



**iSencia**

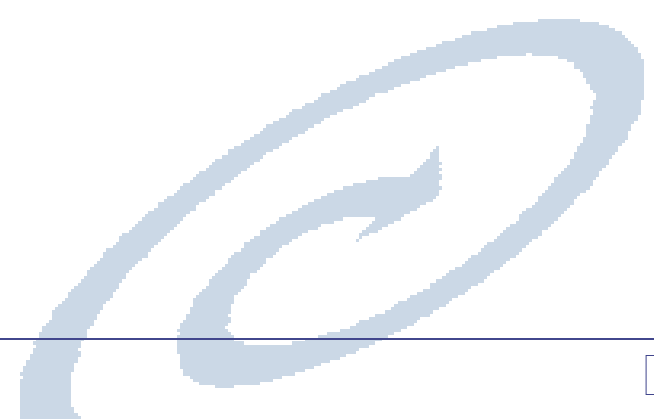
EMPOWERING BUSINESS INFORMATION

# iSencia, Passerelle & EDNA

[www.isencia.com](http://www.isencia.com)



- ▶ Who's iSencia?
- ▶ Software offerings
- ▶ Some Passerelle use cases
- ▶ EDNA and Passerelle



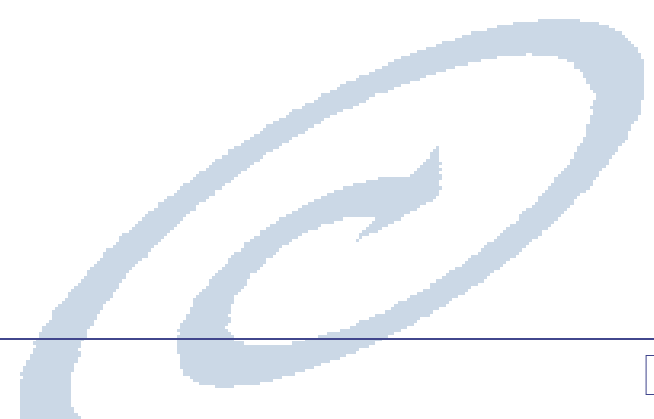
# Who's iSencia?

- ▶ Software solutions provider with offices in Ghent (Belgium) and Barcelona
- ▶ We build & maintain integrated solutions based on Java open source frameworks enhanced with own additions & improvements
  - ▶ Advanced web/intranet solutions
  - ▶ Middle tier, messaging & process engines
- ▶ Customers in different domains  
telecom, health-care, non-profit, science, logistics, ...



## SherpaBeans

- ▶ modular, services-based Java-centric web framework
- ▶ avoids explosion of XML-configuration-files
- ▶ rich web UI with desktop-like components and interactivity
- ▶ integration and many enhancements of J2EE standards, Hibernate, Wicket, OSGi, ...



# SherpaBeans samples

DaCapo

version @version@

Load

### Inventory Turnover

Article:  Turnover:  <=

Din Code:

View | 1/45 | Lines: 20

| Article                                  | Cost Sold Articles | Average inventory cost | Turnover |
|--|--------------------|------------------------|----------|
| <input type="checkbox"/> R0600,230050Z   | 0.00               | 0.00                   | 0.00     |
| <input type="checkbox"/> PWVTGK50120 GZ  | 0.00               | 0.00                   | 0.00     |
| <input type="checkbox"/> MWVZVG40016 Z   | 0.00               | 0.00                   | 0.00     |
| <input type="checkbox"/> MWCTVK30012 A2  | 0.00               | 0.00                   | 0.00     |
| <input type="checkbox"/> 7504P,39025 A2  | 0.00               | 0.00                   | 0.00     |
| <input type="checkbox"/> 0GGPF,39030 FOS | 0.00               | 0.00                   | 0.00     |
| <input type="checkbox"/> 07991,12130 Z-  | 0.00               | 0.00                   | 0.00     |
| <input type="checkbox"/> 07991,08090 A2  | 0.00               | 0.00                   | 0.00     |
| <input type="checkbox"/> 07981,42045 A2  | 0.00               | 0.00                   | 0.00     |
| <input type="checkbox"/> 00965,06020 Z   | 0.00               | 0.00                   | 0.00     |
| <input type="checkbox"/> 00933,04016 Z   | 0.00               | 0.00                   | 0.00     |
| <input type="checkbox"/> 00603,08030 A2  | 0.00               | 0.00                   | 0.00     |
| <input type="checkbox"/> 00571,10280 Z   | 0.00               | 0.00                   | 0.00     |
| <input type="checkbox"/> 00073,05005 TA  | 0.00               | 0.00                   | 0.00     |
| <input type="checkbox"/> ZBGSR,63125 Z   | 56,550.00          | 56,550.00              | 1.00     |
| <input type="checkbox"/> ZBG16,63125 Z   | 81,720.00          | 81,720.00              | 1.00     |
| <input type="checkbox"/> ZBG16,63090 Z   | 4,628.00           | 4,628.00               | 1.00     |
| <input type="checkbox"/> ZRG16.63075 Z   | 54.070.00          | 54.070.00              | 1.00     |



