the "SAMPificator"

Y. Granet, T. Fenouillet, C. Surace, L. Paioro
and the FASE consortium
Laboratoire d'Astrophysique de Marseille

as part of

OPTICON FASE developments

contact : yohann.granet@oamp.fr
Laboratoire d'Astrophysique de Marseille, OAMP, Université de Provence, CNRS, 38 rue Frédéric Joliot-Curie, F-13388 Marseille Cedex 13, France

FASE
Future astronomical Software Environments

- International project (ESO, ESA, INAF, Lund Obs, IA Göttingen, RAL, LAM, NOAO, NRAO, NVO)
- Kick off in 2002 under FP6-OPTICON program
- Financed in the FP7-OPTICON network
- Main goals:
  - Increase software sharing, re-use, and transparency.
  - Allow the use of legacy applications with new systems.
  - Ease the integration of new application within new systems
  - Facilitate the interoperability of applications
    - preferred communication protocol : SAMP

https://www.eso.org/wiki/bin/view/Opticon/WebHome

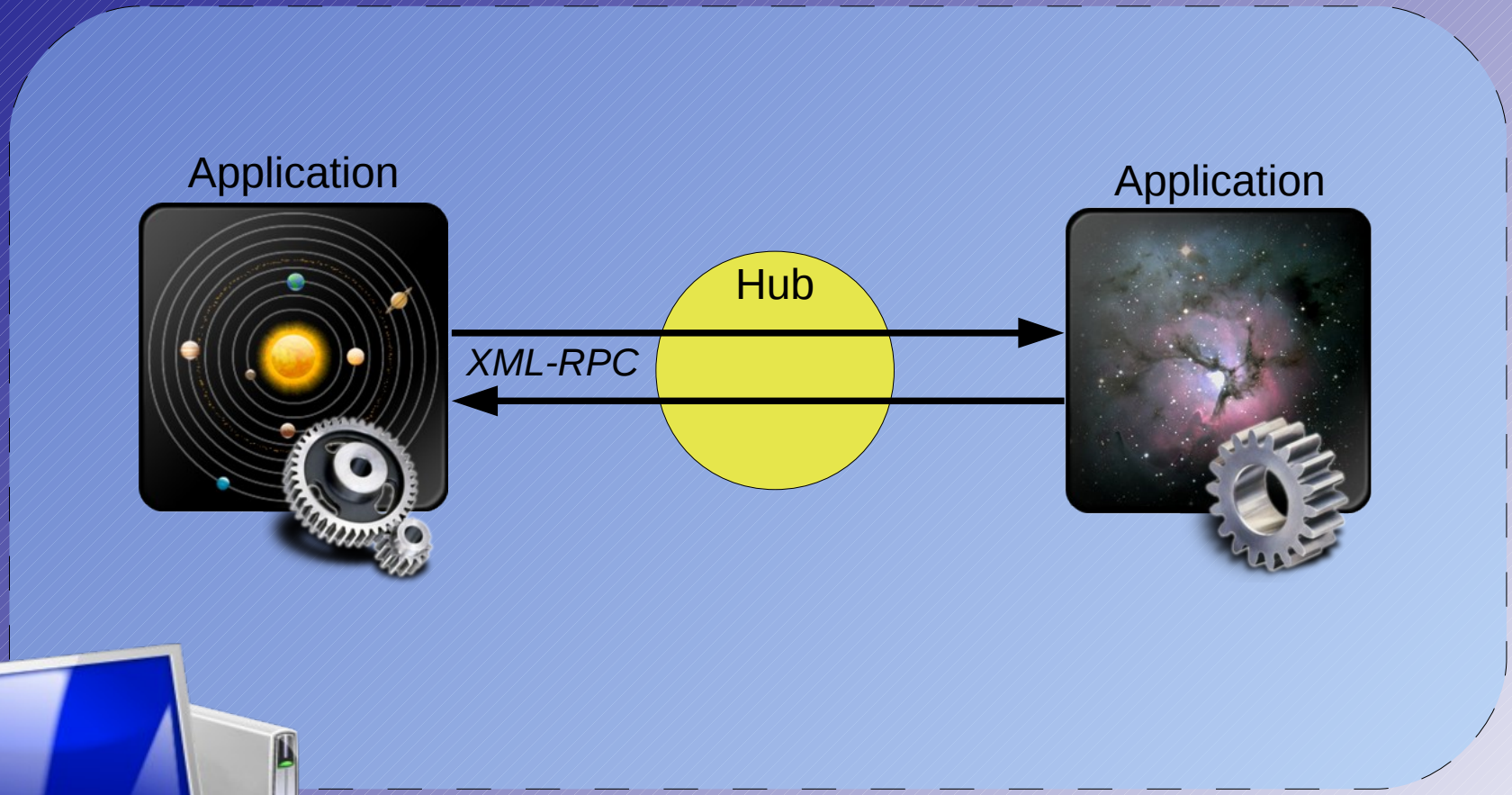
SAMP

Simple Application Messaging protocol

- The direct descendent of PLASTIC
- A communication protocol
- A communication framework:
  - JSAMP (Java, M. Taylor)
  - SAMPy (Python, L. Paioro)
  - Perl SAMP (Perl, A. Allan)

