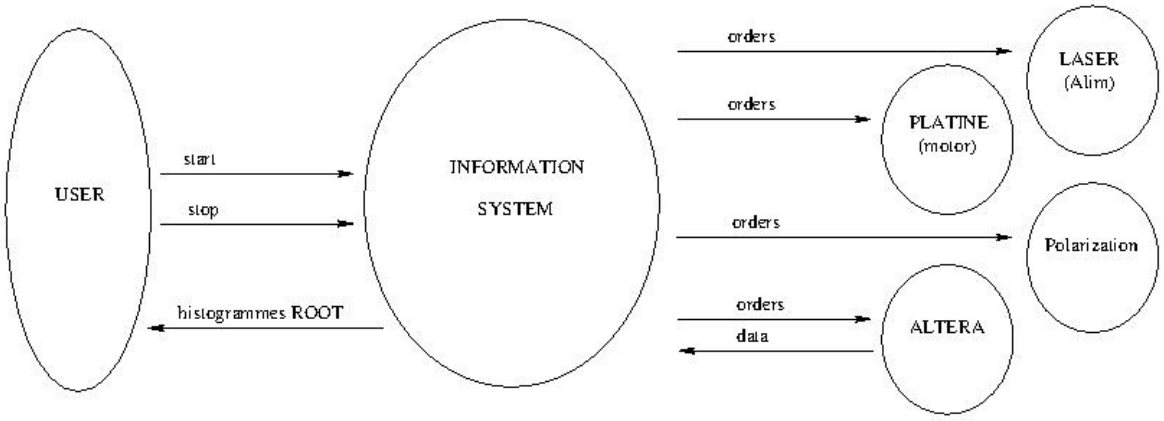
Système d'Information

Contents

1 Introduction

Ce document décrit le système d'information de l'expérience LC. Nous utilisons ici la méthode MERISE pour décrire le système.

2 Modèle conceptuel



2.1 In

Le système d'information réagit aux évènement suivants :

- Ordre 'start' en provenance de l'utilisateur.
- Ordre stop' en provenance de l'utilisateur.
- Données reçues du FPGA ALTERA.

2.2 Out

Ce système communique à l'extérieur les évènements :

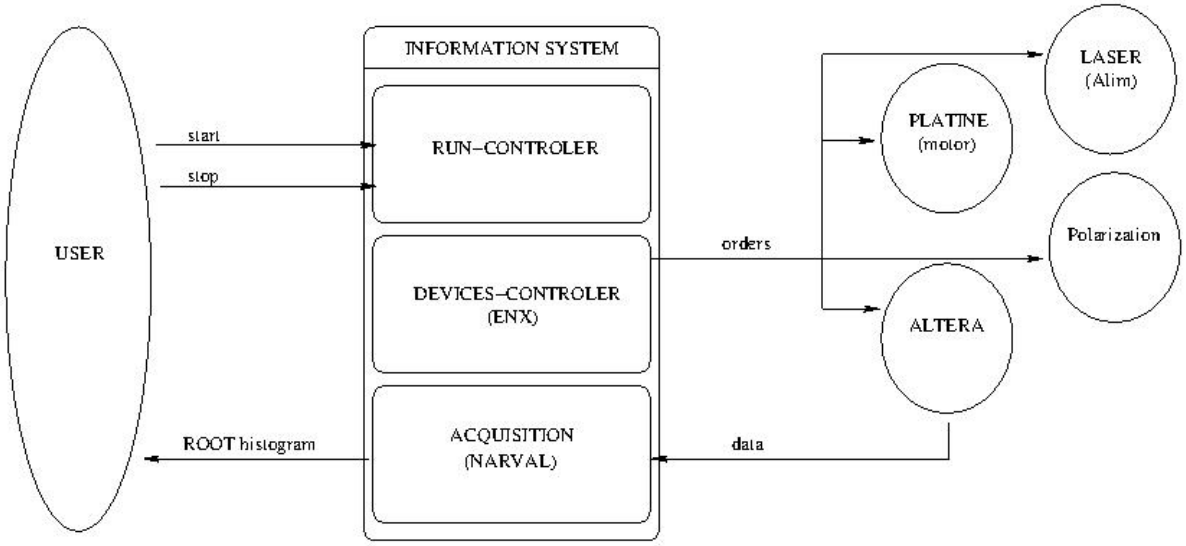
- Ordres à destination de la platine.
- Ordres à destination de l'alimentation responsable de la polarisation.
- Ordres à destination du lazer.
- Ordres à destination de l'ALTERA.

