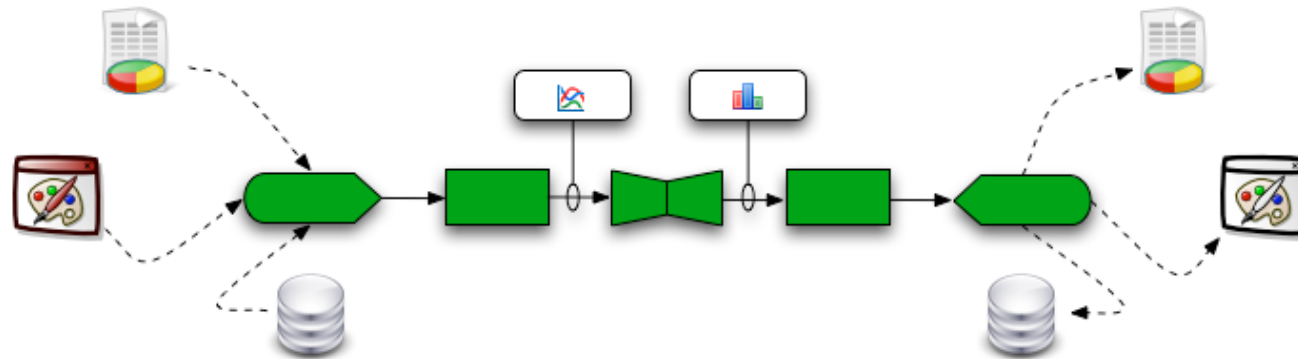


introducing the

Java Data Processing Framework



Paolo Ciccarese, PhD

On behalf of the JDPF Team

Pavia, December 11, 2007

Some history

- 2003 the first idea [for performing temporal abstractions]
- 2004 the pipeline-based architecture and a web-based user interface
- 2005 a rough component-based implementation
- 2006 presented in AMIA
- 2007 the OSGI-based open source implementation

The team

Paolo Ciccarese (since 2003)

Bruno Farina e Paolo Mauri (~ 16 man-months)

Ezio Caffi (Help with maven and OSGI)

Some simple tasks

- Reduce sets of pictures for publishing them online
- Clean some data erasing the outliers
- Read some financial data from the internet and perform some temporal analysis
- ...

Usage of different tools . . .

even if:

- It is usually necessary to run several tools as it is difficult to find tools that can be applied to really different problems (ex: data analysis and image processing)
- It is hard to organize multiple data processing to start automatically (batch mode)
- Some of the good tools are not running in server mode
- Some tools are not free

Or write some code . . .

even if:

- It is expensive to re-write complex algorithms
- It is difficult to reuse some already written code as a whole (\Rightarrow pipeline)
- It is difficult to foster reuse without a common framework (\Rightarrow jdpcf)

Java Data Processing Framework

- It helps you in the definition, generation and execution of standard and custom data processing procedures.
- It has been designed to be modular and extendable
- It promotes code sharing across the community
- It is open source

Pipeline



Filters: They give an output defined within the same metric space of the input (same data types for input and output). Examples are blocks embedding outliers filtering in a time series as well as noise reduction algorithm for images;



Transformers: Their output metric space is different from the input one (different data types for input and output). Examples can be a mechanism for qualitative abstraction performing quantitative data mapping into symbolic values as well as algorithms for the transformation at a different color depth of an image.

