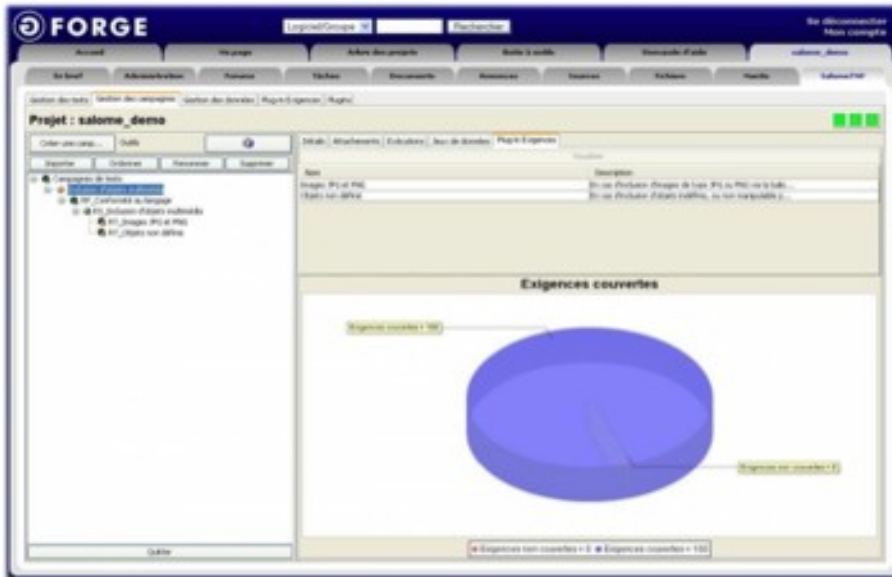


Test management and load testing with Salomé-TMF and CLIF is a Load Injection

Bruno Dillenseger
Orange Labs

OW2 Tech Day, May 15th 2008
Montbonnot (Grenoble), France



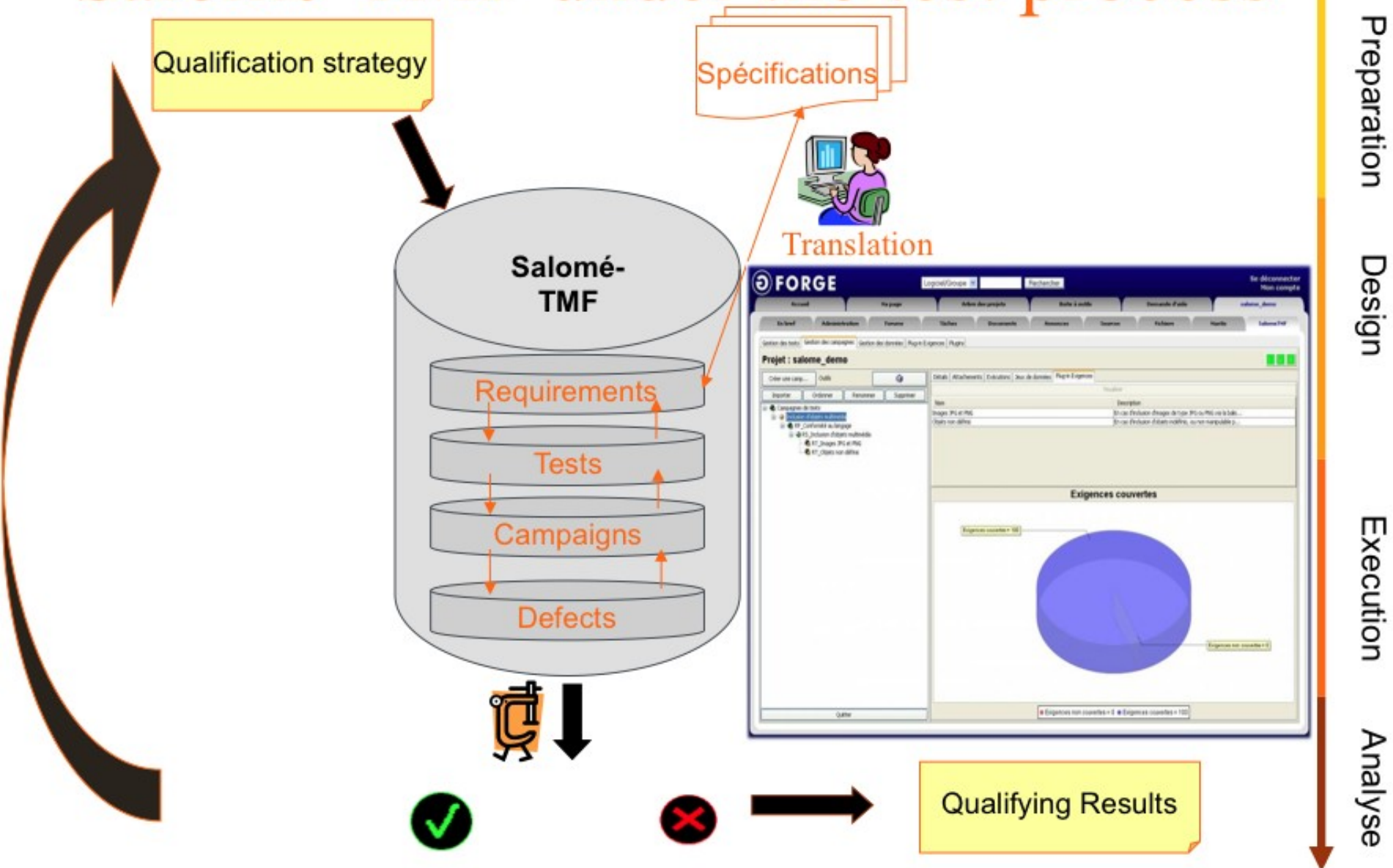


Salomé - Test management Framework

Marche Mikael, Thomas Escalle



Salomé-TMF under the test process



Key benefits of Salomé-TMF

➔ An open-source solution alternative to QualityCenter

- Cost reduction of licence
- Multi-OS (100% “Pure Java”)

➔ A simplified solution

- Management of the test plan, campaigns (simplified compared to QualityCenter)
- Requirements and defects management integrated
- Generation of documents (HTML, PDF, ...)