# SherpaBeans samples - eSengo

eSengo Organizer 20080908 (Dirk Jacobs) - Mozilla Firefox

http://www.isencia.com/esengo/bridge/app/

**eSengo organizer** Domain: iSencia Belgium Home | Preferences | About | Sign Out

New Process | Organizer | Contacts | Configuration | Administration

**Calendar**

September 2008

| wk | mon | tue | wed | thu | fri | sat | sun |
|----|-----|-----|-----|-----|-----|-----|-----|
| 36 | 1   | 2   | 3   | 4   | 5   | 6   | 7   |
| 37 | 8   | 9   | 10  | 11  | 12  | 13  | 14  |
| 38 | 15  | 16  | 17  | 18  | 19  | 20  | 21  |
| 39 | 22  | 23  | 24  | 25  | 26  | 27  | 28  |
| 40 | 29  | 30  |     |     |     |     |     |

**Events**

+ New Event

**Inbox**

Inbox Projects All + New Folder

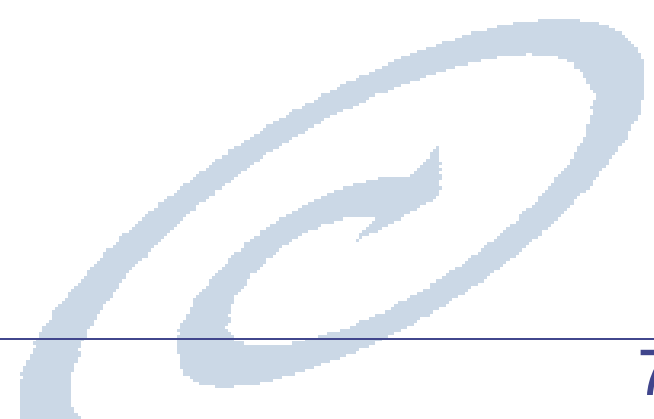
Activity:  Search

1/1 10

| Activity  | Type      | Pr | Assign Date | Due Date   |
|---|-----------|----|-------------|------------|
| Later   |           |    |             |            |
| allow to set the codepage to use for exporting to txt format (Continuo)     | Execution | -  | 02-09-2008  | 01-10-2008 |
| deleting an attachment in the CMS system results in a white page (Continuo) | Execution | -  | 02-09-2008  | 01-10-2008 |
| Report for open demand (Continuo)   | Execution | -  | 02-09-2008  | 01-10-2008 |

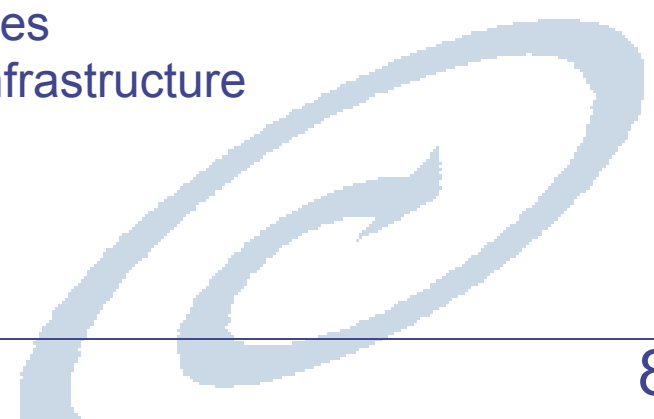
## Passerelle

- ▶ Solutions suite for model-based solutions assembly
- ▶ Graphical model defines flow and execution semantics
- ▶ Based on picking, configuring and interconnecting reusable components (actors) from a library
- ▶ Support different roles in solution delivery & maintenance (designers, developers, users, administrators)
- ▶ Tools and infrastructure for
  - ▶ Maintaining component libraries
  - ▶ Assembling solutions
  - ▶ Testing & deployment



## Some important concepts

- ▶ Flows / processes /sequences  
Support for all typical flow concepts (parallel branches, loops, ...)
- ▶ Actors with multiple named input and output ports  
Each port has specific, identifiable function  
Actors have own execution thread
- ▶ Standardized internal messages  
System headers, App headers, App data  
Supporting wide variety of data content types  
Automated data content type conversion infrastructure