http://www.ivoa.net/cgi-bin/twiki/bin/view/IVOA/SampInfo

SAMP - Standard Usage



SAMP - Issues

- Integration of a SAMP client layer in an application:
  - Dependency on a particular framework...

  - Modification to legacy applications...

(SAMP frameworks: Java, Python and Perl ...

What about Fortran ??...)

- FASE specific issue:
  - Doen'st allow to launch applications on demand...

the Operations Server, alias the "SAMPificator"

In collaboration with INAF – IASF Milano

Operations Server Concept

SAMP world

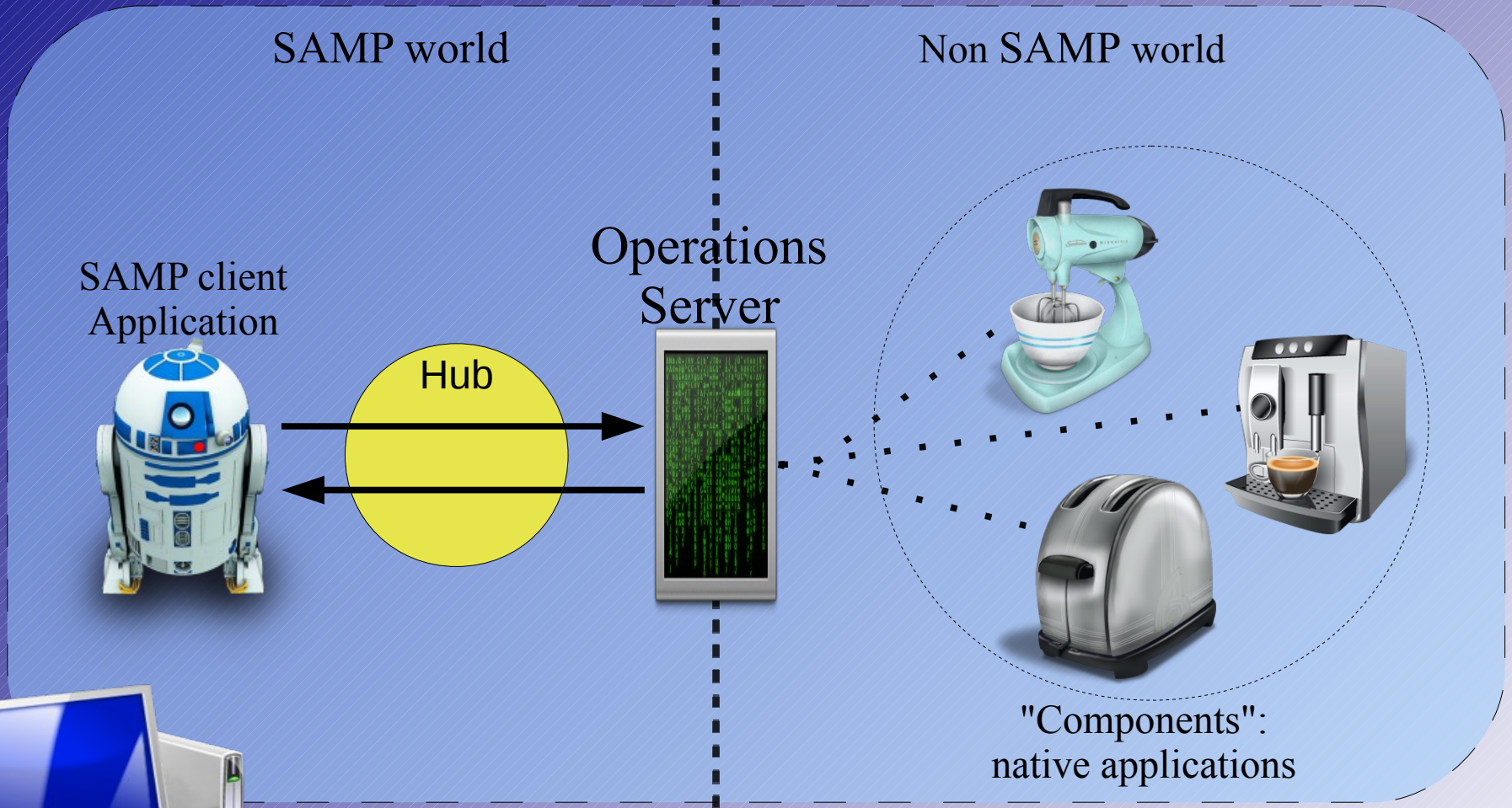
Non SAMP world

SAMP client Application

Hub

"Components": native applications

Operations Server Concept



Operations Server Principle

- The Operations Server drives components thanks to 2 kinds of files:
  - "definition" files describing the components and its input(s)/output(s)
  - "dictionary" files establishing correspondences between components operations and SAMP messages