➔ An open solution

- Integration with existing tools :
- Mantis, Codex, Gforge, ...
- Import/Export XML : open format
- Import/Export to QualityCenter (8&9)
- Plug-ins architecture

➔ A solution for tests automation

- GUI Web (Selenium) , ...
- Java Scripted Tests (Beanshell)
- Open API (remote Ant, command line, ...)

Salomé-TMF : Roadmap for 2008

➤ Current Version 3.0

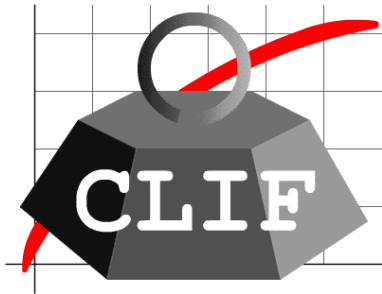
- Corrective maintenance
- Integration with Codex (Xerox Forge)

➤ Minor version 3.1 : September 2008

- Improving existing functions:
 - copy/paste , filter on requirements, and some features requests
- Implementation of test templates:
 - Template = abstract tests can define actions, and user fields
- CLIF Plug-in : defining and executing load tests from Salomé-TMF environment
- Improve internationalisation (New Language : German, Spanish)

➤ Minor version 3.2 : End 2008

- Adding the concept of a directory above the families of tests
- Extension of the SOAP interface to all plug-ins



CLIF is a Load Injection Framework

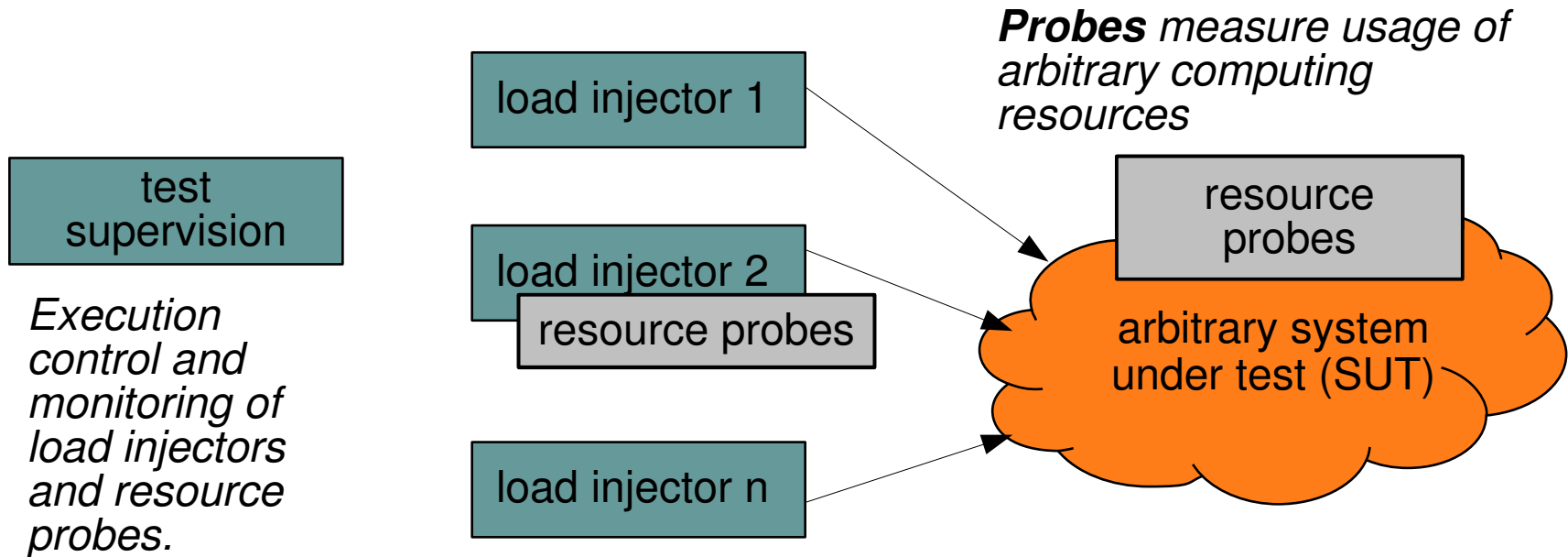
Bruno Dillenseger, Thomas Escalle
Orange Labs

clif.ow2.org / clif@ow2.org



Overview of Concepts, Principles and Architecture

Big picture of (distributed) load testing



Load injectors :

- send requests, wait for replies, measure **response times**
- according to a given **scenario**
- for example, emulating the load of a number of real users (through "**virtual users**")

CLIF is a Load Injection Framework...