# Passerelle

The screenshot shows the Passerelle IDE 4.1 interface. On the left is the Explorer showing a project structure with folders like 'Soleil Actors' and 'Ptolemy Actors'. The main window displays a sequence diagram for a test sequence. The diagram consists of several actors: RequestSource (yellow circle with a clock), ParameterSetter (grey circle with a clock), Backend Requestor (blue triangle), TimeOutTimer (clock icon), Results dumper (blue triangle), and ResultVerification (grey square with a checkmark). The flow starts with RequestSource sending 'requestFinish' to ParameterSetter. ParameterSetter then sends 'requestFinish' to Backend Requestor. Backend Requestor sends 'requestFinish' to TimeOutTimer and 'resultOnlyOutput' to ResultVerification. TimeOutTimer sends 'requestFinish' to Results dumper. Results dumper sends 'requestFinish' to ResultVerification. ResultVerification sends 'okOutput' and 'errorOutput' to the final output port. A 'Passerelle Std' logo is present in the diagram area.

keleton for a test sequence for new backend adapters, implemented as sequence components (a.k.a. "actors").

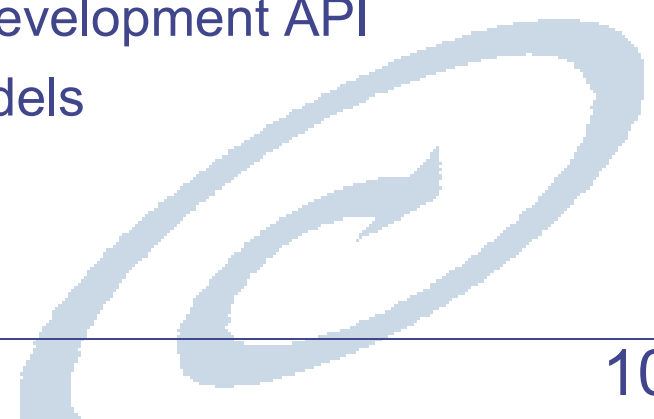
est requests are injected via the RequestSource. These requests contain the normal data defined by DAC client-app  
he ParameterSetter adds extra data, as would be determined in a fully operational DAC from other backend calls, but  
he Backend Requestor finds its required input data from the incoming request-with-extra-parameters,  
orms the call to the backend, stores the results in the DAC DB, and sends the enhanced request message and the  
ugh its output ports.

here possible, a ResultVerification algorithm will be automated.  
hen this is not present, the test result must be evaluated manually, using the DARE Workbench or another result vis

parallel with the actual backend request, a TimeOut Timer is activated. When the backend result is not obtained within  
is also reported as an error.

## Benefits

- ▶ Designed for reuse
  - ▶ For solution assemblers (graphical solution assembly)
  - ▶ For component developers (actor dvp framework)
- ▶ Pipe-and-filter architecture
  - ▶ Promotes clear definition of component responsibilities
  - ▶ Promotes low coupling of component implementations
- ▶ Designed for extensibility
  - ▶ Well-defined, simple component development API
  - ▶ Ready for different application models



## Benefits

- ▶ Take advantage of standard “good software engineering” architectural ideas (cfr above) and practices
  - ▶ Several test levels (unit test, system test, ...)
  - ▶ Take advantage of test tools available in Java domain (JUnit, Mock tests, continuous build & test, Ant/Maven build scripting, test reporting tools & build servers,...)
  - ▶ Version mgmt of workflows
- ▶ Support for “auto-analysis” of executing workflows
  - ▶ Integration of rules engine component
- ▶ Workflow dynamic routing per request i.o. through “static” cfg changes