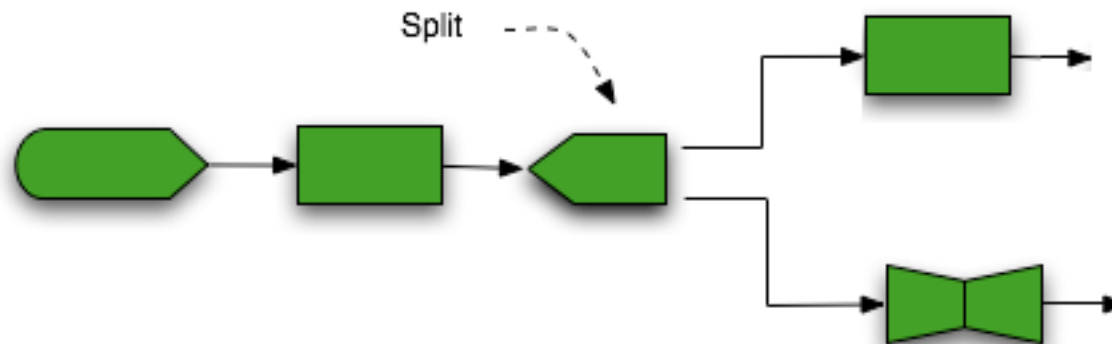
Net



Splits accept a single data stream and give two streams as output.



Aggregators accept as input the data coming from two different pipelines and give as output a single data stream derived from the application of an operator tuned, as usual, through a set of parameters



Component



Is a pipeline/net with:



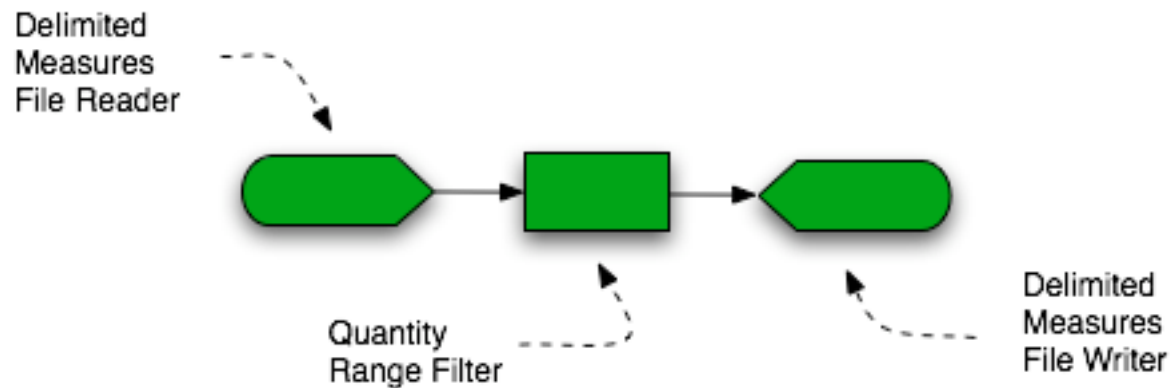
Generators provide the input to the pipeline out of binary data (xml, images, text, html, zip) or other kind of sources such as specific tables in a relational database;



Serializers represent the end of the pipeline and transform the pipeline content into data in different binary formats (xml, images, text, html, zip, pdf) or into tables of a relational database;

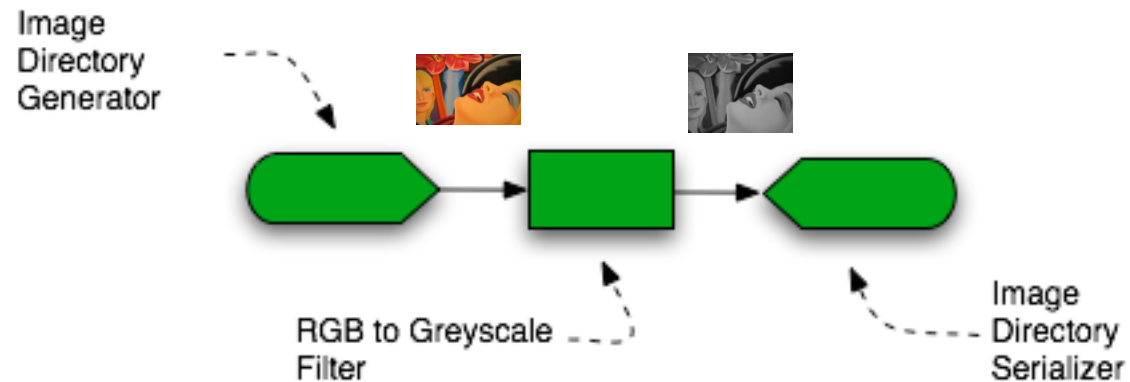
Example (1)