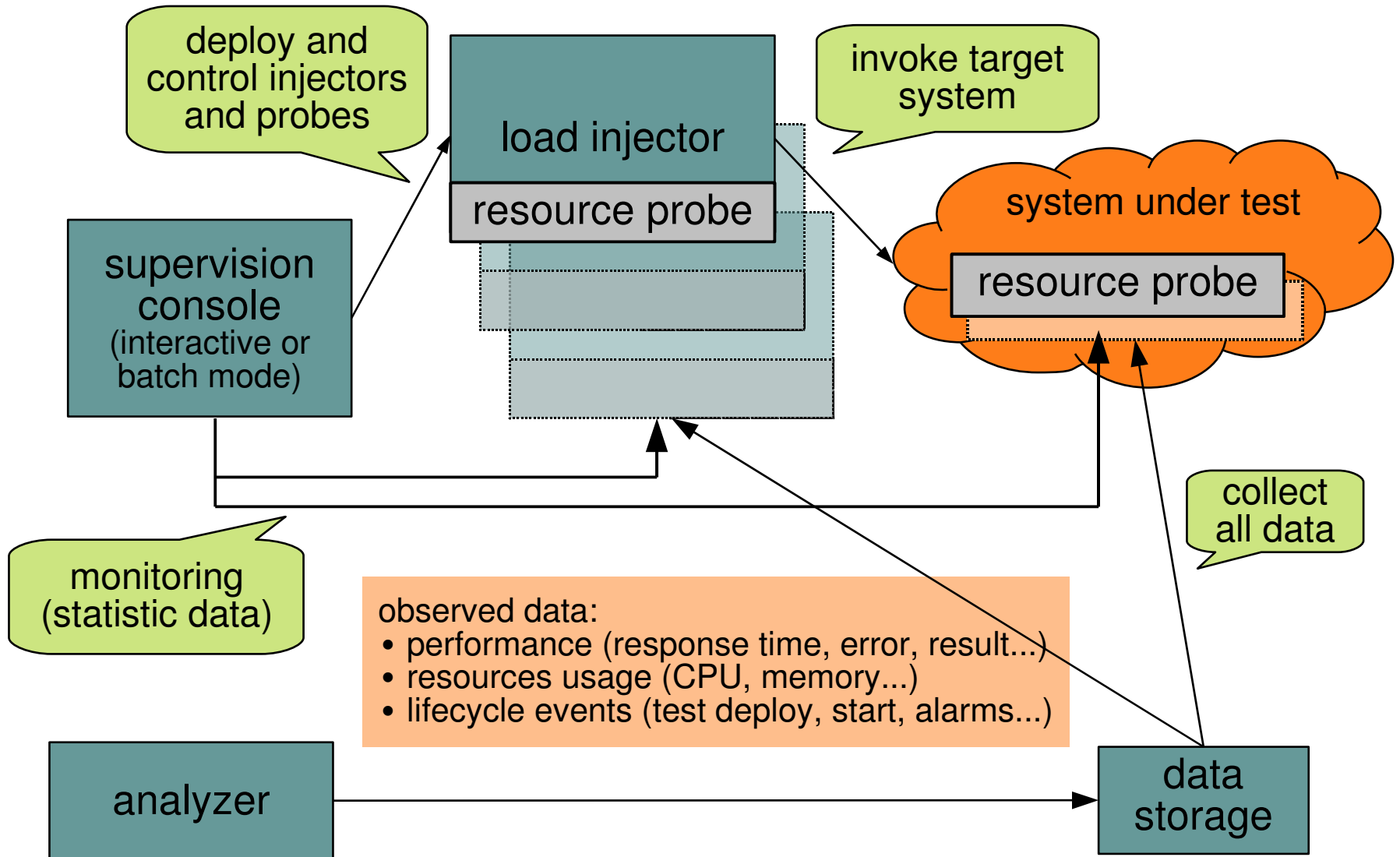
➔ CLIF is dedicated to:

- high-level (distributed) load injection
- performance measurement (response time, error occurrence, etc.)
- resources consumption measurement (probes)

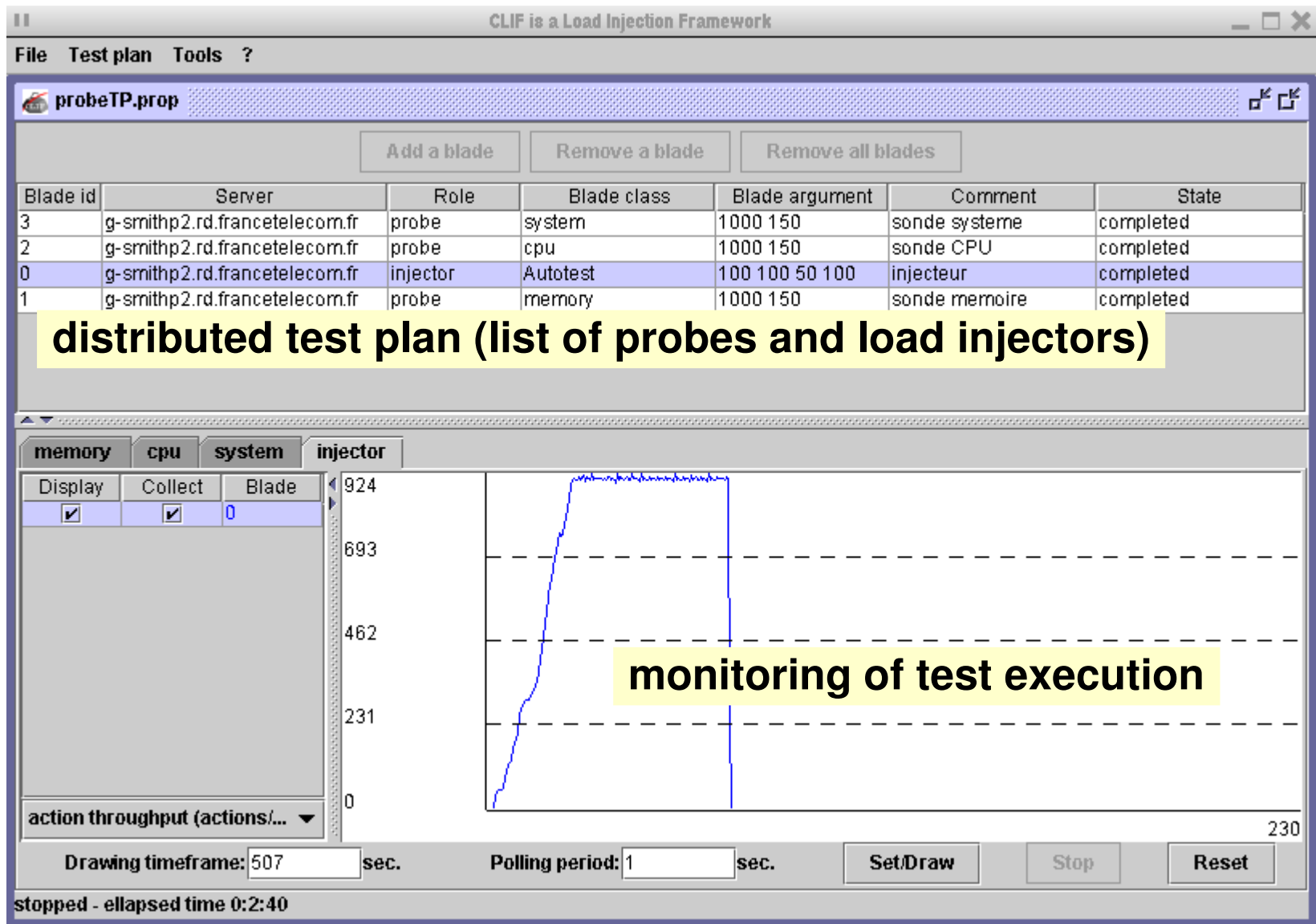
➔ ... open and flexible...