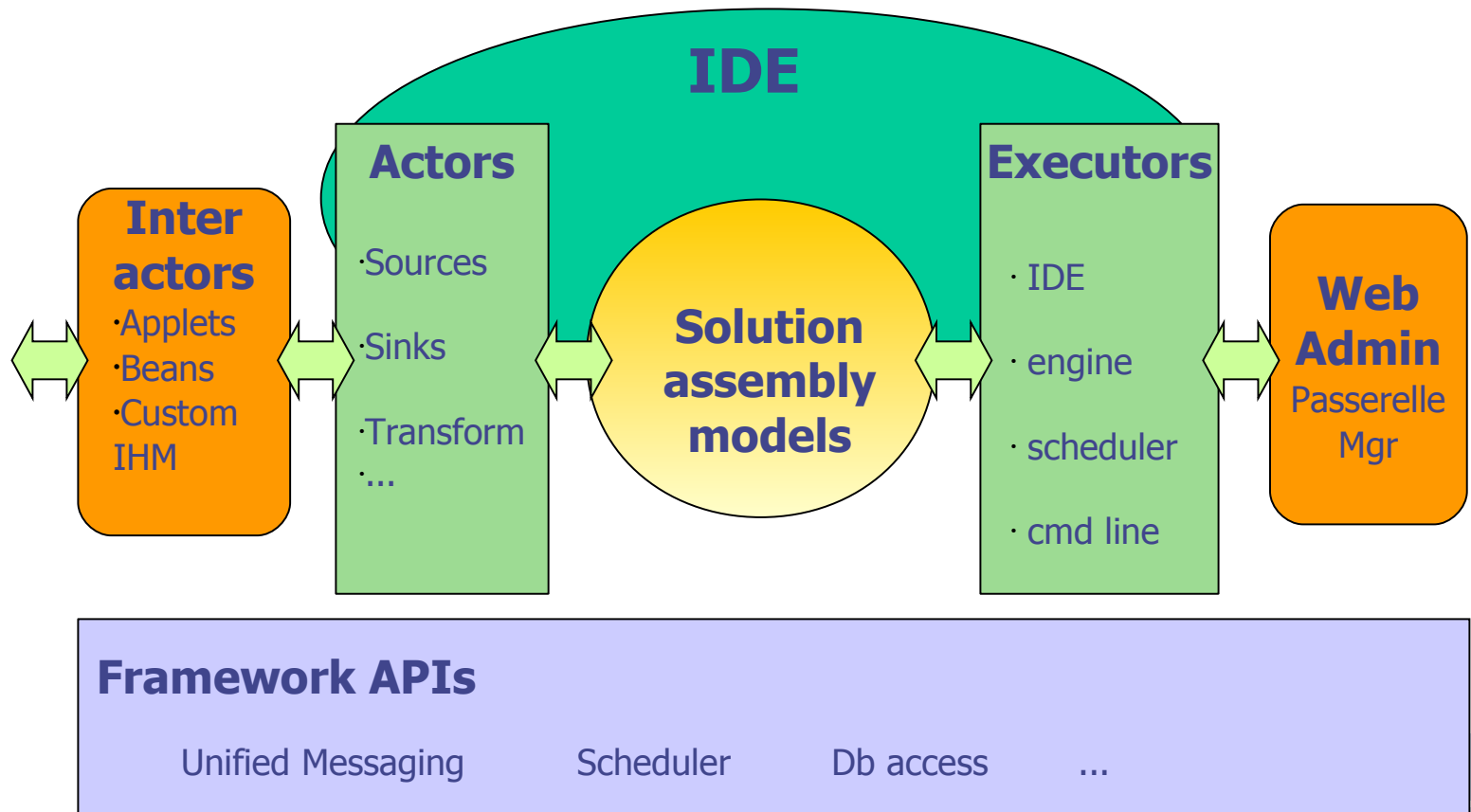


## Actor-based solutions

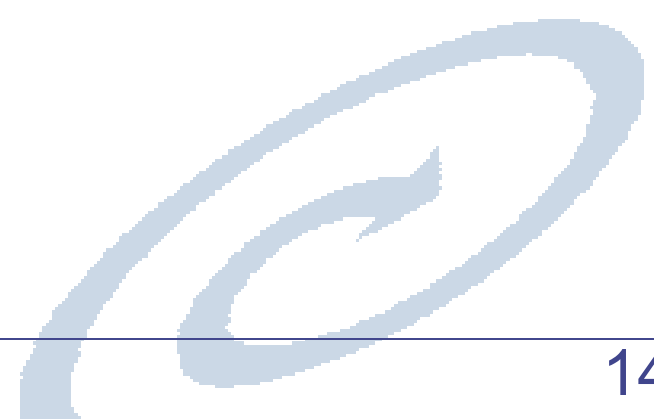
- ▶ Based on actor-based paradigm of Ptolemy project
- ▶ Enriched with :
  - ▶ Usage & admin Tools : customized IDE, integration with Quartz scheduler, server run-time engine with DB persistence web admin UI, web services, ...
  - ▶ Dvp Tools : improved actor development API, standardized internal messaging model, automated data type conversions, improved error handling, run-time engine extensions, ...