Reading a file of numeric data and filter outliers (with or without clipping) taking care of unit of measures



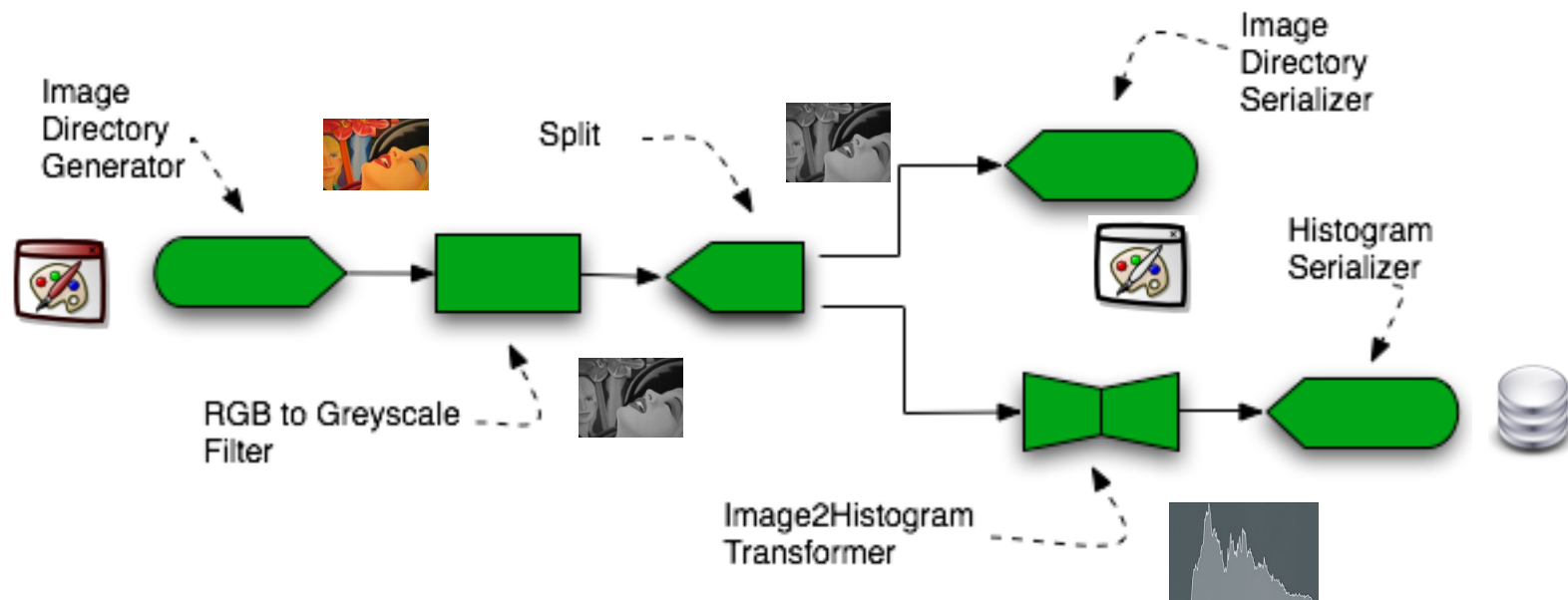
Example (2)