Operations Server Principle illustration

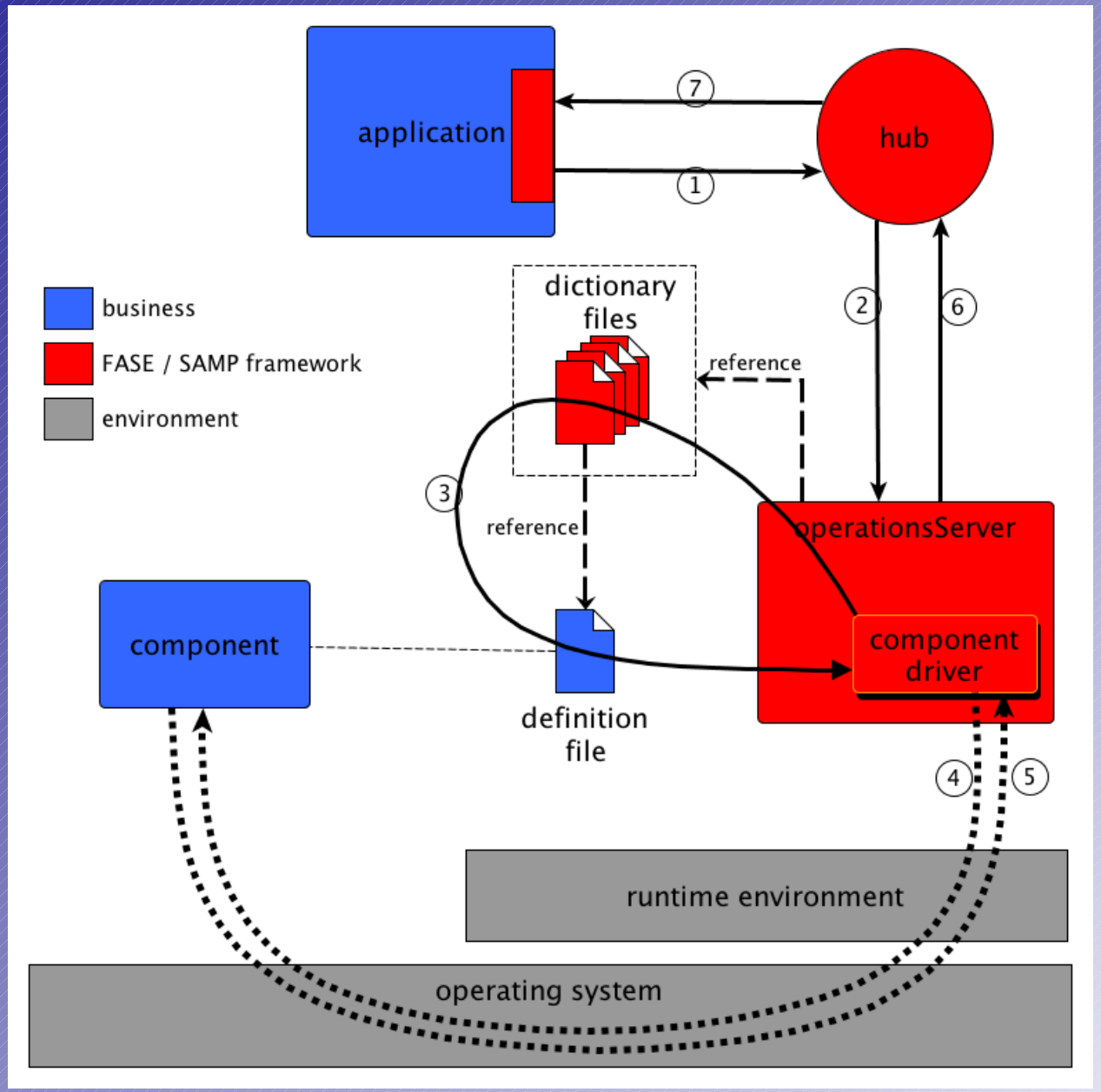
Definition file

Name:
    coffeeMachine
Package:
    org.householdAppliance
Operation:
    Name:
        makeCoffee
    Input:
        groundCoffee
        water
        container
    Output:
        cupOfCoffee