- the framework is independent from:
 - the scenario definition mode,
 - system under test (protocols, observed resources)
 - the final analysis and reporting tasks
- the architecture uses Fractal Component Model
- CLIF is open source software (<http://clif.ow2.org/>)

CLIF distributed infrastructure



CLIF's Java swing-based console



The screenshot displays the CLIF Java Swing-based console interface. The title bar reads "CLIF is a Load Injection Framework". The menu bar includes "File", "Test plan", and "Tools ?". The main window shows a test plan named "probeTP.prop" with three buttons: "Add a blade", "Remove a blade", and "Remove all blades". Below these buttons is a table listing the blades:

Blade id	Server	Role	Blade class	Blade argument	Comment	State
3	g-smithp2.rd.francetelecom.fr	probe	system	1000 150	sonde systeme	completed
2	g-smithp2.rd.francetelecom.fr	probe	cpu	1000 150	sonde CPU	completed
0	g-smithp2.rd.francetelecom.fr	injector	Autotest	100 100 50 100	injecteur	completed
1	g-smithp2.rd.francetelecom.fr	probe	memory	1000 150	sonde memoire	completed

A yellow highlight is placed over the table with the text "distributed test plan (list of probes and load injectors)".

Below the table, there are tabs for "memory", "cpu", "system", and "injector". The "injector" tab is selected. On the left, there is a table with columns "Display", "Collect", and "Blade":

Display	Collect	Blade
<input checked="" type="checkbox"/>	<input checked="" type="checkbox"/>	0

Below this table is a dropdown menu labeled "action throughput (actions/...)". To the right of the table is a line graph showing the monitoring of test execution. The y-axis ranges from 0 to 924. The graph shows a blue line that rises from 0 to approximately 924, stays flat for a short period, and then drops sharply to 0. A yellow highlight is placed over the graph with the text "monitoring of test execution".

At the bottom of the console, there are input fields for "Drawing timeframe: 507 sec." and "Polling period: 1 sec.", along with buttons for "Set/Draw", "Stop", and "Reset". The status bar at the bottom left shows "stopped - elapsed time 0:2:40".

Defining load test scenarios (workload)

What is a load injector?

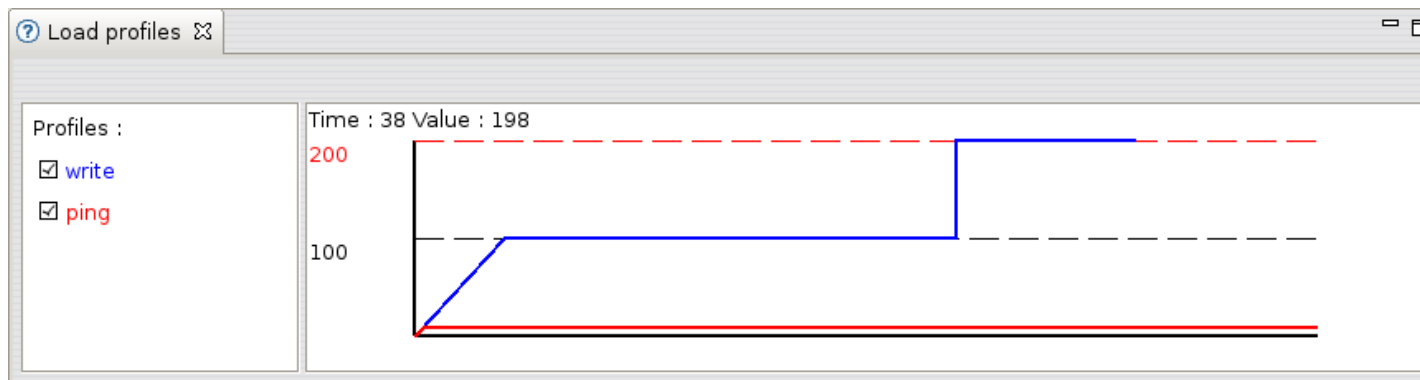
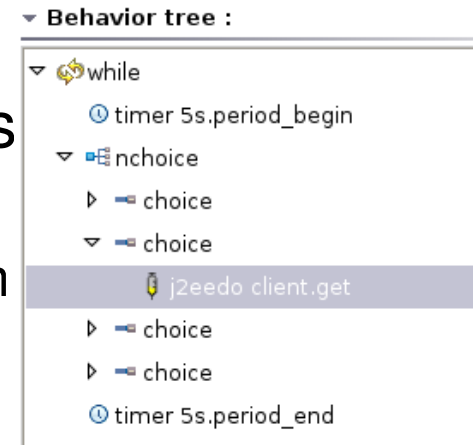
- ➔ **A load injector is a (Fractal) component with an autonomous activity that generates**
 - requests on the System Under Test
 - response time measurement events
- ➔ **Fractal and Java developers may define arbitrary load injectors**
 - from scratch (Fractal component)
 - inheriting from the MTScenario helper (Java class)
- ➔ **ISAC provides an advanced framework for defining load test scenarios**

ISAC is a Scenario Architecture for CLIF

ISAC scenarios

➔ ISAC provides a generic, formal and user-friendly way to define a load test scenario:

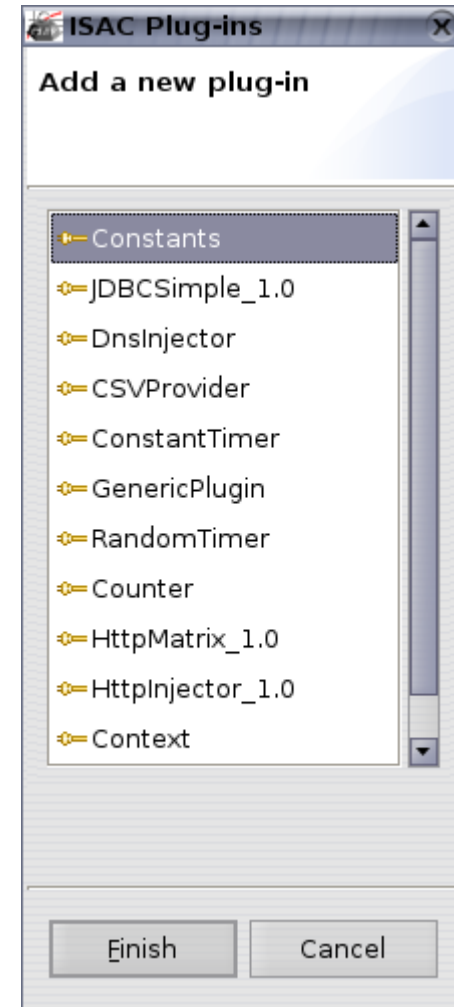
- definition of behaviors (virtual users)
 - sequences of requests and think times
 - with control constructs
 - if, while, probabilistic branching, preemption
- one load profile per behavior
 - number of active instances = $f(\text{time})$



