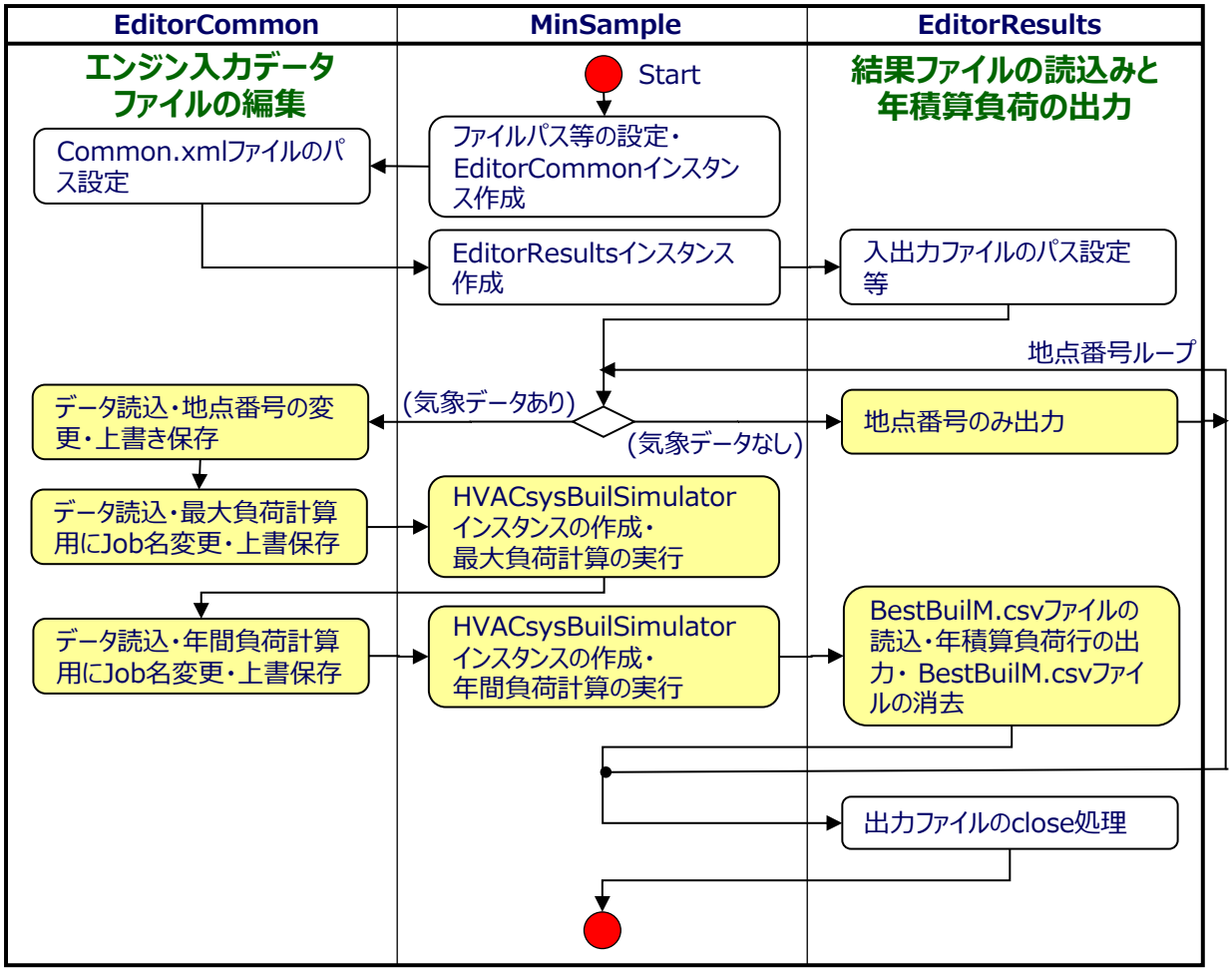
  欠測地点をはずしながら836地点の最大・年間熱負荷を実行し、年間熱負荷結果を抽出して別のファイルに保存する。

