

JDemetra+

*an open framework for seasonal
adjustment and time series
methods for official statistics*

ESTP training

Outline

- Objectives of JDemetra+ (JD+)
 - General
 - For seasonal adjustment (SA)
- What is really JD+ ?
- Architecture, design
- Seasonal adjustment framework
 - Overview, pre-processing, decomposition
- State space framework
 - Goals, overview
- Some examples
- Final remarks

General Objectives

- Providing algorithms for the production/analysis of [official] statistics
 - Regular time series (from monthly to yearly)
 - Algorithms for
 - Seasonal adjustment, business cycle analysis
 - Benchmarking, temporal disaggregation
 - Modelling (forecasting, estimation of missing values, outliers detection)
- Reusable modules, compatible with common IT infrastructure
 - Java, WEB services...
- Designed for the whole statistical process
 - From research to bulk production (flexible, high-performance)
- Maintainable
 - Open source solution

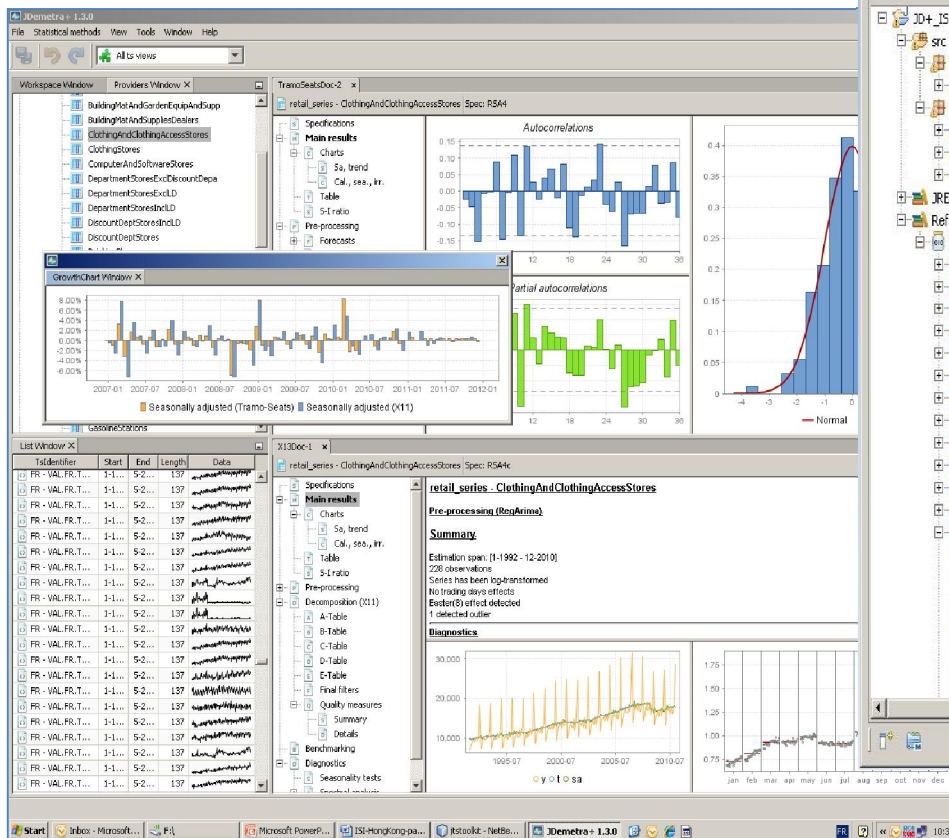
Objectives for SA

- Java implementation of the leading algorithms
 - Tramo-Seats, X12-ARIMA...
- Flexible design
 - Easier modifications of the core engines
 - Developments of additional tools/algorithms
- Challenge
 - Keeping
 - similar results
 - high performances
 - with
 - flexible (more general) design and algorithms
 - slower technical solution

What is JD+ (I) ?

