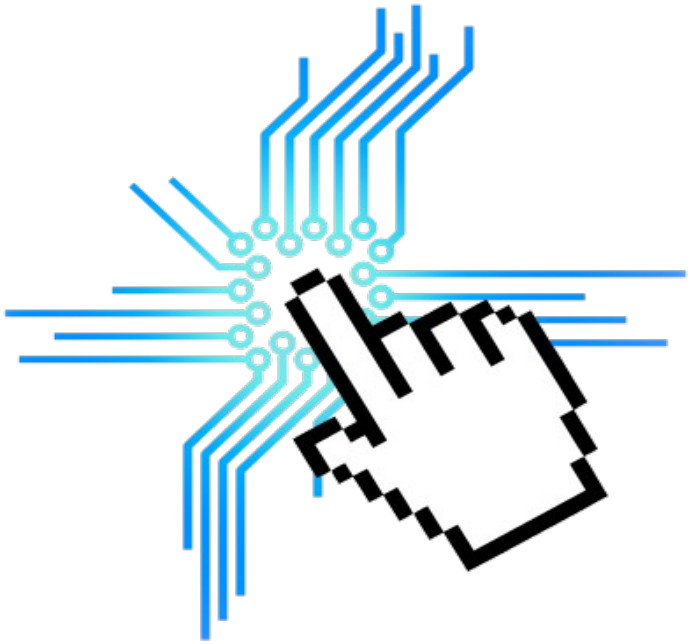
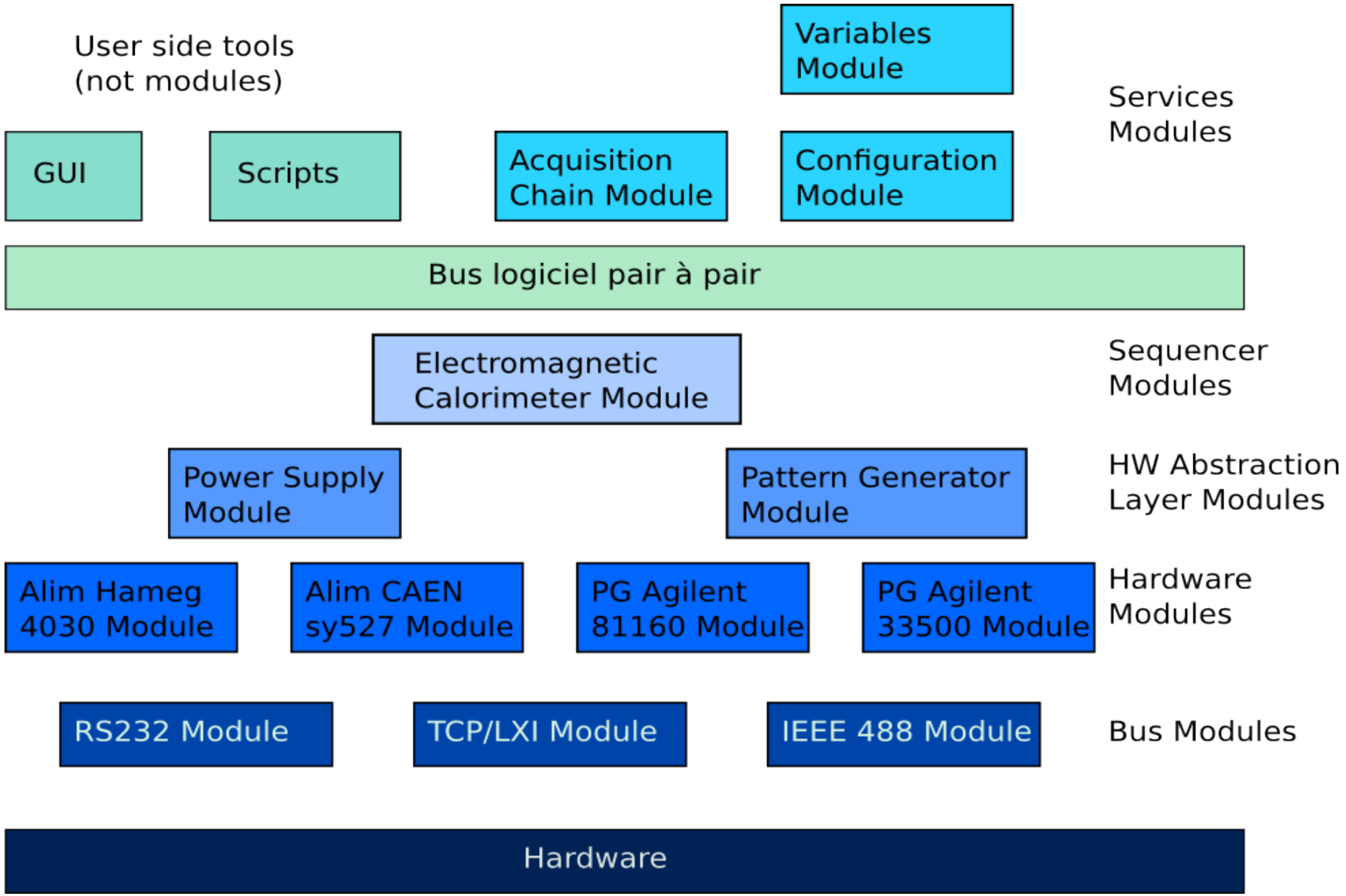