- Commonデータファイルの編集クラス　**EditorCommon**

  Common.xmlファイルを読込み、地点名やジョブ名を変更して上書き保存する。

- 年間負荷計算結果の編集クラス　**EditorResults**

  BestBuilM.csvファイルを読込み、別のファイル(bestAnnualLoad.csv)に、地点番号とともに書き出す。



アクティビティ図

実行準備

環境変数

計算に利用するデータ類が保存されたフォルダーを環境変数で設定する。
熱負荷計算用に設定が必要な環境変数は下表参照

利用する外部プログラム

新規作成プログラムの実行に必要なjar(BESTプログラムを除く)は、オープンソース資料に含まれるbest-sample.libフォルダー内にある。

熱負荷計算用の環境変数と設定例

環境変数	フォルダーパス	
	フォルダーの内容	パス例
BEST_INSTRUCTED	エンジン入力データ	D:¥BEST-P¥work¥Files_ObjectInfo¥Object001¥Instructed
BEST_SPEC	〃	D:¥BEST-P¥work¥Files_ObjectInfo¥Object001¥Instructed
BEST_RESULT	計算結果	D:¥BEST-P¥work¥Files_ObjectInfo¥Object001¥Result
BEST_WEATHER	気象データ	D:¥BEST-P¥work¥weather
BEST_XML	建築データベース	D:¥BEST-P¥work¥XML
BEST_USER_DB	ユーザー定義データベース	D:¥BEST-P¥work¥userXML
BEST_XSD	建築XSD	D:¥BEST-P¥sys¥XML¥xsd

＊ パス例は、BEST-Pフォルダーを利用するときの例

年積算負荷の多地点集計結果

地点番号

		年	月	日	時	分	曜日	外気温度	外気絶対湿	外気相対湿	水平面全日	南面全日射	西面全日射	北面全日射	東面全日射	全ゾーン_6	全ゾーン_6	全
2		–	–	–	–	–	–	℃	g/g	%	MJ/m2	MJ/m2	MJ/m2	MJ/m2	MJ/m2	MJ/m2	MJ/m2	M
3								気象	気象	気象	気象	気象	気象	気象	気象	室負荷S_暖	室負荷S_冷	室
4	10	2006	13	99	99	99	99	6.61	0.0055	77.11	3923.01	2907.72	2385.54	1254.45	2197.54	-104	139.27	
5	20	気象観測終了地点の行																
6	30	2006	13	99	99	99	99	7.03	0.0056	75.39	4207.42	3167.8	2444.3	1287.08	2386.48	-96.11	153.66	
7	40	2006	13	99	99	99	99	5.67	0.0054	81.76	4005.8	3194.8	2361.53	1269.04	2338.86	-107.16	135.61	
8	50	2006	13	99	99	99	99	5.83	0.0054	80.51	3999.63	2987.49	2443.26	1292.58	2168.84	-115.07	136.42	
9	60	2006	13	99	99	99	99	7.42	0.0057	74.97	3911.35	2898.39	2361.95	1275.62	2103.64	-100.5	150.32	
10	70	2006	13	99	99	99	99	6.77	0.0056	77.4	3807.72	2827.74	2224.61	1284.35	2126.37	-109.52	142.32	
11	80	2006	13	99	99	99	99	5.96	0.0054	80.01	4093.83	3151.04	2340.25	1301.95	2361.19	-108.87	140.09	
12	90	2006	13	99	99	99	99	5.46	0.0053	82.3	3941.92	2803.9	2280.81	1293.78	2199.77	-128.88	136.13	
13	100	2006	13	99	99	99	99	6.29	0.0053	76.56	3991.26	3161.27	2255.71	1287.91	2395.75	-101.24	141.81	
14	110	2006	13	99	99	99	99	5.56	0.0054	81.09	3997.7	2968.02	2332.11	1290.74	2239.76	-121.28	138.08	
15	120	2006	13	99	99	99	99	6.04	0.0055	79.86	3925.17	2837.86	2344.44	1285.78	2120.2	-118.98	138.36	
16	130	2006	13	99	99	99	99	5.86	0.0053	79.89	3895.38	2695.14	2237.35	1285.47	2091.71	-127.19	136.77	
17	140	2006	13	99	99	99	99	6.28	0.0055	80.2	4008.22	2851.73	2356.55	1296.25	2099.88	-124.57	147.47	
18	150	2006	13	99	99	99	99	5.85	0.0054	80.91	3981.59	2951.37	2311.45	1307.08	2090.48	-125.9	142	
19	160	2006	13	99	99	99	99	5.57	0.0053	80.56	4121.44	3145.28	2399.01	1337.37	2298.61	-115.85	139.4	
20	170	2006	13	99	99	99	99	6.16	0.0056	81.55	4061.18	3026.68	2359.61	1312.69	2225.52	-119.85	145.71	
21	180	2006	13	99	99	99	99	6.05	0.0054	80.05	4216.57	3125.16	2438.28	1335.47	2376.98	-116.77	149.25	