Dictionary file

MessageType:
    coffee.prepare ⇔ makeCoffee
Message content:
    cup ⇔ container
    water ⇔ water
    coffee ⇔ groundCoffee
Response content:
    result ⇔ cupOfCoffee

Operations Server Working



Operations Server Advantages

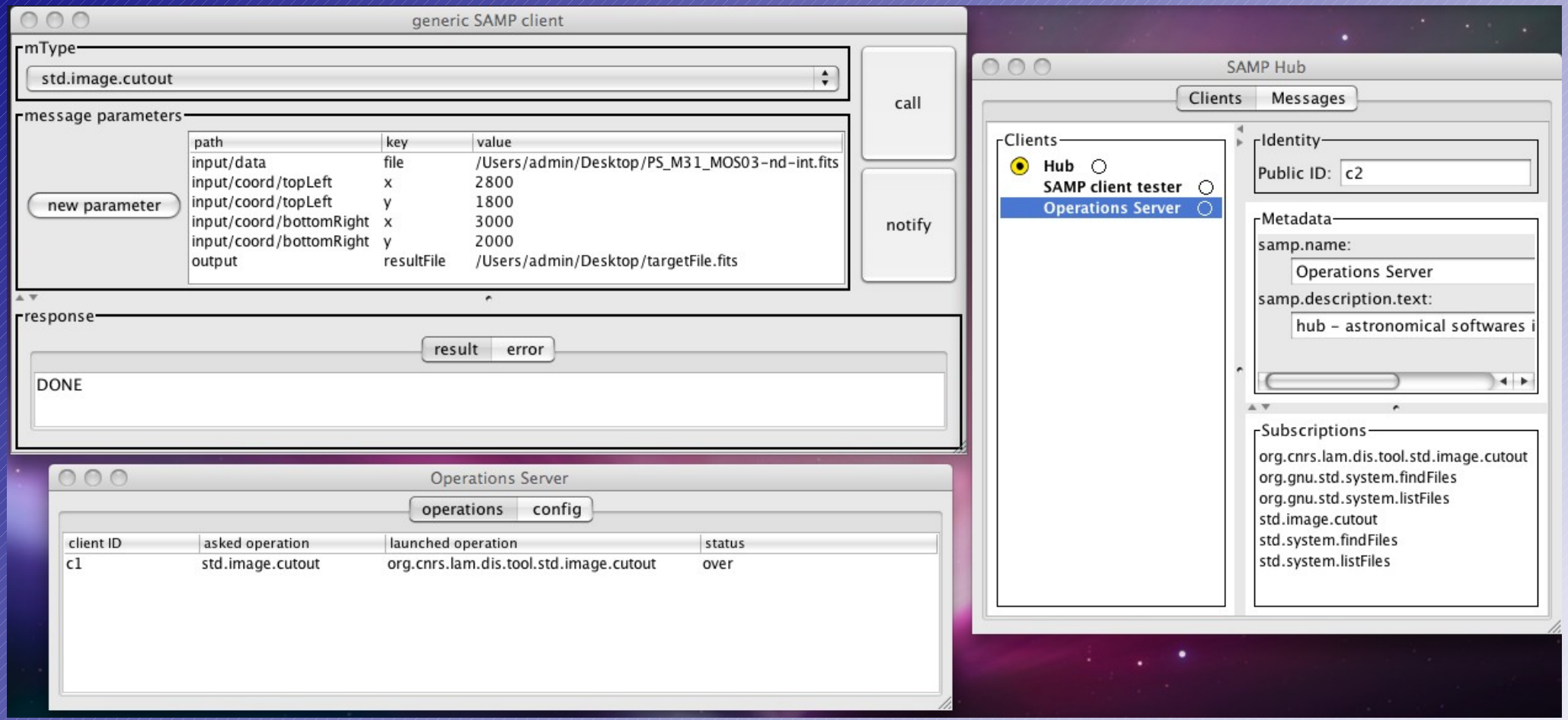
- Almost any native application can be easily plugged to the SAMP world
- No dependency on a particular framework
- "On demand" application launching

Operations Server Limitations

- Special characters in the in/out streams are not supported (SAMP standard)
- Only applications using the standard in/out streams are pluggable (for time being...)
- In case of applications still running on after the end of the operation : special "end of result" token needed to finalize the output.

NB: it is possible to use a specific command to produce this token

Operations Server Demo...



TODO

- Add fucntionnalities:
  - Abort operations
  - Stop or kill tasks
  - Hot-plug new components
  - ...
- Add communication layers:
  - Use its own in/out streams
  - ICE
  - ...
- Enrich the GUI
- Provide a wizard for new components plugging