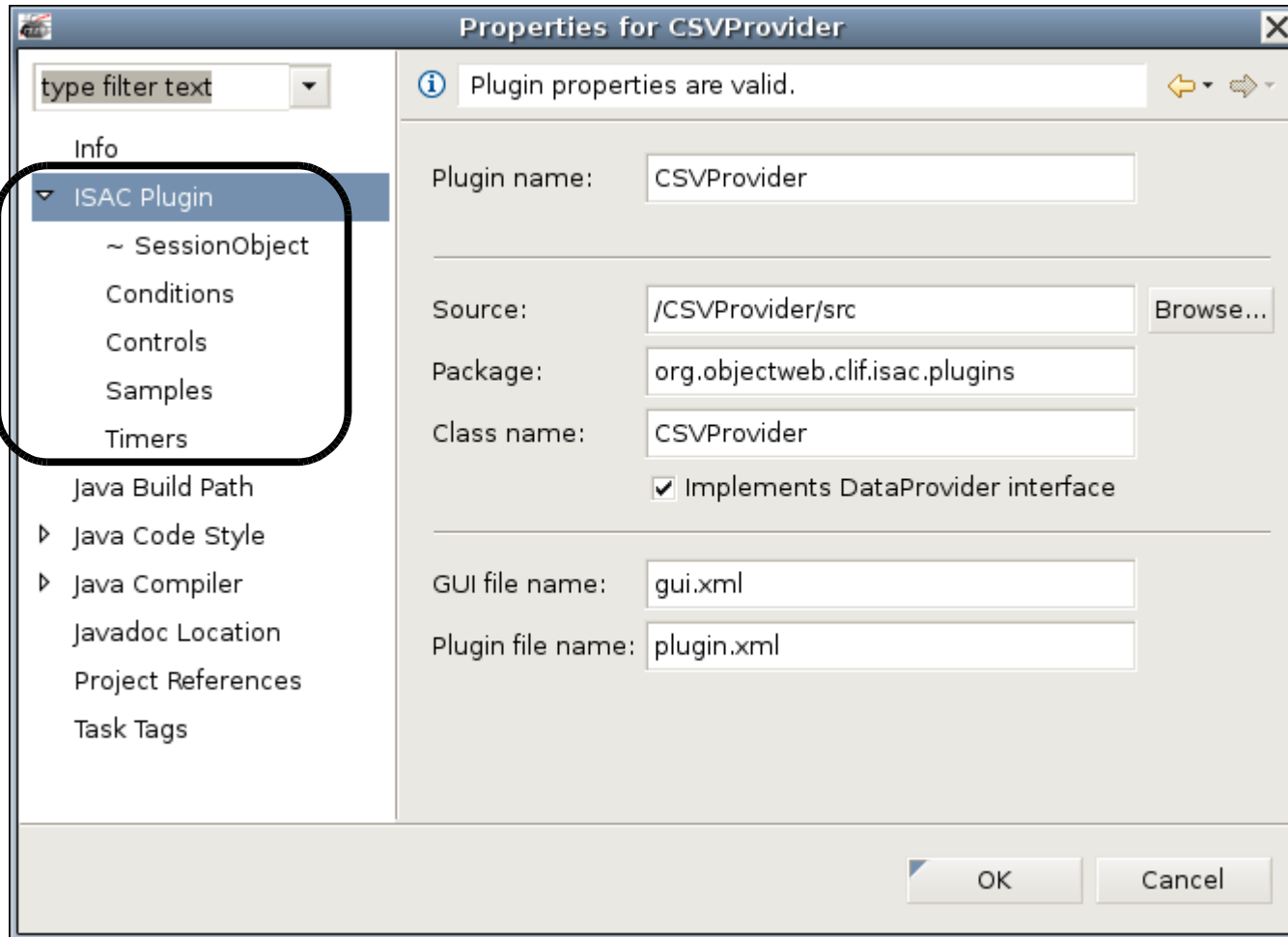
ISAC plug-ins

➔ ISAC is generic and extensible through plug-ins providing:

- invocation protocols
DNS, UDP, TCP, LDAP, HTTP, JDBC...
- delays think times
fix, random, arbitrarily computed delays
- conditions
for if/while/preemption constructs
- test data set provisioning
data picked from files, computed, randomly generated, extracted from replies...