Transformation of one or more color images into grayscale

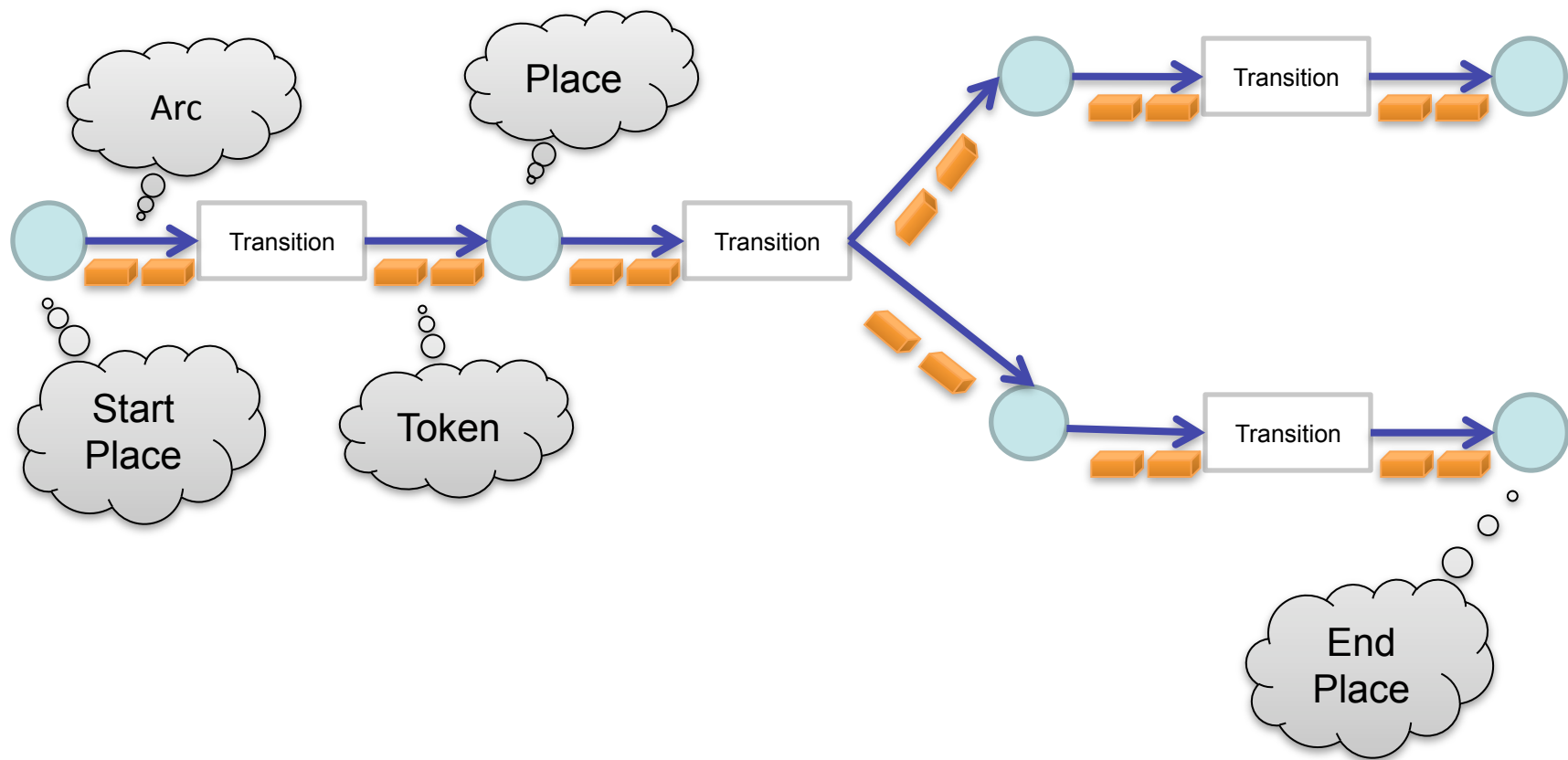


Example (3)