3 Cycle de vie des objets

- **configuration** de l'expérience.

4 Diagramme de flux

Les diagrammes de flux affinent le modèle conceptuel avec un découpage récursif par activités.



4.1 Niveau n+1

Le système d'information divisé en 3 activités :

- RUN-CONTROLER
- DEVICES-CONTROLER
- ACQUISITION

4.2 Événements

Les évènements sont les éléments déclencheurs des différents traitements. Dans la plupart des cas, les évènements mettent en évidence la transition d'un traitement à un autre. Les évènement sont l'abstraction d'un appel à une fonction, à un script, à l'entrée dans le bloc de code d'un switch/case, ou encore à une entrée dans la CRONTAB. Si le traitement est un démon alors l'évènement peut être un signal ou toute autre quelqconque condition.

Le système d'information réagit avec l'environement via les évènements suivants :

- *start* depuis l'UTILISATEUR au traitement controler
- *stop* depuis l'UTILISATEUR au traitement controler
- *orders* depuis le traitement platine à la PLATINE
- *orders* depuis le traitement polarization à la POLARISATION
- *orders* depuis le traitement laser au LASER
- *orders* depuis le traitement to-altera1 à l'ALTERA
- *orders* depuis le traitement to-altera1 à l'ALTERA
- *data* depuis l'ALTERA1 au traitement from-altera1
- *data* depuis l'ALTERA2 au traitement from-altera2
- *histograms* depuis le traitement grui-server à l'UTILISATEUR.

Attention, les activités intermédiaires ne sont que des conteneurs abstrait dont l'unique rôle est de clarifier la présentation. Autrement dit, au niveaux des noeuds non-terminaux de l'arbre, qui représentent les activités, on ne retrouve aucun traitement associé. C'est au niveau des feuilles de cet arbre, qui représentent les traitements, que l'on trouvera les algorithmes. La description des traitements se fera avec un autre diagramme appelé MCTA.

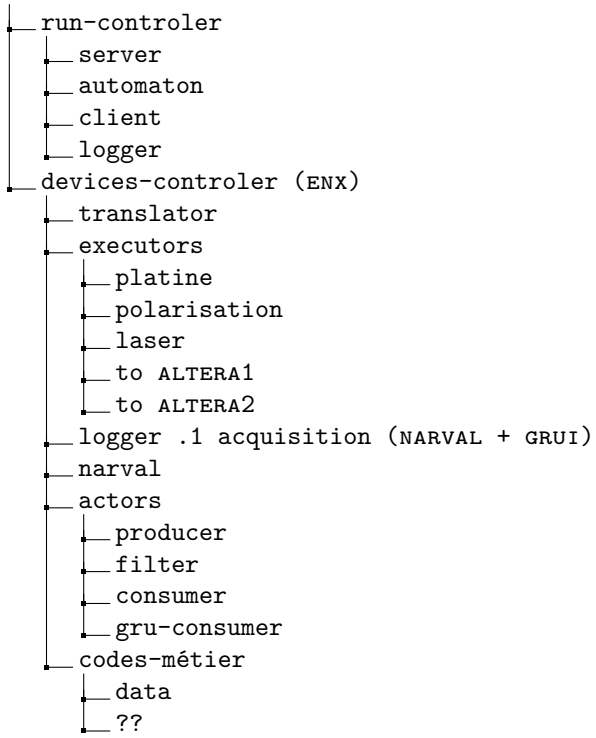


Figure 1: *approche descendante*

Ci-dessous l'énumération des traitements par ordre alphabétique :
acquisition ALTERA1, ALTERA2, consumer data filter gru-consumer laser, logger, narval platine, polarisation, producer translator

5 Implémentation

Info : on utilise la machine *lpnws5210* pour les tests.

