Self-managed Component-based Software Architecture for Business Process Management

Bassem Debbabi[1] ,Thomas Calmant[1], Olivier Gattaz[1], Sandra Massonnat[2], Patrick Emin[2]

1 - IsandlaTech, 3 chemin du vieux chêne,38240 Meylan, France

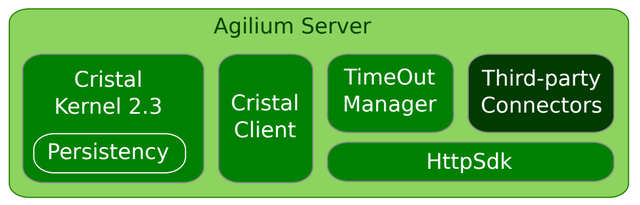
2 - M1i, 15 route de Nanfray, 74960 Cran Gevrier, France

CONTEXT

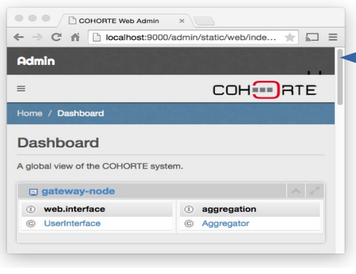
- Business Process Management (BPM) is a system that manages the entire chains of events, activities and decisions that ultimately produce added value for an organization [1].
- Agilium is a BPM tool that is currently developed and sold to industrial and commercial customers. It is used in different domains like retail, logistics, production management and/or order processing.
- Agilium is based on CRISTAL technology [2] outcome from the European Center for Nuclear Research (CERN) in Geneva.

PROBLEMS

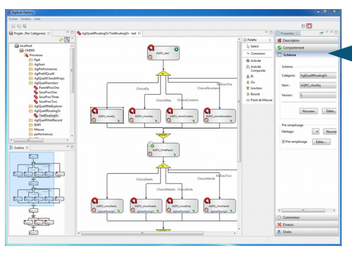
- The Agilium server is a classic Java application that runs on one JVM.
- known runtime issues:
  - **Libraries hell**: The dependencies and classpaths should be handled rigorously.
  - **Third-party code safety**: when third-party code crashes, all the server goes down. We should identify the source of the problem while trying, as quickly as possible, to relaunch the server.



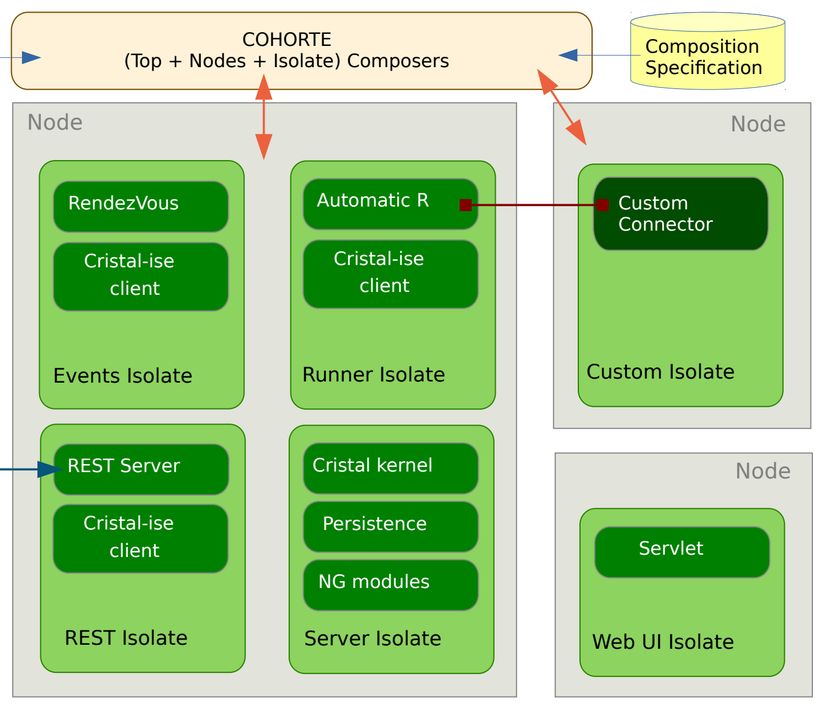
APPROACH



Cohorte Monitoring



Agilium Factory



- Using the developed Cohorte framework* for self- managing and distributing the different components of the application
- The new architecture (Agilium NG) is composed of:
  - initial five isolates: Server, Runner, Events, REST and Web UI.
  - At runtime, the Service-Oriented Components of the application are automatically dispatched on the different isolates by the multi-level composer according to the constraints defined in the composition specification.
  - If a component goes down, it is progressively isolated from the rest of components

* http://cohorte.github.io

RESULTS

- **Global crash prevention**: the failure of a third-party component deployed in a custom isolate has no impact on the availability of the Agilium application.
- **Load adaptation**: A new instance of an automatic runner component can be deployed in a new remote isolate to accept dynamically an increase of the load.
- **Easy enhancement and deployment**: the distribution of components in many isolates is an effective way to brake the limits of the management of the libraries dependencies (cf. The java class-loading hell).

WHAT IS COHORTE ?

Cohorte Framework is based on OSGi standard and provides a novel architecture for self-managing distributed components over several containers (isolates).

It is based on three main concepts:

(1) the instantiation and composition of application's components is handled automatically by a multi-level composers;

(2) the handling of heterogeneous service- based component models including Apache Felix iPOJO [3] for Java and IsandlaTech iPOPO [4] for Python; and

(3) a connectivity layer that abstracts the remote services between components.

REFERENCES

[1] DUMAS, Marlon, LA ROSA, Marcello, MENDLING, Jan, et al. Fundamentals of business process management. Berlin : Springer, 2013.

[2] J. Shamdasani, et al. CRISTAL-iSE - Provenance Applied in Industry. ICEIS (3)) 2014: 453-458

[3] Escoffier, C., et al. iPOJO: An extensible service-oriented component framework. SCC 2007 IEEE.

[4] Thomas C., et al. A dynamic and service-oriented component model for python long-lived applications. CBSE'12

M1i
ZA du Levray
15 route de Nanfray
74960 CRAN GEVRIER
Tel : +33 (0) 450 234 594