bestAnnualLoad

集計結果ファイル(Resultフォルダー内のbestAnnualLoad.csv)**の内容例**


```
package bestSample;
import jp.or.ibec.best.domain.building.test.control.HVACsysBuilSimulator;

public class MainSample {
    public static void main(String[] args) {
        MainSample main=new MainSample();
        main.run();
    }

    public void run() {
        String pathXml=System.getenv("BEST_INSTRUCTED")+"¥¥Common.xml";//Commonファイルパス
        String pathRead=System.getenv("BEST_RESULT")+"¥¥bestBuilM.csv";//BEST結果ファイルパス
        String pathPrint=System.getenv("BEST_RESULT")+"¥¥bestAnnualLoad.csv";//多地点年間負荷出力ファイルパス
        int[] cityIdMissing= {20, 960, 2560, 3160, 3640, 5640};//気象データのない地点番号

        EditorCommon editorCommon=new EditorCommon();//Commonデータの編集インスタンス
        editorCommon.initialize(pathXml);
        EditorResults editorResults=new EditorResults();//結果編集インスタンス
        editorResults.initialize(pathRead, pathPrint);

        int iMissing=0;
        for(int cityId=10; cityId<=8420; cityId=cityId+10) {//開始、最終の地点番号は修正可能
            boolean target=true;
            if(iMissing<cityIdMissing.length) {
                while(cityId>cityIdMissing[iMissing]) iMissing++;
                if(cityId==cityIdMissing[iMissing]) {//気象データのない地点
                    target=false;
                    iMissing++;
                }
            }
            if(!target)  {//気象データのない地点
                editorResults.pickupResult(String.valueOf(cityId));//地点番号のみ出力
            }else {//気象データのある地点
                editorCommon.editCity(cityId);//Commonデータの地点番号の変更
                try{
                    //最大負荷計算
                    editorCommon.editJobName("peak");
                    HVACsysBuilSimulator simulator=new HVACsysBuilSimulator();
                    simulator.start();
                    simulator.join();
                    //年間負荷計算
                    editorCommon.editJobName("annual");
                    simulator=new HVACsysBuilSimulator();
                    simulator.start();
                    simulator.join();
                } catch (InterruptedException e) {
                    System.out.println(e);
                }
                editorResults.pickupResult(String.valueOf(cityId));//年間負荷の抽出と出力
            }
        }
        editorResults.complete();
    }
}
```

地点番号ループ

メインクラス MainSample

Commonデータ編集クラス EditorCommon

```
package bestSample;
import java.util.List;

import jp.or.ibec.best.domain.building.xml.common.Common;
import jp.or.ibec.best.domain.building.xml.common.EngineSchedule;
import jp.or.ibec.best.domain.building.xml.common.Weather;
import jp.or.ibec.best.domain.building.xml.util.MarshalUtil;
import jp.or.ibec.best.domain.building.xml.util.UnMarshalUtil;

public class EditorCommon {
    private String pathXml;

    /**
     * 初期設定
     * @param pathXml : Common.xmlファイルのパス
     */
    public void initialize(String pathXml){
        this.pathXml=pathXml;
    }

    /**
     * 地点番号の変更
     * @param cityId : 地点番号
     */
    public void editCity(int cityId){
        if(cityId<=0 || cityId>8420) return;
        Common common=this.getCommon(pathXml);//Commonデータの取得
        List<Weather> weathers=common.getWeather();
        for(Weather weather : weathers){
            if(weather.getSpecification().equals("design")
                || weather.getSpecification().equals("reference")
                || weather.getSpecification().equals("yearly")){
                //気象データタイプが設計用、標準年、実在年のとき、地点番号の変更
                weather.setGeographicNumber(String.valueOf(cityId));
            }
        }
        this.setCommon(common, pathXml);//Commonデータの保存
    }

    /**
     * ジョブ名の変更
     * @param calType : 計算タイプ。 "peak"のとき最大負荷計算、"annual"のとき年間計算
     */
    public void editJobName(String calType){
        Common common=this.getCommon(pathXml);
        List<EngineSchedule> engineSchedules=common.getEngineSchedule();
        for(EngineSchedule engineSchedule : engineSchedules){
            String simType=engineSchedule.getSimulationType();
            if((calType.equals("peak") && simType.equals("peakLoad"))
                    || (calType.equals("annual") && simType.equals("normal"))){
                common.setJobName(engineSchedule.getJobName());//ジョブ名の設定
            }
        }
        this.setCommon(common, pathXml);
    }
```