# Passerelle suite



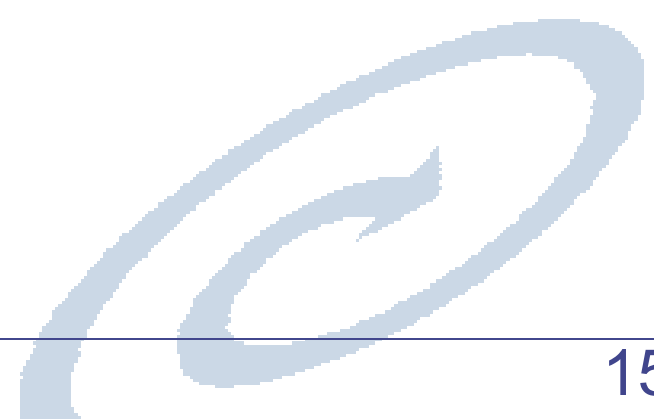
- ▶ Who's iSencia?
- ▶ Software offerings
- ▶ **Some Passerelle use cases**
- ▶ EDNA and Passerelle



# Some Passerelle Use Cases

## Light-weight communication engine

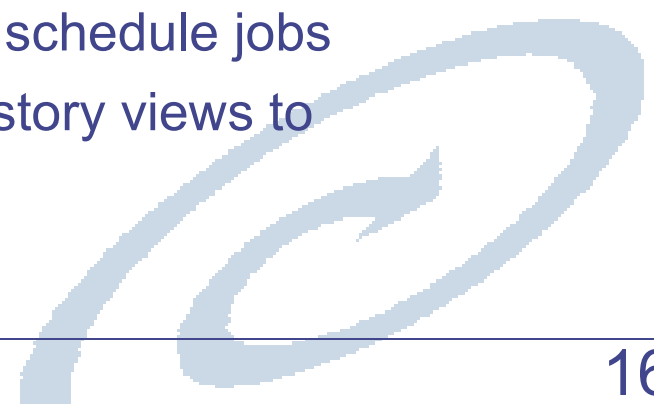
- ▶ Reuse of Passerelle UME
- ▶ Select components/libs matching communication needs
- ▶ Assemble communication layer once, embedded in app
- ▶ Passerelle completely hidden for end-users



# Some Passerelle Use Cases

## Batch job management

- ▶ Scheduling “expensive” actions out-of-business-hours (automated reporting, data replication & cleanup,...)
- ▶ Small assemblies, auto-terminating, limited number of executions per day
- ▶ Simple actors (DB access, running external scripts etc)
- ▶ Simple or no input data needed in the assemblies, only configuration data
- ▶ Using the Passerelle Manager
  - ▶ Secure environment to define and schedule jobs
  - ▶ Offers integrated job status and history views to follow/check execution status





# Some Passerelle Use Cases

## Process control and data acquisition

- ▶ Complex assemblies, auto-terminating, limited number of executions per day
- ▶ Adapted device control actors, combined with results analysis and reporting
- ▶ Simple or no input data needed in the assemblies, only configuration data
- ▶ Using the Passerelle Manager
  - ▶ Offers integrated job status and history views to follow/check execution status
  - ▶ Scheduler not often needed, plain “launch now”



# Some Passerelle Use Cases

## Personal automation tool

- ▶ Implement repetitive tasks via actors  
(personal backups, mailings, simple system monitoring etc)
- ▶ Replaces explosion of shell-scripts, macros, ...
- ▶ Auto-terminating assemblies
- ▶ Simple or no input data needed in the assemblies, only configuration data
- ▶ Plain Passerelle IDE is sufficient
  - ▶ Define personal “templates” in the User Library
  - ▶ Easy configuration of the personal tasks
  - ▶ Less reliability and tracing needed