Pyrame, un framework de prototypage rapide pour systèmes online

Frédéric Magniette - LLR

Un framework ultra-modulaire et distribué

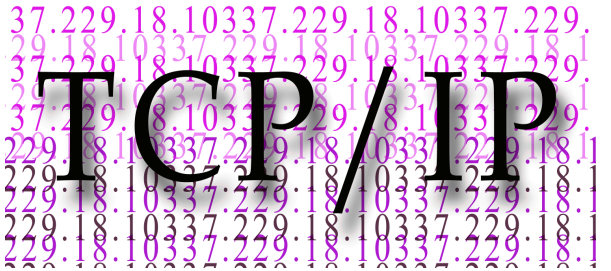


Un protocole de communication

Les commandes sont transmises en XML à travers des sockets TCP.

Exemple de commande :

```
<cmd name="set_rgen_dif">
    <param>0</param>
    <param>120</param>
    <param>50</param>
</cmd>
```



Exemple de réponse :

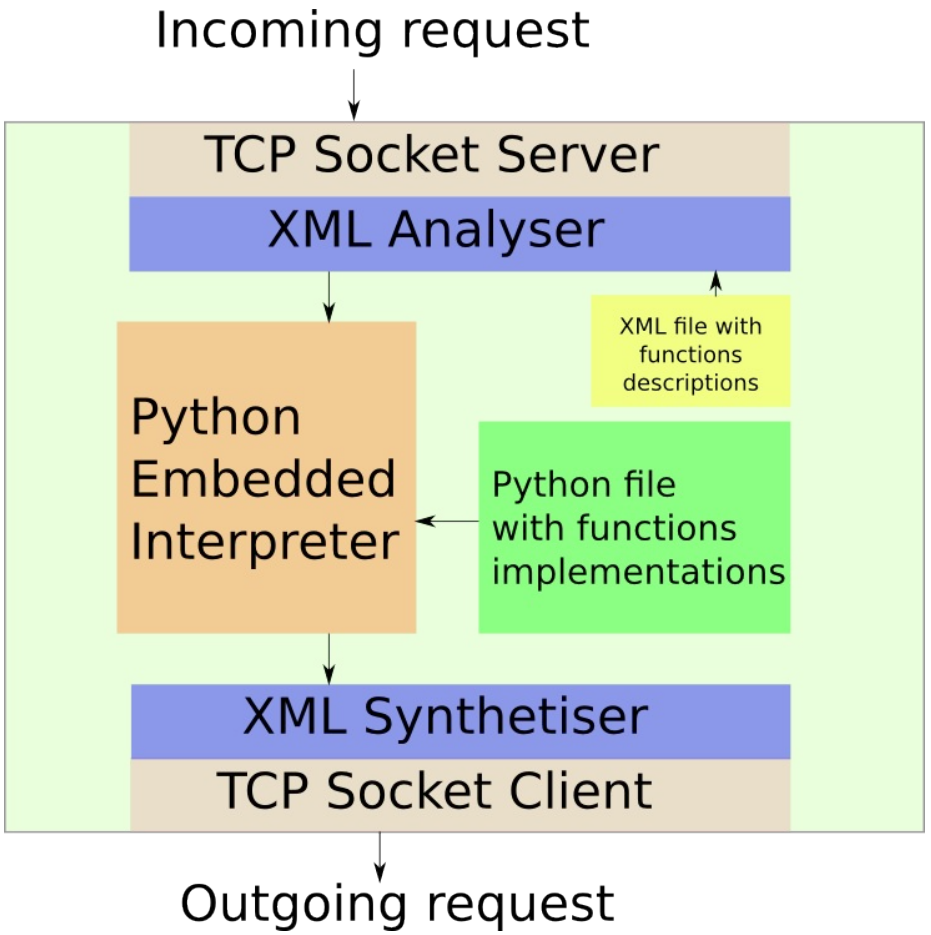
```
<res retcode=1>
    <![CDATA[random generator setted]]>
</res>
```