地点番号の変更

ジョブ名の変更

```
    /**
     * Commonデータの読込み
     * @param pathXml : Common.xmlのパス
     * @return : Commonデータ
     */
    private Common getCommon(String pathXml){
        Common common=null;
        try {
            common = UnMarshalUtil.getCommon(pathXml, null);
        } catch (Exception e) {
            e.printStackTrace();
        }
        return common;
    }

    /**
     * Commonデータの出力
     * @param common : Commonデータ
     * @param pathXml : Common.xmlのパス
     */
    private void setCommon(Common common, String pathXml){
        try {
            MarshalUtil.putCommon(common, pathXml, null);
        } catch (Exception e) {
            e.printStackTrace();
        }
    }
}
```

Commonデータ読込み

Commonデータ書出し

計算結果データ編集クラス EditorResults

```
package bestSample;
import java.io.BufferedReader;
import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileReader;
import java.io.FileWriter;
import java.io.IOException;
import java.io.PrintWriter;
import java.nio.file.Files;
import java.nio.file.Paths;

public class EditorResults{
    private String pathRead;
    private BufferedReader reader;
    private PrintWriter writer;
    private boolean first=true;

    /**
     * 初期設定
     * @param pathRead : bestBuilM.csvファイルのパス
     * @param pathPrint : 出力ファイルのパス
     */
    public void initialize(String pathRead, String pathPrint){
        this.pathRead=pathRead;
        File fileRead=new File(pathRead);
        if(fileRead.exists()) this.delete(pathRead);

        try {
            File file=new File(pathPrint);
            writer=new PrintWriter(new FileWriter(file));
        } catch (IOException e) {
            e.printStackTrace();
        }
    }

    /**
     *  年間負荷の抽出とファイル出力
     * @param stringCityId : 地点番号
     */
    public void pickupResult(String stringCityId){
        File fileRead=new File(pathRead);
        if(!fileRead.exists()){//結果ファイルがないとき
            writer.println(stringCityId);//地点番号のみ出力
            return;
        }

        reader=null;
        try{
            reader=new BufferedReader(new FileReader(pathRead));
        }catch(FileNotFoundException e){
            e.printStackTrace();
            return;
        }
```

年間負荷の抽出と別ファイルへの書出し

```
        if(first==true){//ラベル行の読込みと出力
            for(int i=0; i<3; i++) {
                String s=this.getNextLine();
                writer.println(","+s);
            }
            first=false;
        }else {
            for(int i=0; i<3; i++) this.getNextLine();
        }

        for(int i=0; i<12; i++) this.getNextLine();
        String s=this.getNextLine();//年間負荷の出力行の読込み
        writer.println(stringCityId+","+s);//地点番号と年間負荷行の出力
        writer.flush();

        try{
            reader.close();
        } catch( IOException e){
            e.printStackTrace();
        }
        this.delete(pathRead);
    }

    public void complete(){
        writer.close();
    }

    private void delete(String pathRead){
        try{
            Files.delete(Paths.get(pathRead));
        }catch(IOException e){
            e.printStackTrace();
        }
    }

    private String getNextLine(){
        String str=null;
        try{
            str=reader.readLine();
        }catch (IOException e) {
            e.printStackTrace();
        }
        return str;
    }
}
```

参考文献

1) 郡：外皮の高性能化技術の変遷と性能予測、日本建築学会シンポジウム「多角的な視点から見た今後の外皮性能のあり方」、pp.1-6、2019.3
2) BEST-P　建築操作マニュアル、https://www.ibec.or.jp/best/tec_info.html