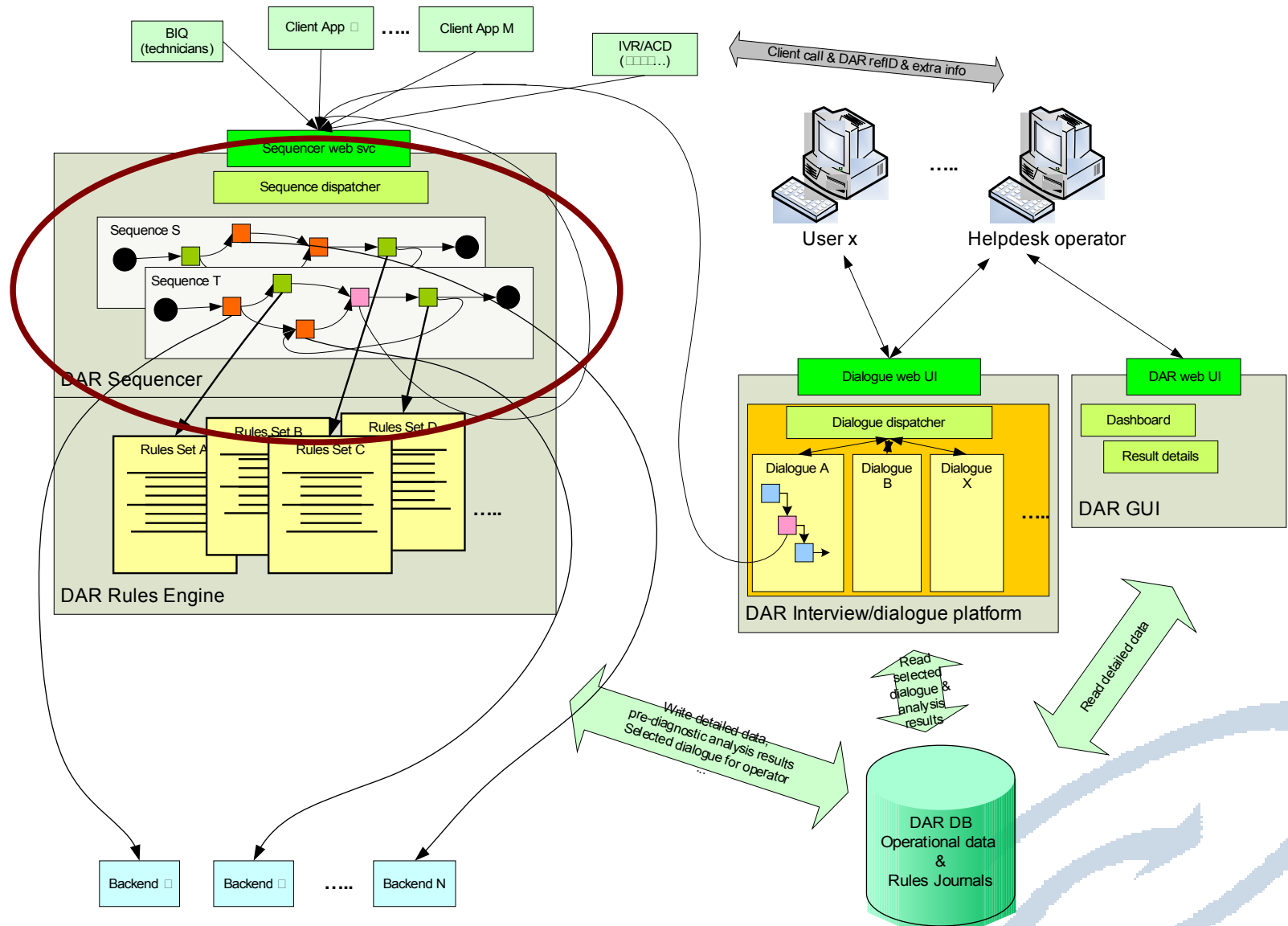
# Some Passerelle Use Cases

## Diagnostic process engine

- ▶ Full-blown middle-tier processing engine
- ▶ Each diagnostic process represented as a graphical flow
- ▶ Limited number of complex assemblies, continuously running, non-auto-terminating
- ▶ Diagnostic is triggered by input messages
- ▶ Potentially high-throughput requirements
- ▶ Combines data collection tasks from approx 50 ≠ sources with rules-based data analysis tasks in integrated sequences/processes
- ▶ Part of a complete enterprise application environment



# Diagnostic process engine

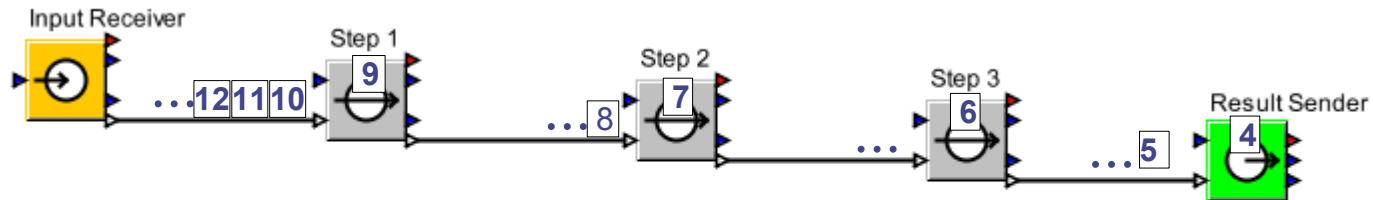


# Diagnostic process engine

- ▶ Implemented on Passerelle Manager
  - ▶ Std job status & history not sufficient, need for persistence of extra execution trace details & data and analysis results via a generic data model (analysis to promote as std product feature is ongoing)
  - ▶ Needs integration in complete enterprise software infrastructure (clustered application servers, SOA, EJBs ...)
- ▶ Takes full advantage of Ptolemy's PN : multithreaded, internal buffering etc, leading to a high-throughput solution with a staged event driven execution model (cfr. automobile production/assembly line)