ISAC plug-in creation wizard for Eclipse

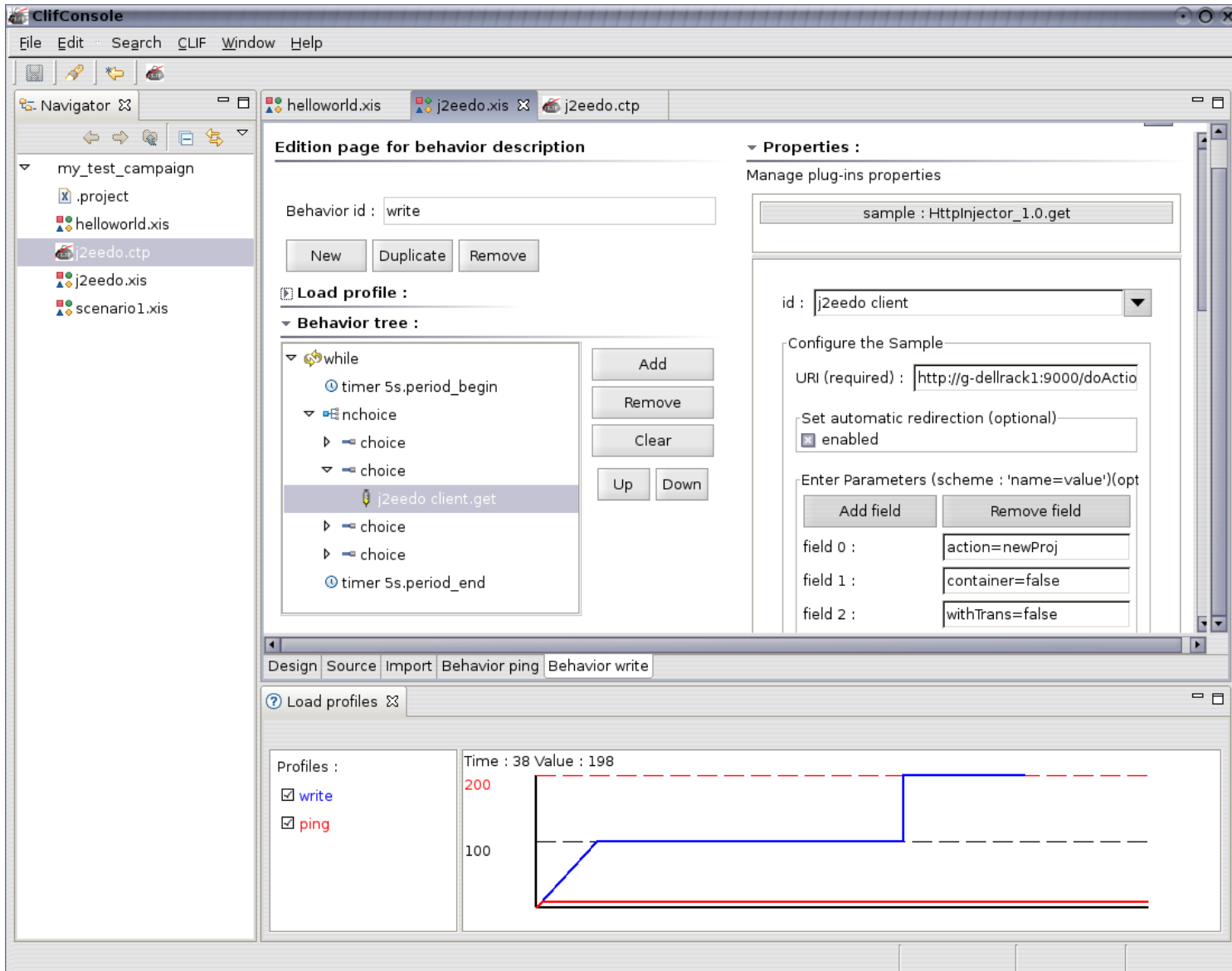


ISAC execution engine

➔ The ISAC execution engine provides advanced features:

- dynamically resizable pool of threads
 - possibly millions of virtual users per load injector
 - up to one million req/s throughput per injector
 - > *in other words, with the right tuning, the engine overhead is negligible*
- dynamic load profiles override
 - change the number of active virtual users
 - interactively or programmatically
- alarms when the execution engine is overloaded
 - detection of think time violation, with adjustable threshold

ISAC scenario graphical editor



The screenshot shows the ClifConsole graphical editor interface. The main window is titled "Edition page for behavior description". On the left, a "Navigator" pane shows a tree structure under "my_test_campaign" with items: ".project", "helloworld.xis", "j2eedo.ctp" (selected), "j2eedo.xis", and "scenario1.xis". The main area is divided into several sections:

- Behavior id:** A text field containing "write". Below it are "New", "Duplicate", and "Remove" buttons.
- Load profile:** A section with a "Load profile" icon and a "Behavior tree" section.
- Behavior tree:** A tree view showing a "while" loop containing:
 - timer 5s.period_begin
 - nchoice
 - choice
 - choice (highlighted)
 - j2eedo client.get (highlighted)
 - choice
 - choice
 - timer 5s.period_endButtons "Add", "Remove", "Clear", "Up", and "Down" are to the right.
- Properties:** A section titled "Manage plug-ins properties" showing a "sample : HttpInjector_1.0.get" field. Below it is a dropdown menu for "id" set to "j2eedo client".
- Configure the Sample:** A section with a "URI (required)" field containing "http://g-dellrack1:9000/doActio". Below it is a "Set automatic redirection (optional)" section with an "enabled" checkbox.
- Enter Parameters (scheme : 'name=value')(opt):** A section with "Add field" and "Remove field" buttons. It lists three fields:
 - field 0 : action=newProj
 - field 1 : container=false
 - field 2 : withTrans=false

At the bottom, there is a "Load profiles" section with a "Profiles" list containing "write" and "ping" (both checked). To the right is a graph showing "Time : 38 Value : 198". The graph has a blue line that starts at (0,0), rises to a value of 100, stays constant until time 38, then jumps to 198. A red line is constant at 0. Dashed horizontal lines are at 100 and 200.