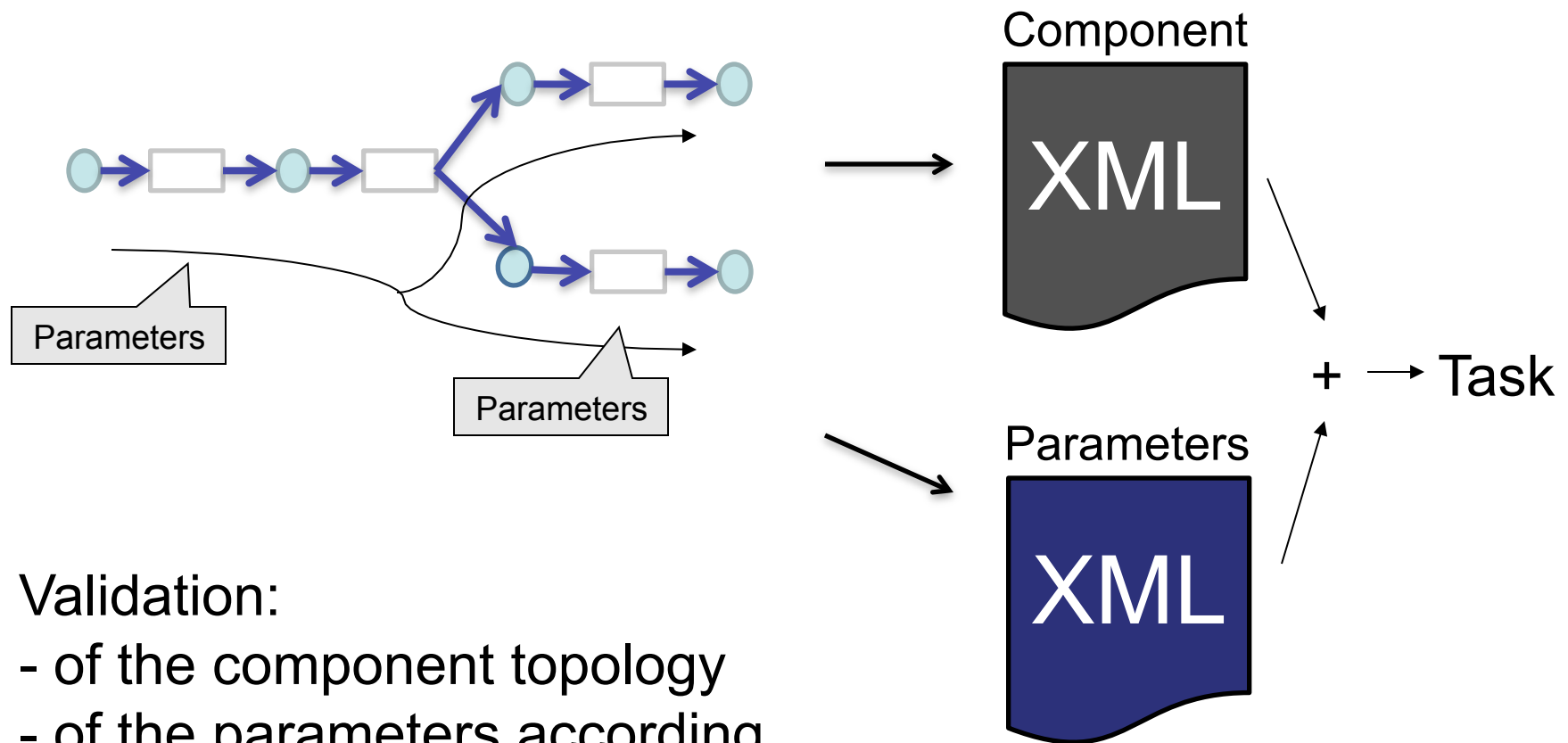
Transformation of one or more color images into grayscale and generation of images histograms