Advanced Java toolkit for time series
(SA) processing (IT-teams, researchers)

Rich graphical application (end-users)



```
Java - JD+_ISI/src/isi/App.java - Eclipse Platform
File Edit Source Refactor Navigate Search Project Run Window Help

Package Explorer Hierarchy
JD+_ISI
src
  isi
    App.java
    Isi.signal
    Burman.java
    Kalman.java
    McElroy.java
  JRE System Library [jdk1.6.0_16]
  Referenced Libraries
    demetra-toolkit-1.3.0.jar - D:\JD+_ISI\src\isi\demetra-toolkit-1.3.0.jar
    ec.benchmarking
    ec.benchmarking.cholette
    ec.benchmarking.denton
    ec.benchmarking.simplots
    ec.benchmarking.ssf
    ec.benchmarking.ssf.multivariate
    ec.benchmarking.ssf.nonlinear
    ec.businesscycle
    ec.businesscycle.impl
    ec.satoolkit
    ec.satoolkit.algorithm.implementation
    ec.satoolkit.benchmarking
    ec.satoolkit.diagnostics
    AutoRegressiveSpectrumTest.d
    CombinedSeasonalityTest.class
    FriedmanTest.class
    FTest.class
    IBTest.class
    Item.class
    KruskalWallisTest.class

App.java
public static TsData series(int n) {
    DataBlock data = new DataBlock(n);
    data.randomize();
    TsPeriod start = new TsPeriod(TsFrequency.Monthly, 1960, 0);
    return new TsData(start, data);
}

public static UcarimaModel canonical() {
    SarimaModel sarima = new ec.tstoolkit.sarima.SarimaModelBuilder()
        .createAirlineModel(12, -.6, -.4);
    TrendCycleSelector tsel = new TrendCycleSelector();
    SeasonalSelector ssel = new SeasonalSelector(sarima.getSpecification()
        .getFrequency());

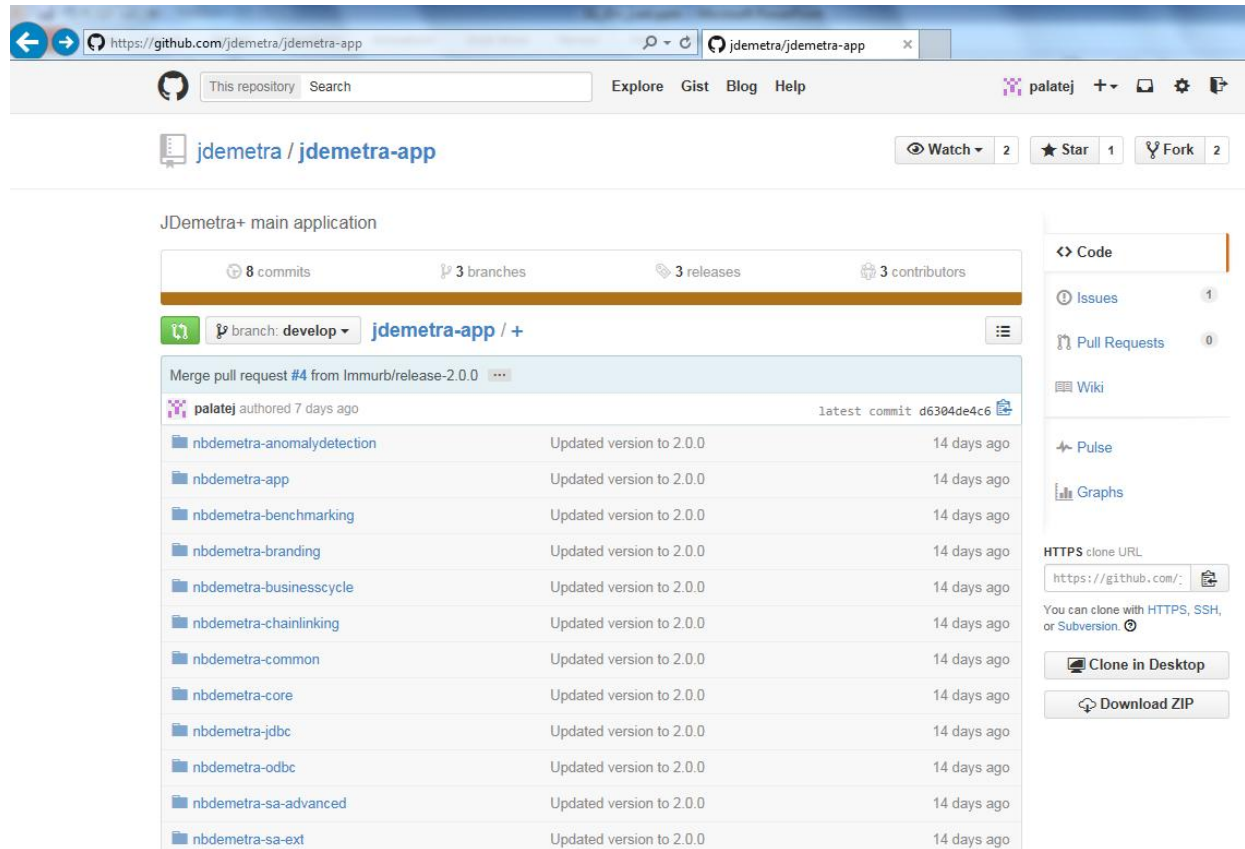
    ModelDecomposer decomposer = new ModelDecomposer();
    decomposer.add(tsel);
    decomposer.add(ssel);

    UcarimaModel ucm = decomposer.decompose(ArimaModel.create(sarima));
    ucm.setVarianceMax(-1);
    ucm.simplify();
    return ucm;
}

public static TsData saMcElroy(UcarimaModel m, TsData ts) {
    McElroyEstimates mc = new McElroyEstimates();
    mc.setUcarimaModel(m);
    mc.setData(ts);
    double[] s = mc.getComponent(1);
    TsData seas = new TsData(ts.getStart(), s, false);
    return ts.minus(seas);
}
```