- retcode contient le code de retour
  - 0 pour l'échec
  - 1 pour le succès
- La chaîne de résultat est protégée par un CDATA

Des modules matériels

- Modules bas-niveau (bus) : RS232, IEEE 488 (GPIB), Ethernet Raw, TCP, UDP, USB
- Modules Alimentation (Agilent, CAEN, Hameg, Keithley…)
- Modules générateurs de pattern (Agilent)
- Modules motion controllers (Newport)
- Modules digital storage oscilloscopes (Lecroy)

Le module de commande : une aide au développement online

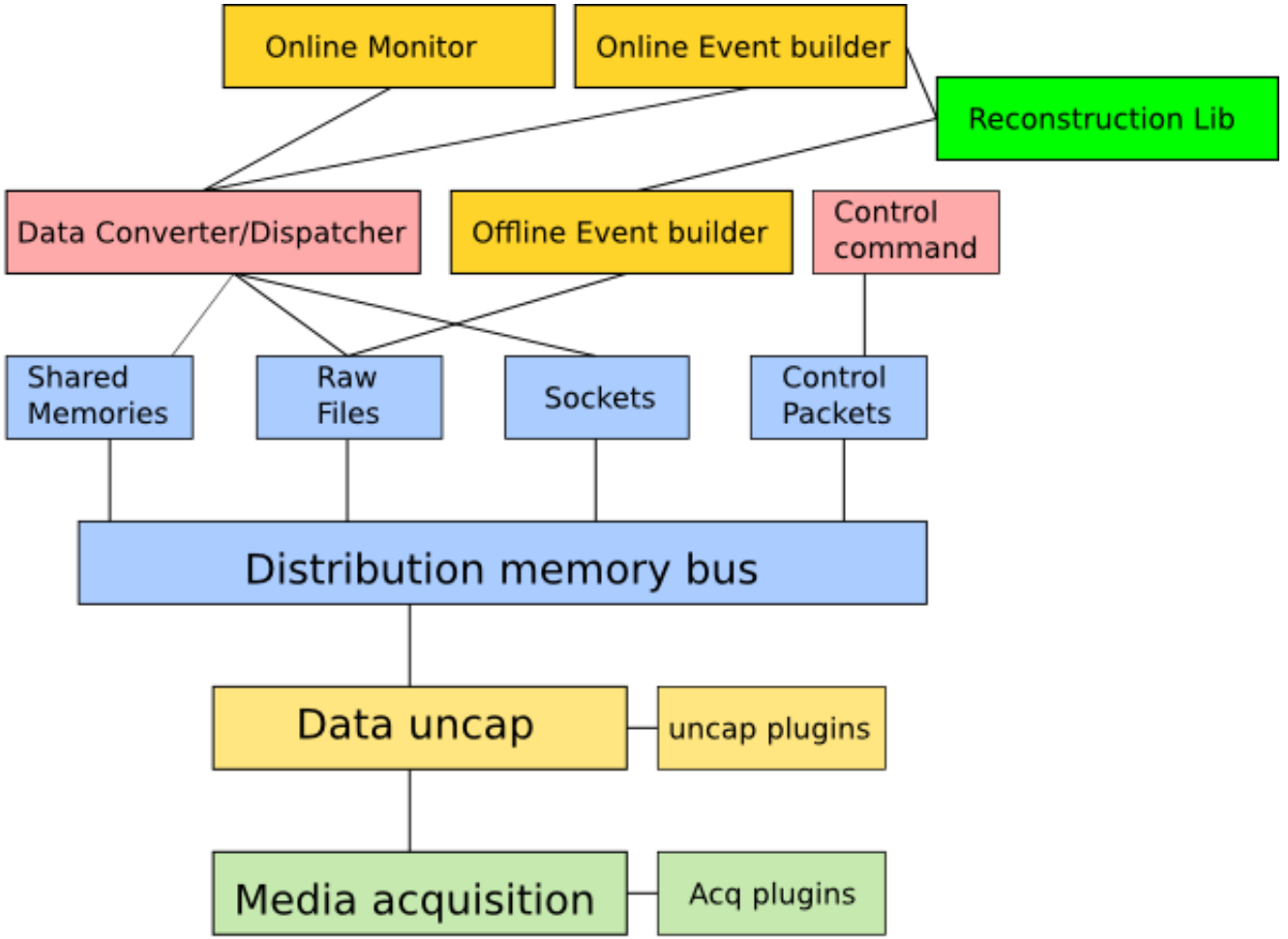


- Abstrait la partie protocolaire (TCP et XML)
- Gère l'aspect distribué du framework
- Embarque une machine Python
- Performance correcte pour un banc-test : 1.6ms/op

Les modules de services

- Module de variables
  - Partage et opérations int et str
  - Collecte de stats
- Module de configuration :
  - Alloue les identifiants aux matériels
  - Sert à repérer les matériels sur le réseau
  - Contient en temps-réel la configuration de tous les composants
  - Exporte la configuration globale en XML

Une chaîne d'acquisition multi-médias et haut débit (1.7Gb/s)