Petri-Nets architecture



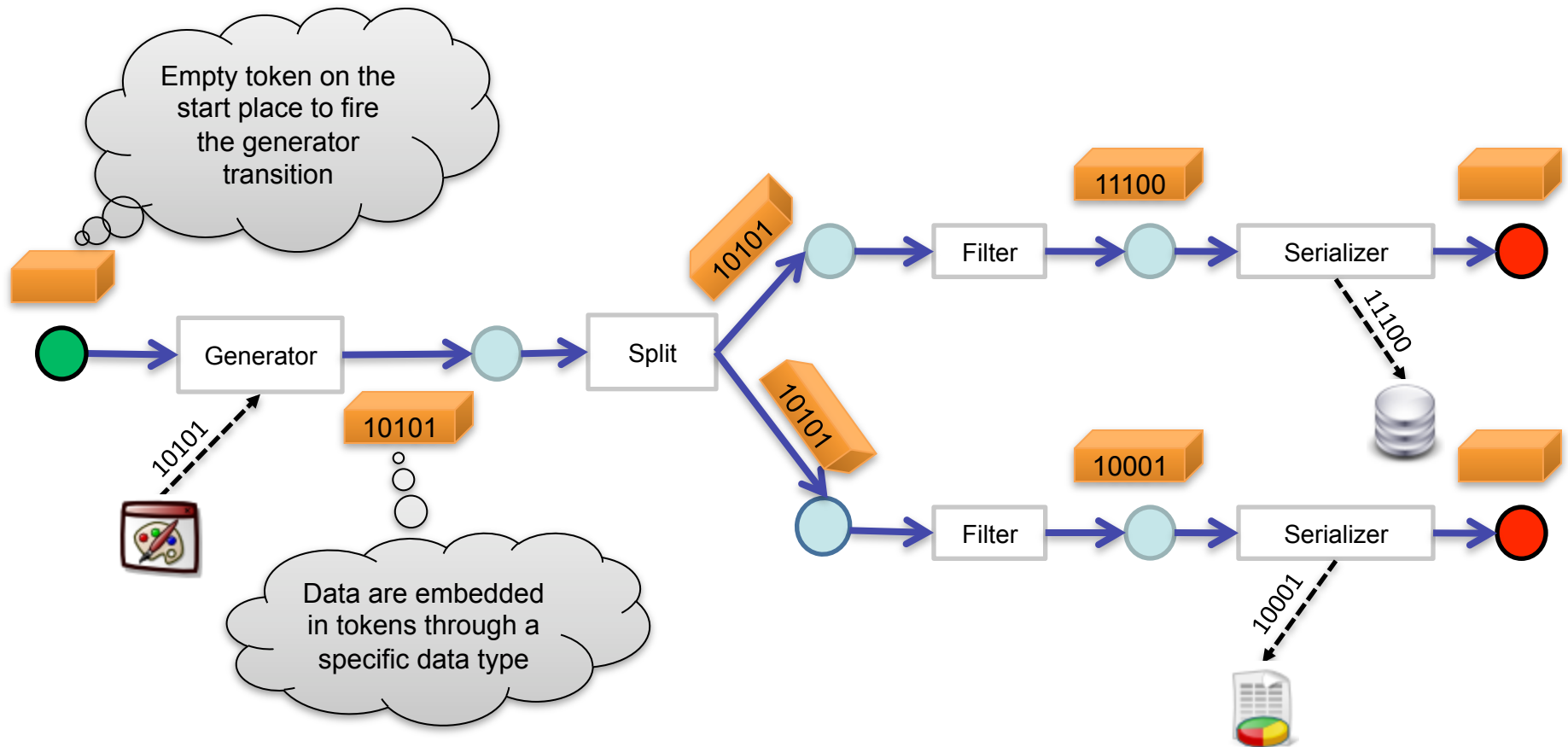
Component and Parameterization



Validation:

- of the component topology
- of the parameters according to the component

Component Execution



Summarizing

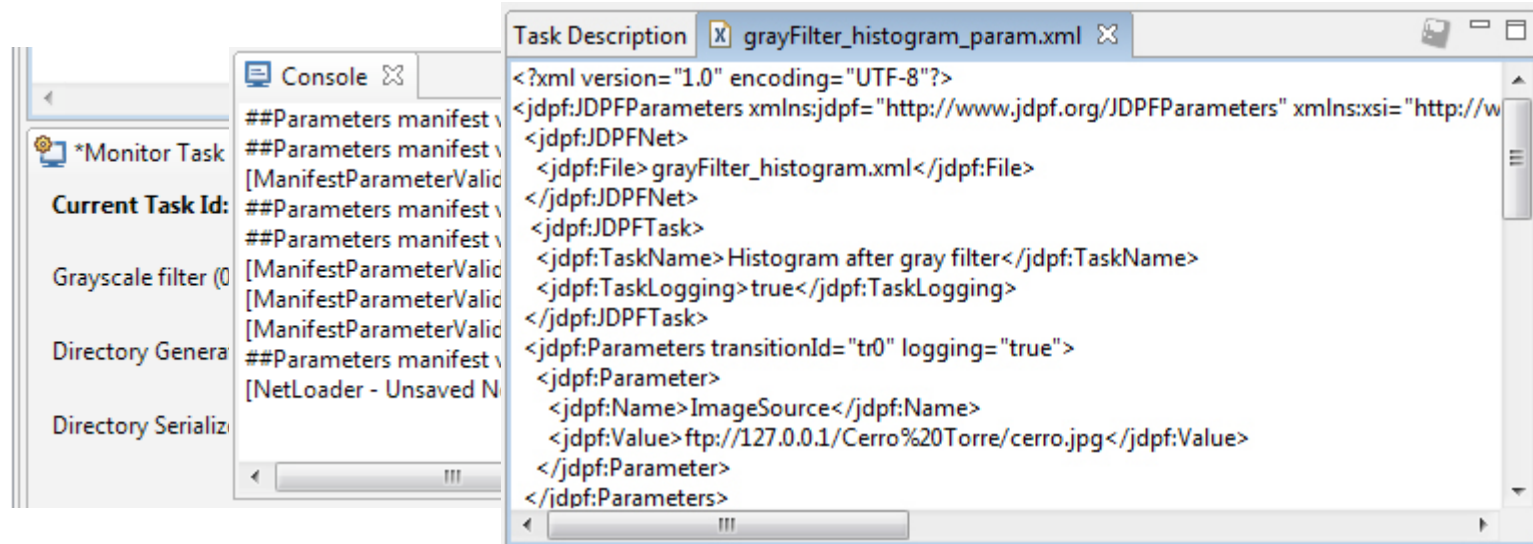
- Every block in the component implements an algorithm that works on data
- The user can reuse blocks available in the library or develop his own solution (extending the library) focusing on the algorithm implementation

JDPF *customers*

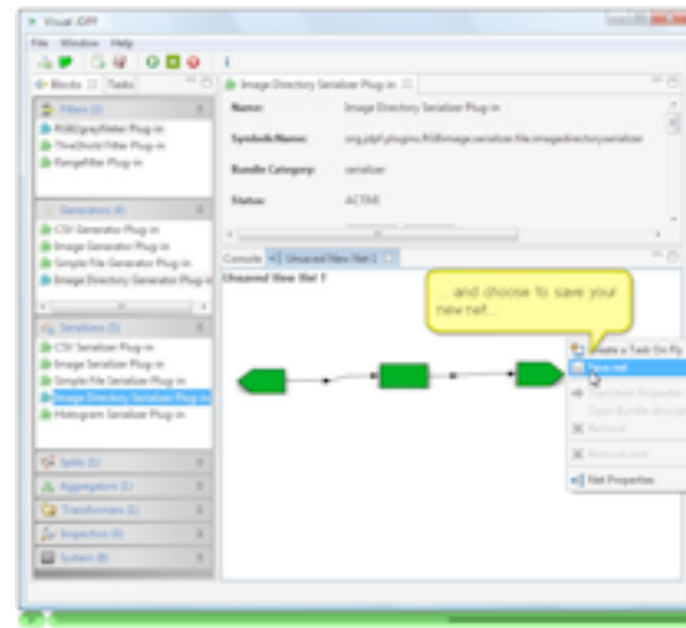
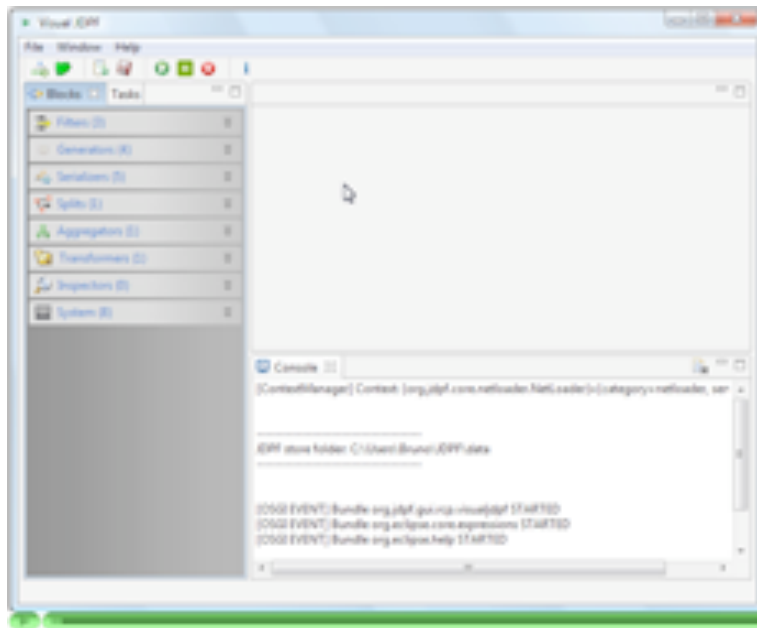
- Users that simply want to use what is available without any coding effort and through the visual tools
- Developers that can develop new blocks with the help of the JDPF framework which limits the scope of the effort