What is JD+ ?

(II)



Open Source project
(EUPL license)

- Supported by Eurostat

- Developers:

- NBB

- Bundesbank

- ...

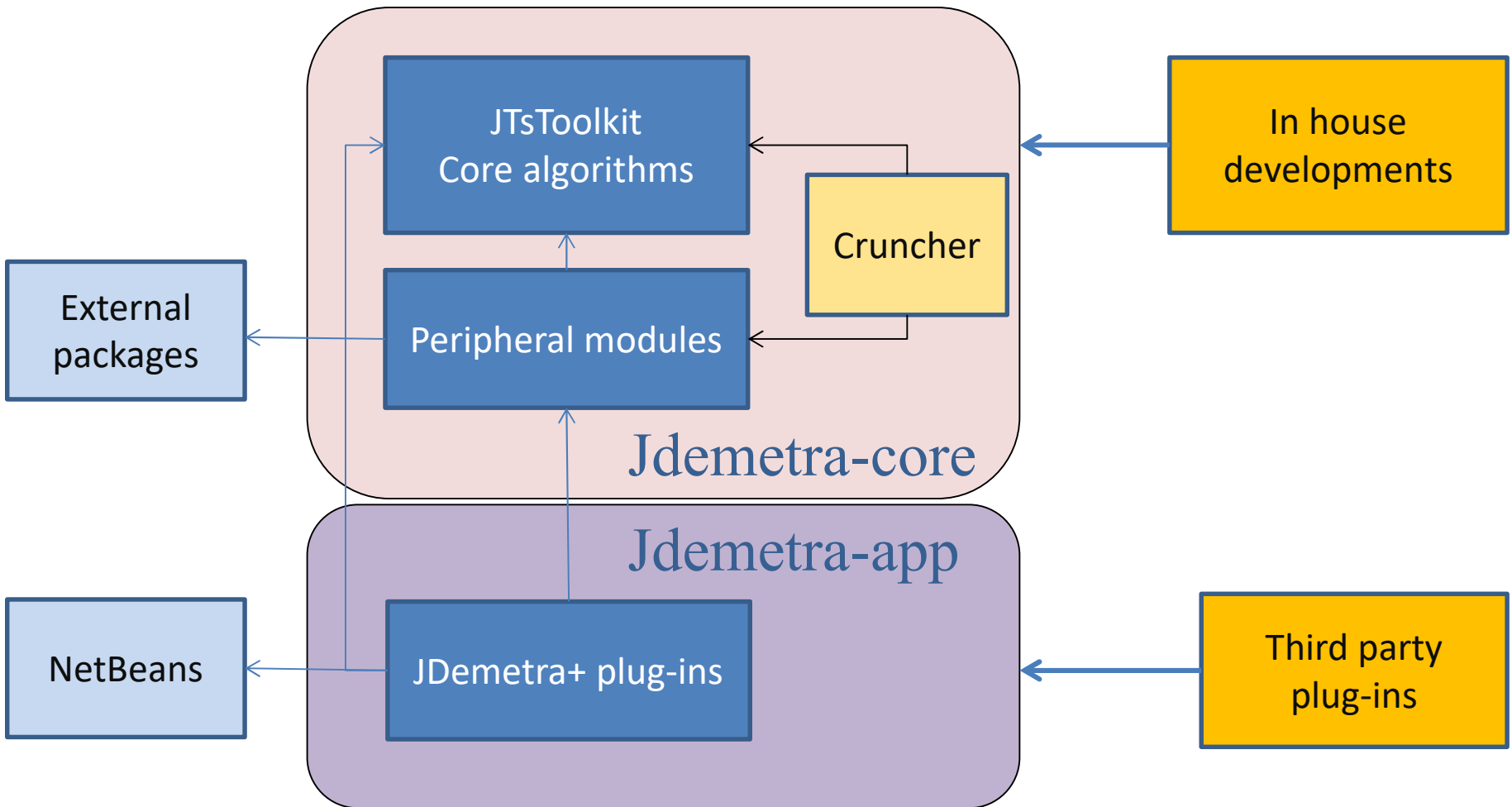
- Originally based on:

- Tramo-Seats
(BDE)

- X12-Arima
(USCB)

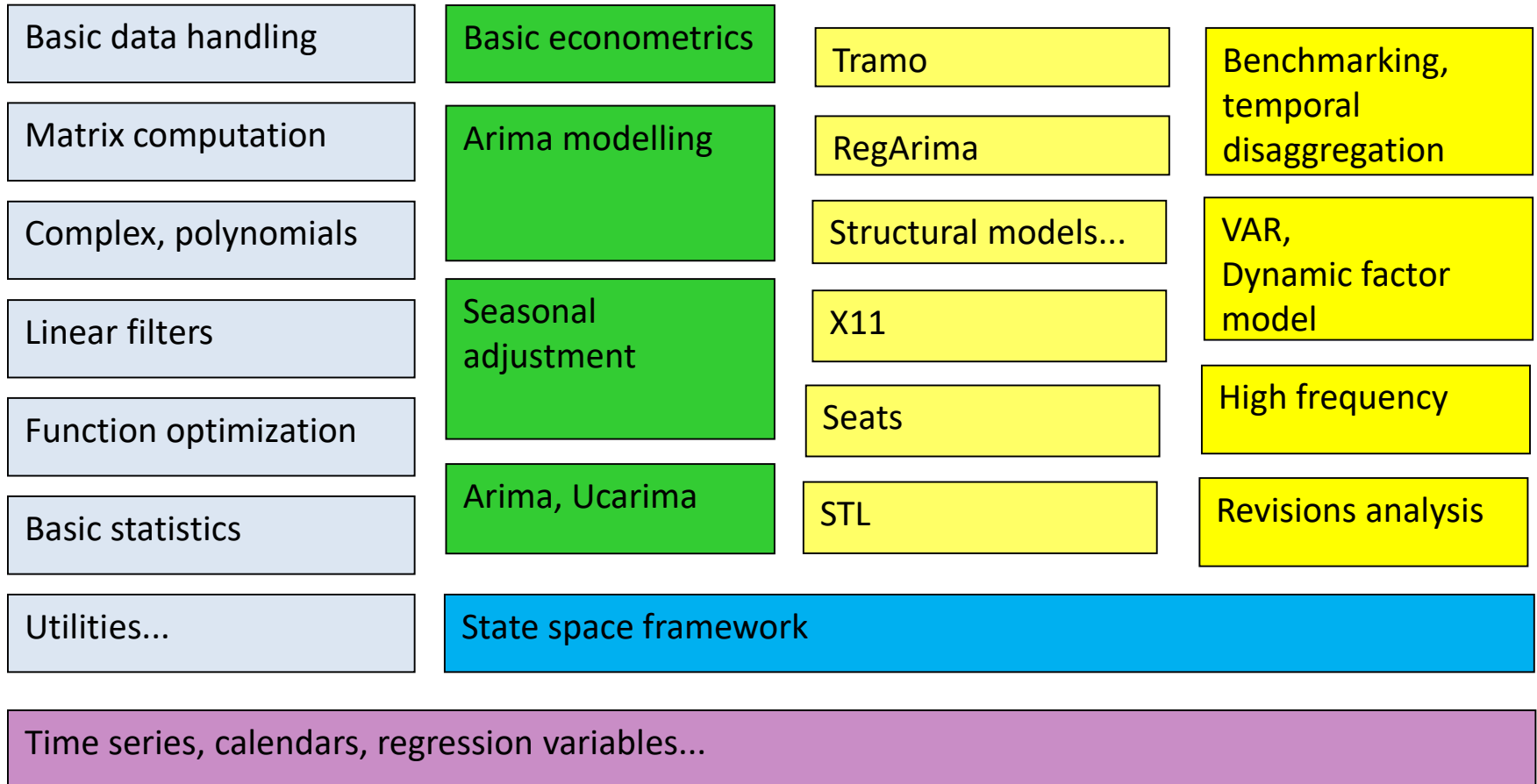
<https://github.com/jdemetra>

Architecture (I)