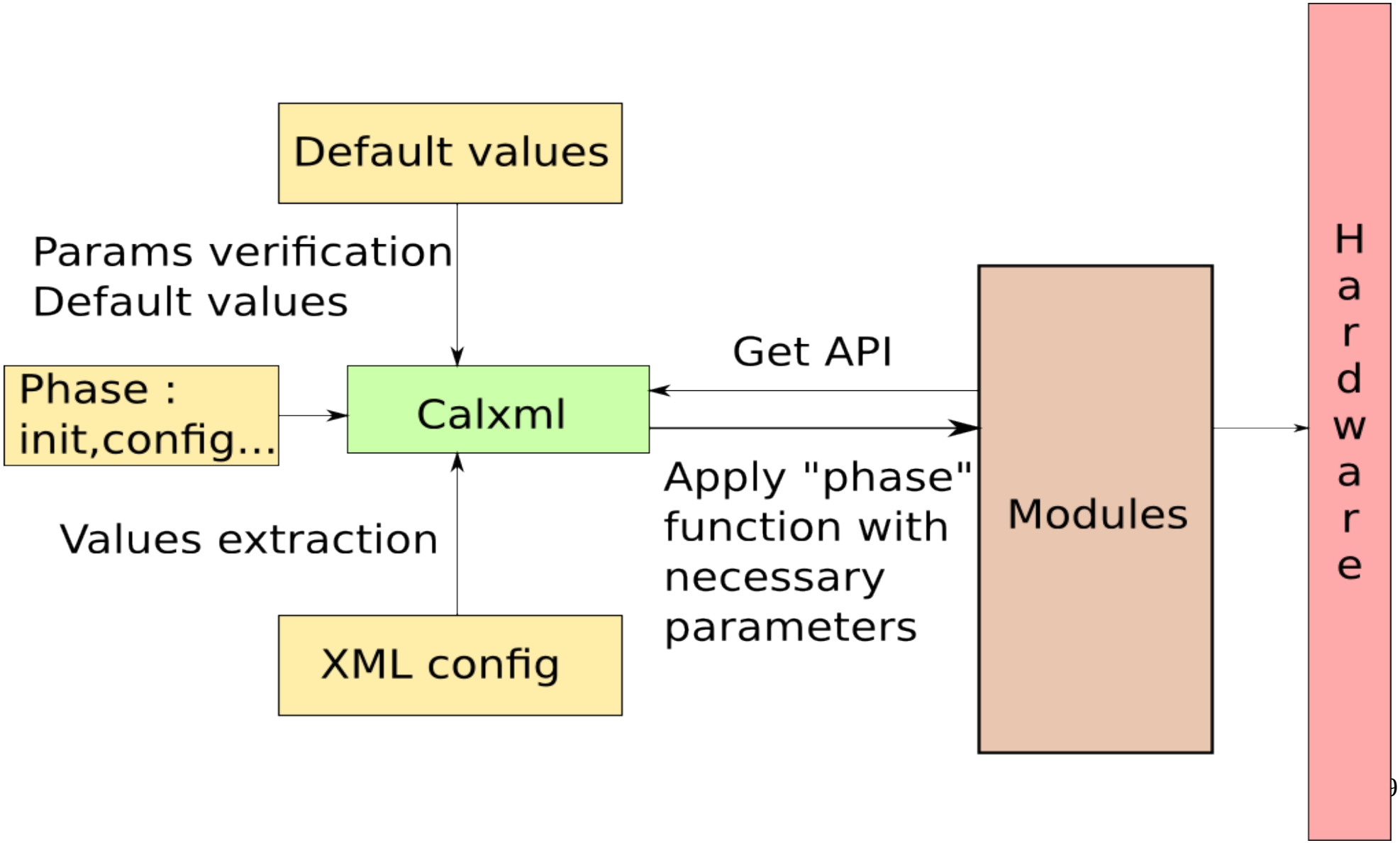


Le format des configurations

- Format lisible basé sur XML
- Grande compacité :
  - Valeurs par défaut
  - Partage des paramètres communs
  - Déclaration implicite

```
<gdcc name="gdcc1_1">
    <param name="gdcc_pc_interface">em2</param>
    <param name="dif_alim_mode">PP</param>
    <param name="dif_nb_skiroc">4</param>
    <dif name="dif1_1_1">
        <param name="dif_gdcc_port">8</param>
    </dif>
    <dif name="dif1_1_2">
        <param name="dif_lda_port">1</param>
    </dif>
</gdcc>
```

Applicateur de configuration



Ouverture vers l'extérieur

- Intégration facilitée par le format ouvert (XML/TCP)
- Bindings C, C++, Python, R, Labview
- Nouveaux bindings très faciles à écrire
- Interfaçage avec les SCADA : Tango (prototype), OPCUA (CTA), Xdaq

Résultats

- Testé sur plusieurs test-beams au DESY et au labo
- Plusieurs manips avec des besoins différents (flexibilité) : CTA, ILD-ECAL, Banc laser...
- Acquisition très stable : centaines d'heures consécutives de prise de données.
- Contrôle-commande très stable : procédures de calibration, à plus de 100000 configurations successives.
- Version 2.0 release Sept 2014

Tests et intégration continue

- Intégration continue via les hooks de mercurial
- Vérification de la compilation
- Verification des tests unitaires
- Parfait pour les modules de service

- Problème de la disponibilité hardware
- Temps moyen de détention d'un appareil : - de 3 jours
- Rôle essentiel des test-beams