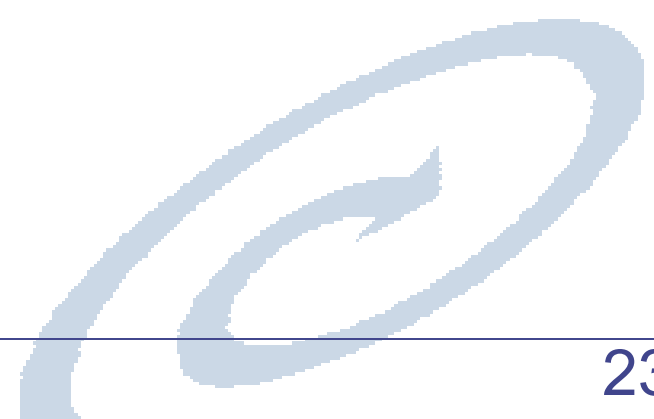
# PN domain as “assembly line”



- ▶ Consecutive messages handled concurrently in different actors
- ▶ Each actor has own processing thread(s)
- ▶ Temporary congestion is handled by internal buffering on each input port

*Leads to high-throughput with limited system resources, resilient for temporary peak loads*

- ▶ Who's iSencia?
- ▶ Software offerings
- ▶ Some Passerelle use cases
- ▶ EDNA and Passerelle



# EDNA and Passerelle

## EDNA :

*“The aim is to develop a **configurable application** able to **launch sequentially** the crystal **characterization** and the **data collection strategy steps** by executing **one or several parallel** external programs according to the user configuration”*

## Passerelle :

Graphical configuration and application assembly

Flows with support for sequential & parallel steps

Generic component model, support for automation, data collection and analysis etc





# EDNA and Passerelle

## **EDNA :**

*Application components are “plugins” with well-defined lifecycle*

*Plugins invoke a limited number of external programs*

*Main development language Python*

## **Passerelle :**

Actor-based components with similar life-cycle

Already support for simple execution of external scripts

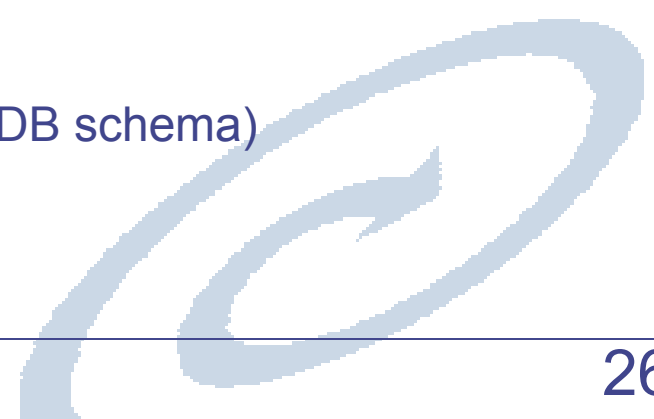
Basic Jython support



# EDNA and Passerelle

## **Passerelle discussion topics : (related to some EDNA issues discovered in docs/minutes)**

- ▶ Java-based → platform-independent
- ▶ Integrated error reporting/handling mechanisms
- ▶ Actor mock-mode or external system simulators for testing
- ▶ Several levels of persistence  
(execution traces, result data model, ...)
- ▶ Automated execution scheduling
- ▶ Administration via web application
- ▶ Support for GRID?
- ▶ Consulting result data via web UI  
(SherpaBeans app for ISPyB?  
→ no need to create HTML, UI directly on DB schema)



# EDNA and Passerelle

## Possible approaches :

### 1. Invoke EDNA plugins from Passerelle

- ▶ Develop Java wrapper API around Python Plugin API or use Jython actors
- ▶ Develop limited set of actors (1?) on API
- ▶ Exploit actor parameters for configuration (generate EDNA cfg file(s) for correct plugin functioning?)
- ▶ Least-intrusive : EDNA-Python users can continue as now



# EDNA and Passerelle

## Possible approaches :

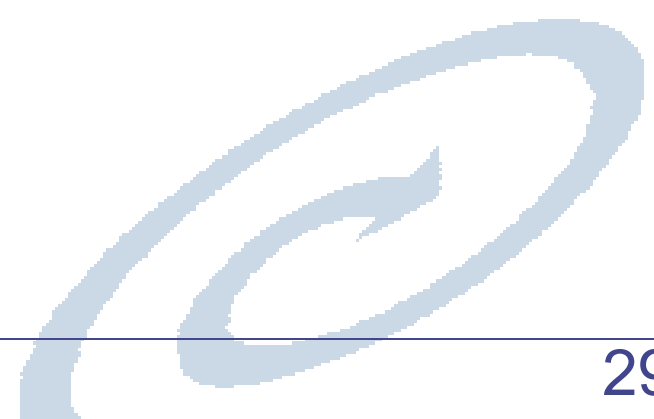
### 2. Directly invoke external programs from Passerelle