Running load tests with CLIF

Overview of a CLIF test run process

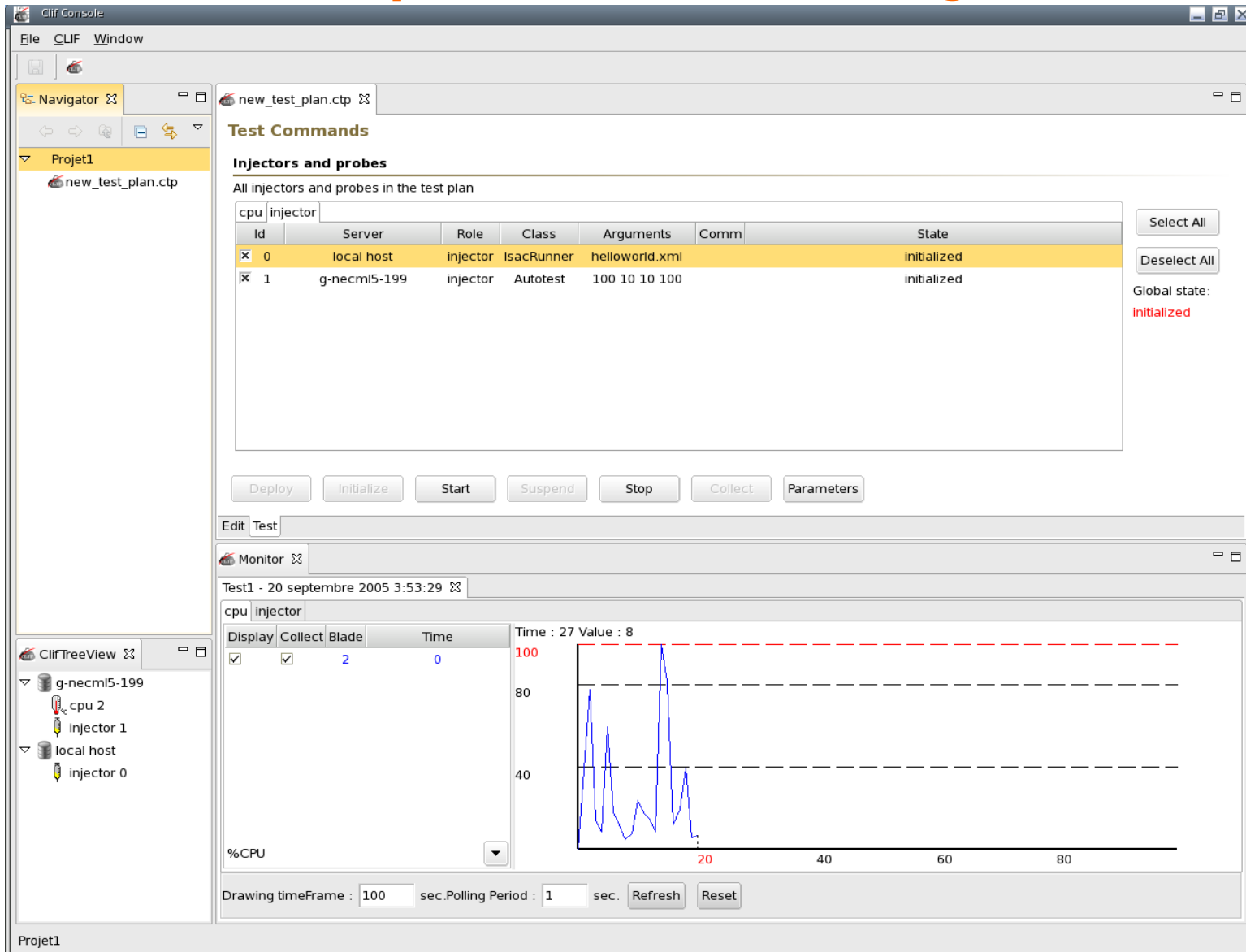
- 1) start the CLIF registry
for distribution support (CLIF servers registration)
- 2) start the CLIF servers
where you plan to deploy probes and load injectors
- 3) deploy your test plan
deployment of load injectors and probes
- 4) run the test once or several times
 - *initialize, start, monitor execution*
 - *possibly abort, suspend/resume the execution*
 - *collect data (measures, alarms, lifecycle events)*
- 5) analyze results

Console for Eclipse: test plan edition

The screenshot displays the ClifConsole application window. The title bar reads "ClifConsole". The menu bar includes "File", "Edit", "Search", "CLIF", "Window", and "Help". The interface is divided into several panes:

- Navigator:** Shows a tree view of a test campaign named "my_test_campaign" containing files ".project", "j2eedo.ctp", and "j2eedo.xis".
- ClifTreeView:** Shows a tree view of the test plan structure, including "local host" (with sub-items "jvm 3" and "injector 0") and "g-ong" (with sub-items "jvm 2" and "memory 1").
- Test Plan Editor:** The main workspace for editing the test plan. It features a tabbed interface with "j2eedo.xis" and "j2eedo.ctp" open. The "Injectors and probes" section displays a table of all injectors and probes in the test plan. The table has columns for "Id", "Server", "Role", "Class", "Arguments", and "Comment". One entry is visible: Id: 0, Server: local host, Role: injector, Class: IsacRunner, Arguments: j2eedo.xis. To the right of the table are buttons for "Add", "Remove", and "Remove All". Below the table is the "Properties" section, which allows managing injector and probe properties. It includes fields for "Id*", "Server*" (with a dropdown and "Refresh" button), "Role*" (with a dropdown), "Class*", "Arguments", and "Comment". At the bottom of the editor are "Edit" and "Test" buttons.

Console for Eclipse: test monitoring



The screenshot displays the Clif Console interface within Eclipse. The main window is titled 'Clif Console' and contains several panes:

- Navigator:** Shows a project named 'Projet1' containing a file 'new_test_plan.ctp'.
- Test Commands:** Displays 'Test Commands' for 'new_test_plan.ctp'. Under 'Injectors and probes', it lists all injectors and probes in the test plan. A table shows the following data:

Id	Server	Role	Class	Arguments	Comm	State
0	local host	injector	IsacRunner	helloworld.xml		initialized
1	g-necmi5-199	injector	Autotest	100 10 10 100		initialized

Buttons for 'Select All', 'Deselect All', and 'Global state: initialized' are present.
- Control Buttons:** A row of buttons includes 'Deploy', 'Initialize', 'Start', 'Suspend', 'Stop', 'Collect', and 'Parameters'.
- Monitor:** Shows a graph for 'Test1 - 20 septembre 2005 3:53:29'. The graph plots '%CPU' usage over time. The y-axis ranges from 0 to 100, and the x-axis ranges from 0 to 80. A blue line shows CPU usage fluctuating between approximately 20% and 80%. A red dashed line is at 100%. Below the graph, controls for 'Drawing timeFrame : 100 sec. Polling Period : 1 sec.' and 'Refresh'/'Reset' buttons are visible.
- ClifTreeView:** A tree view on the left shows the hierarchy: 'g-necmi5-199' (containing 'cpu 2', 'injector 1') and 'local host' (containing 'injector 0').