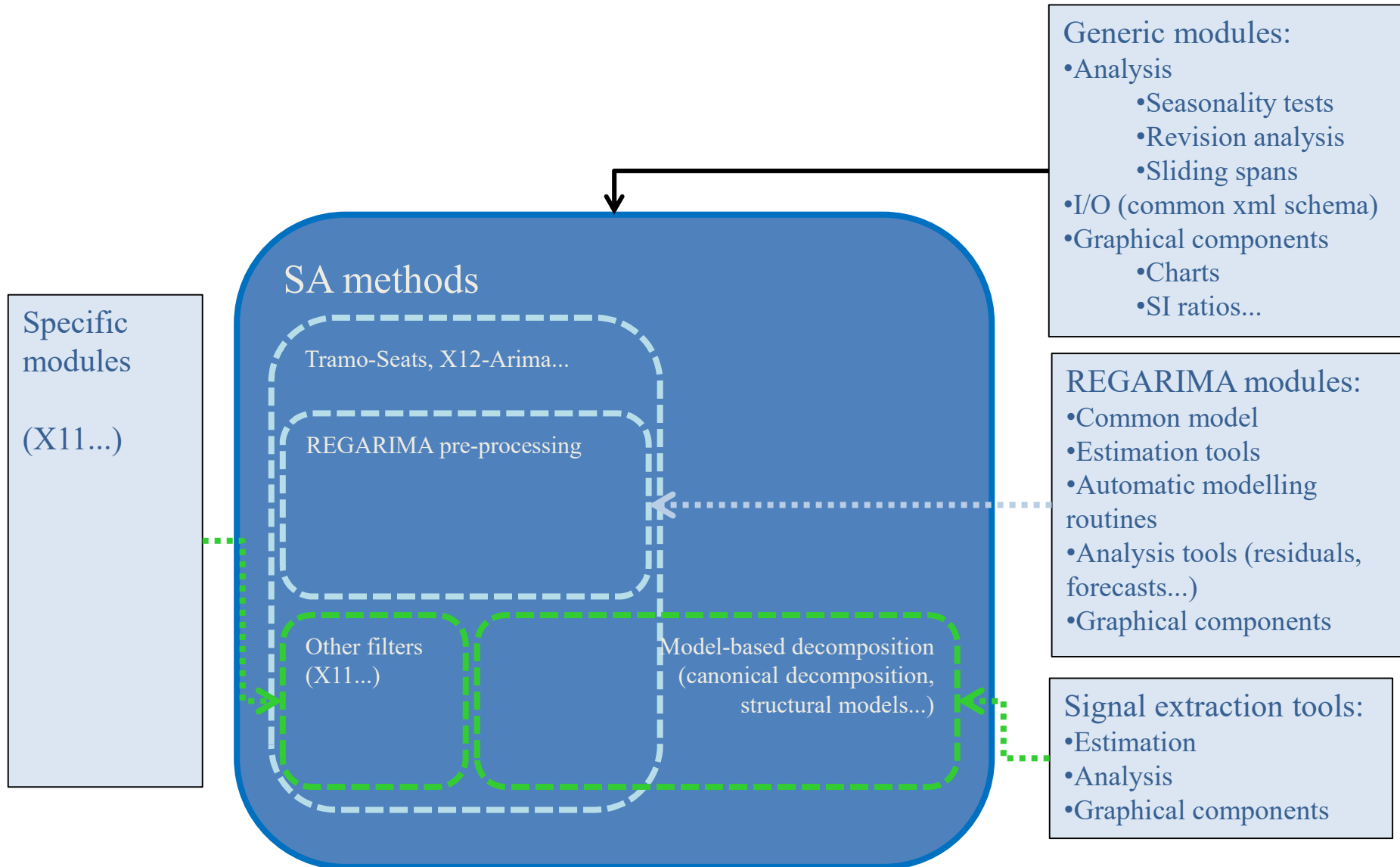


Architecture (II).

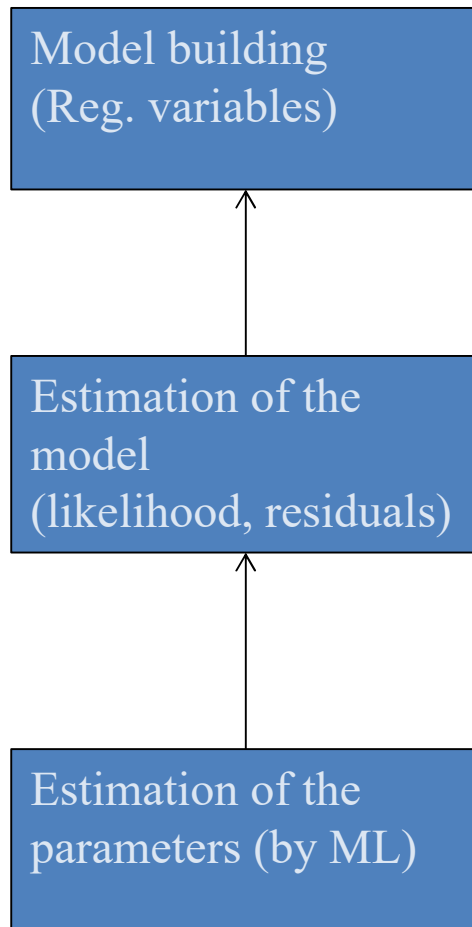
Algorithmic libraries



Seasonal adjustment framework



REGARIMA modelling



- Common definitions for Calendar variables, outliers, intervention variables, user variables...
- Algorithms for likelihood estimation
 - Kalman filter (Tramo-like),
 - Ansley algorithm (Cholesky on banded matrix)
 - (modified) Ljung-Box algorithm (X12-like)
- Equivalent results, different performances
- JD+ uses Kalman filter
 - Up to 4 x faster than Ljung-Box
 - Ansley in specific cases (outliers detection)
- Optimization procedure
 - Levenberg-Marquardt. Tramo-Seats, X12 and JD+ use slightly different variants.

Final remarks

- JD+ is a complete re-factoring of Tramo-Seats and of X12-Arima in an open OO framework. In some cases, the new algorithms may lead to (usually slightly) different results .
- JD+ is also designed for the handling of related time series problems, especially through a rich state space library.
- By developing it as an open source solution, we have tried to create an environment appropriate to external collaborations.