- login: root
- passwd: Aur0r3

fichier */etc/apt/sources.list*

```
deb http://security.debian.org/ lenny/updates main contrib
deb http://ftp.fr.debian.org/debian lenny main contrib non-free
```

Comptes :

```
# adduser narval (Narva1)
# adduser enx (3nx)
# adduser intranet (1ntran3t)
# adduser cvs (Cv5)
# addgroup rd
```

5.1 Mise en place d'un serveur CVS

- fichier */etc/profile* :

  ```
  export CVSROOT=/cvsroot
  export CVSEDITOR=/usr/bin/vi
  ```

- fichier */etc/group* :

  ```
  cvs:x:1004:intranet,nroche
  rd:x:1005:
  ```

- fichier */etc/passwd* :

  ```
  nroche:x:1000:1005:nroche,,,:/home/nroche:/bin/bash
  intranet:x:1003:1005:,,,:/home/intranet:/bin/bash
  ```

- archive cvs :

  ```
  # apt-get install cvs
  # ln -s /home/cvs cvsroot
  # chmod g+w /home/cvs
  # su - cvs

  $ cvs init
  $ ls /cvsroot/CVSROOT
  ```

- ajout d'un projet (éventuelement depuis une autre machine) :

  ```
  $ cvs -d :ext:nroche@lpnws5210:/cvsroot import -m "LC intranet" lci/ nroche v1
  ```

- check-out (éventuelement depuis une autre machine) :

  ```
  $ cvs -d :ext:nroche@lpnws5210:/cvsroot co lci
  ```

5.2 Installation de Root

- création d'un compte utilisateur et groupe *tools* via la commande: *adduser tools (password: T00l5)*
- ajouter tous les utilisateurs amenés à utiliser ROOT au groupe tools
- `su - tools`
- télécharger la version voulue de ROOT via les lien disponible sur la page suivante:

  `http://root.cern.ch/drupal/content/downloading-root`
- dans la cas présent:

  `wget ftp://root.cern.ch/root/root_v5.22.00.source.tar.gz`
- `gunzip root_v5.22.00.source.tar.gz`
- `tar xvf root_v5.22.00.source.tar`
- `mv root root_v5.22.00`
- Créer un fichier de configuration des variables d'environnement *ConfigRoot.sh* contenant les lignes suivantes :

```
!/bin/bash
export ROOTSYS=/home/tools/root_v5.22.00
export PATH=$PATH:$ROOTSYS/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:$ROOTSYS/lib
```

- `source ConfigRoot.sh`
- sous distribution DEBIAN, les packages suivants sont requis pour l'installation de ROOT :
  `apt-get install libX11-dev libxft-dev x11proto-xext-dev libxpm-dev libxext-dev libxml2 libxml2-`
- `cd $ROOTSYS`
- `./configure`
- `make`

Les utilisateurs de ROOT auront simplement à faire:

```
source /home/tools/ConfigRoot.sh
```

```
$ wget ftp://root.cern.ch/root/root_v5.18.00.source.tar.gz
$ tar -zxf root_v5.18.00.source.tar.gz
$ mv root root_v5.18.00
```

5.3 Installation de viGru

```
# su - tools
$ wget http://anonymous:anonymous@wiki.ganil.fr/gap/browser/Documents/GRUdoc/package/GRUv_09_03.tar

$ source ConfigRoot.sh
$ tar -xf GRUv_09_03.tar
$ cd GRU
$ make gruso      /* serveur */
$ make vigru      /* client  */
$ linux/vigru
```