Command-line user interface

➔ A set of Ant-based commands provide full control on tests:

- **config** – Basic settings of CLIF execution environment (CLIF server, Swing GUI)
- **registry** – Runs a CLIF registry.
- **server** – Runs a CLIF server.
- **deploy** – Deploys a test plan
- **init, start, suspend, resume, stop** – Initializes, starts, suspends, resumes or stops a test.
- **join** – Waits until the test execution is terminated
- **collect** – Collects data produced by the test's injectors and probes
- **run** – Short-cut for “init, start, join, collect”
- **launch** – Short-cut for “deploy, run”
- **params** – Lists all parameters that can be changed during execution for a given probe or injector
- **change** – Changes a parameter value for a given probe or injector

CLIF probes

➔ Probes are available for Linux, MacOSX and Windows

- based on the LeWYS project
 - cpu, memory, disk, network
 - simple and low footprint design
 - native code for Windows and MacOSX
 - pure Java for Linux
- jvm

➔ Simple framework to define any kind of probe

- e.g. sample JMX-based probe for JVM
- SNMP probes (not published yet)

Test analysis and reporting

Test results

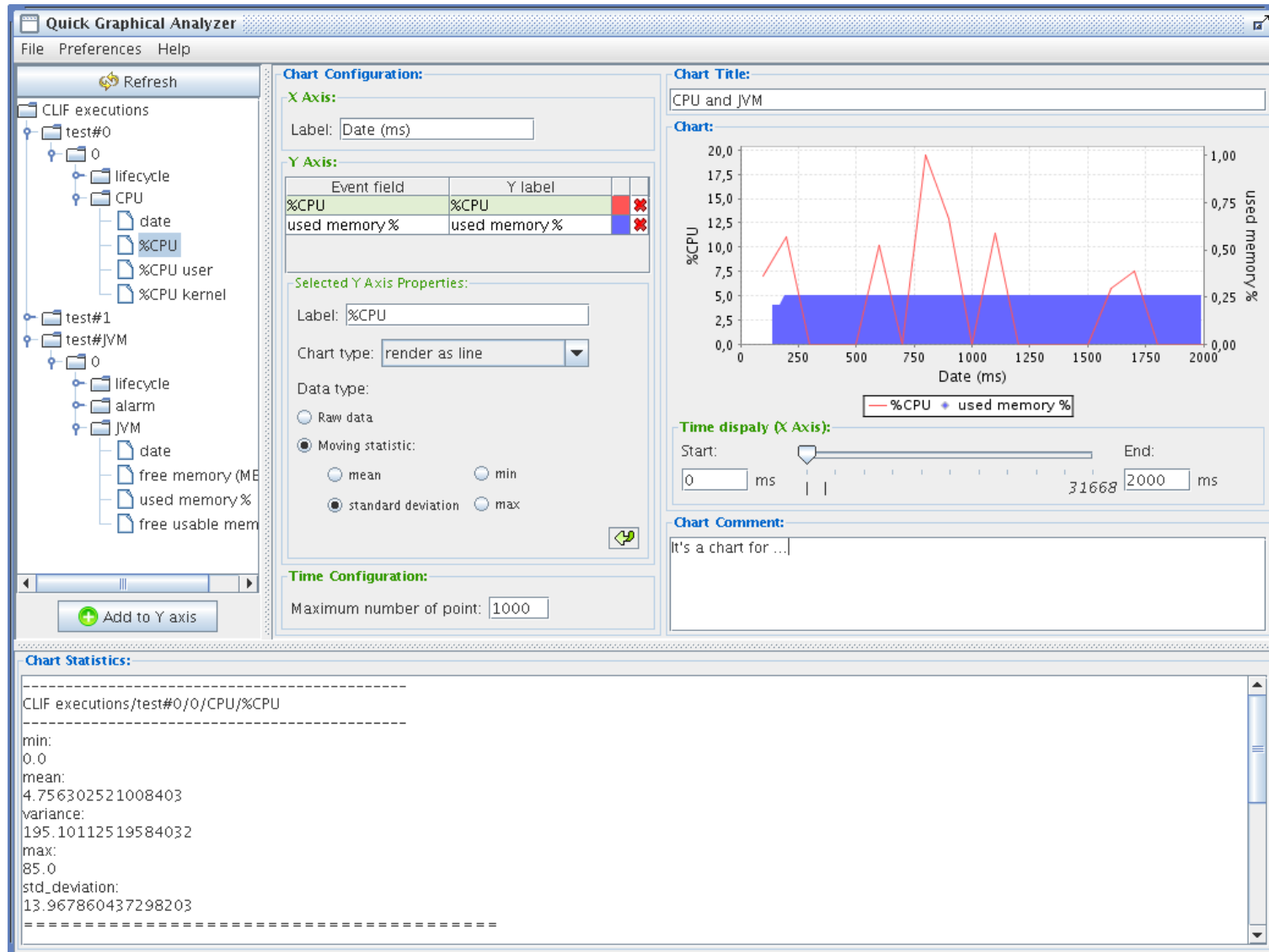
➔ **At runtime: monitoring information**

- min/max/average response time
- requests and errors throughput, error rate
- specific measurements from probes

➔ ***Post mortem analysis***

- measures are available as CSV-formatted files
- JavaSwing-based “Quick graphical analyzer”
 - first version available (simple graphs and reports)
 - ongoing work on advanced statistical features
- integration with the Eclipse BIRT project
(ODA driver for accessing CLIF data)

“Quick graphical analyzer” (v1)



Conclusion

Current status and work plan for 2008

Current status and work plan

➔ Lutèce d'Or 2007 open source award!

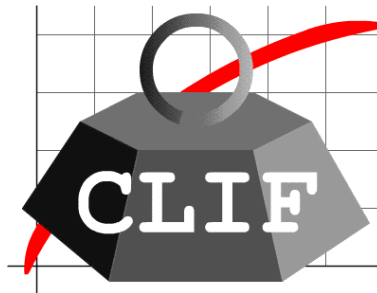


➔ The CLIF community

- France Télécom is a leading user and contributor (more than 30 R&D projects have been/are using CLIF)
- involvement of INRIA and Bull
- growing number of other companies or individuals
- good feedback/testimonials *wrt* other open source load testing platforms

➔ Development plans for year 2008

- full reorganization of the code base into sub-projects
- extension of the JavaSwing-based analysis and reporting tool
- integration with Salomé Test Management Framework



clif.ow2.org

For more information
Please contact
clif@ow2.org