- ▶ External programs executable as shell scripts?  
or reachable via inter-process communication?
- ▶ Develop actor per program
- ▶ Exploit actor parameters for configuration  
no need to generate EDNA cfg file(s)
- ▶ Python-plugin development replaced by Java-actor  
development
- ▶ Take full advantage of std Java / Ptolemy / Passerelle features
- ▶ Large impact, both technological and psychological

# EDNA and Passerelle

## Possible approaches :

3. Mixed approach : option 1. for existing plugins, 2. for new ones
  - ▶ Valid as intermediate approach
  - ▶ Not as long-term solution → divided development and support



# EDNA and Passerelle

## Plugin ↔ Actor mapping :

setInputData : must be overridden! for casting XML into the XSDDataInput type.

checkParameters : called before config for assuring complete input data

config : called before preProcess, used for configuring the plugin

preProcess : called before process, used for setting up files etc. needed by the plugin

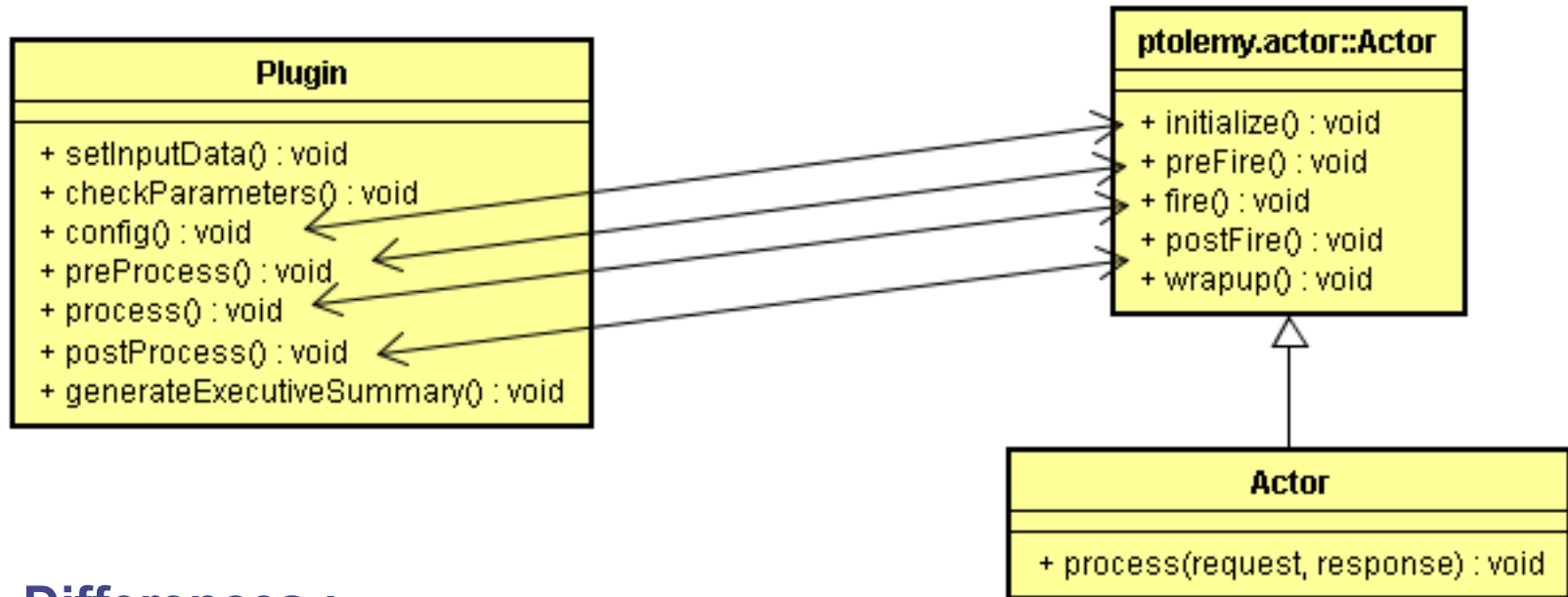
process : The main method. Does not need to be defined if derived from EDPluginExecProcess or EDPluginExecProcessScript

postProcess : called after the process, used for recuperating the data produced by e.g. external programs

generateExecutiveSummary : should write a human readable result summary after the execution of the plugin



# EDNA and Passerelle



## Differences :

- ▶ Actor executes fire-loop repetitively
- ▶ Several validation strategies can be plugged in at start of execution & during each actor fire-iteration
- ▶ `generateExecutiveSummary` could be in `postFire()`, `wrapup()` or even better in separate reporting tool accessing persisted actor results