JDPF Visual Tool [1]

- Monitor dei task
- Console
- Editor testuale



JDPF Visual Tool [2]



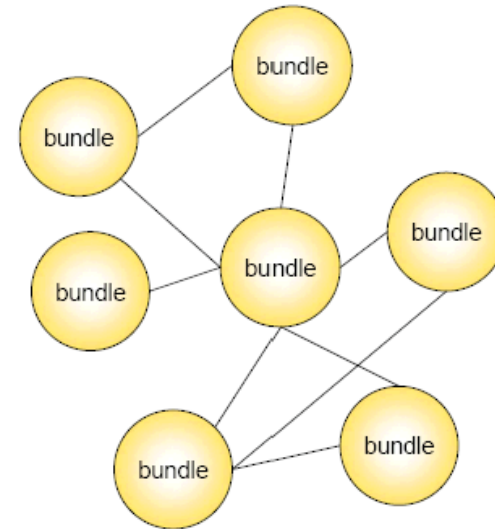
Click for watching the demos

Under the cover

- JDPF is a plugin-based architecture
- It is build on OSGi (Open Service Gateway initiative)
- OSGi technology provides a service-oriented, component-based environment for developers and offers standardized ways to manage the software lifecycle.

OSGi

- A framework for building application out of pieces of software (**bundles**) that collaborate for reaching a goal
- Bundles life cycle managed on the same JVM (security, dependencies...)



JDPF and OSGi

- Every “piece” of JDPF is a bundle
- The systems bundles are taking care of the management of the infrastructure
- The plugins bundles are the blocks embedding business logic (algorithms) and the data types

Creating a bundle . . .

- Create an OSGi bundle (we are going to provide some scripts for making this process faster, easier and safer)
- Create the bundle manifest, including:
 - dependencies
 - bundle specific parameters
 - bundle category
 - allowed input/output data types

A snapshot of a manifest

...

Bundle-Category: generator

Input-DataType: none

Output-DataType: org.jdpf.plugins.datatypes.basic.MeasureSeries

Param-TimestampFormat: string optional;

Param-DataFormat: string;{int,long,float,double}

Param-UnitOfMeasure: string optional;

Param-InputFile: uri

Param-TokensSeparator: string optional;

Algorithm implementation

- Now has to be written in Java (but we want to be able to run modules written in other languages)
- The new block has to extend an abstract implementation of the desired type (it provides logging as well as other useful services that don't have to be re-written)
- Requires to implement the setters for all the parameters defined in the manifest (setParameter(String type)...)

Code Example

```
private URI _inputFile;

public Token process(List<Token> list) throws PluginException{
    MeasureSeries iSeries = new MeasureSeries();
    ...
    algorithm implementation
    ...
    return setOutputToken(iSeries);
}

public void setInputFile(URI fileName) {
    _inputFile = fileName;
}
```

Thank you

- To the team and the audience
- Published on 1st December 2007
- Distributed with Apache Software License v. 2.0
- Collaborations...

www.jdopf.org