Object Oriented Design of VLT software

(1996 VLT Software Status Review, ESO Garching)

G.Chiozzi
24/04/96

The plan to move on OO methodology

Since spring 1994
it has been decided to introduce in the project OO methodologies
taking into account the following constraints:

- **Project already in advanced design and development phase**

- Most team members had:
    - Good experience in the development of real-time control software for telescopes
    - **Small or no experience with OO**

- Compatibility with on-going contracts (external contractors and consortia) not using OO technology

A methodology change could not be requested for these contracts

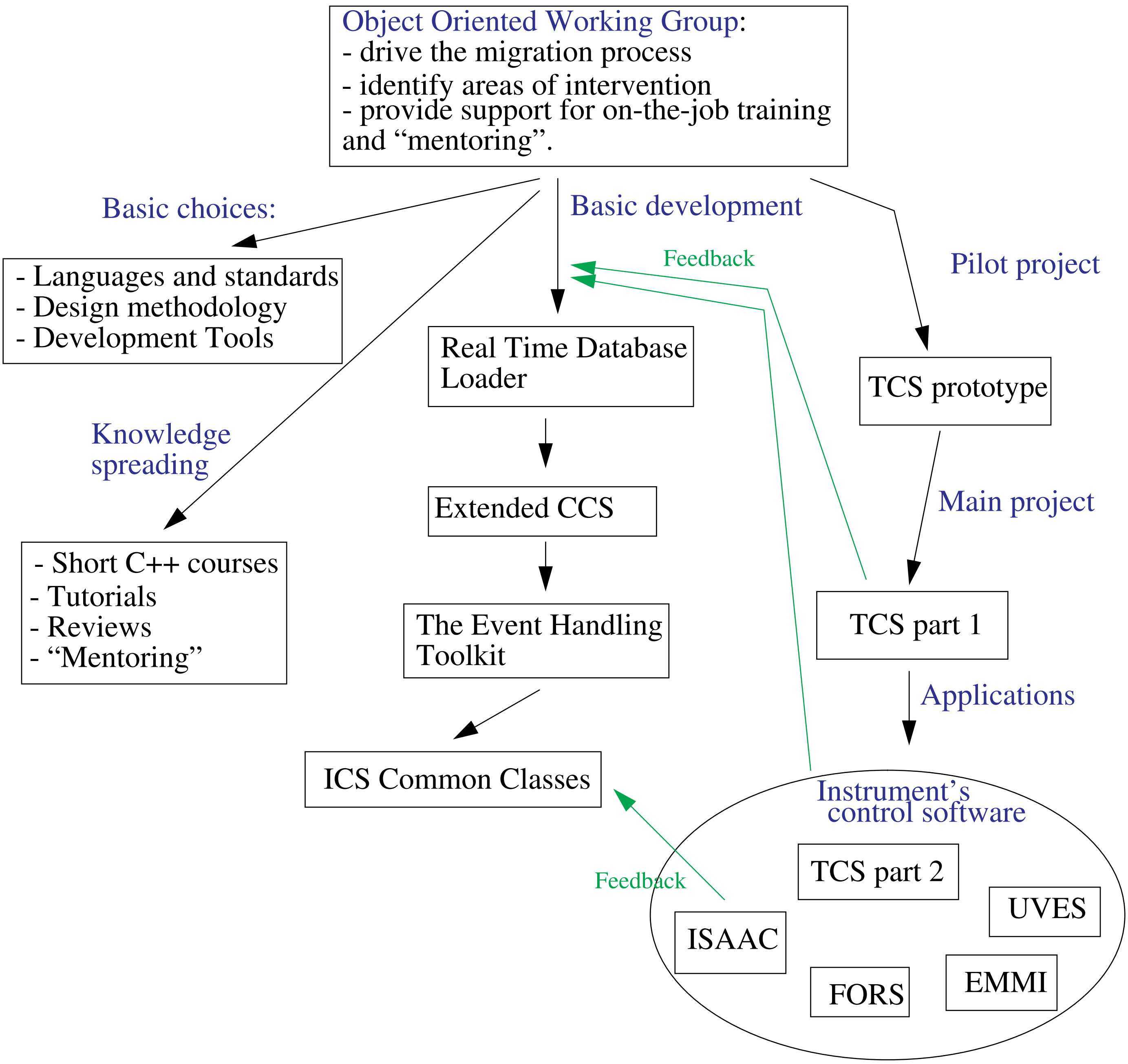
- **To be done without any impact on planning milestones**

The objective

The objective was:

- to introduce the new methodology in a **step-by-step** way
- to have all the people in the group mastering the new methodologies in about **two years**
- having at the same time **developed significant components of the software**.

The strategy

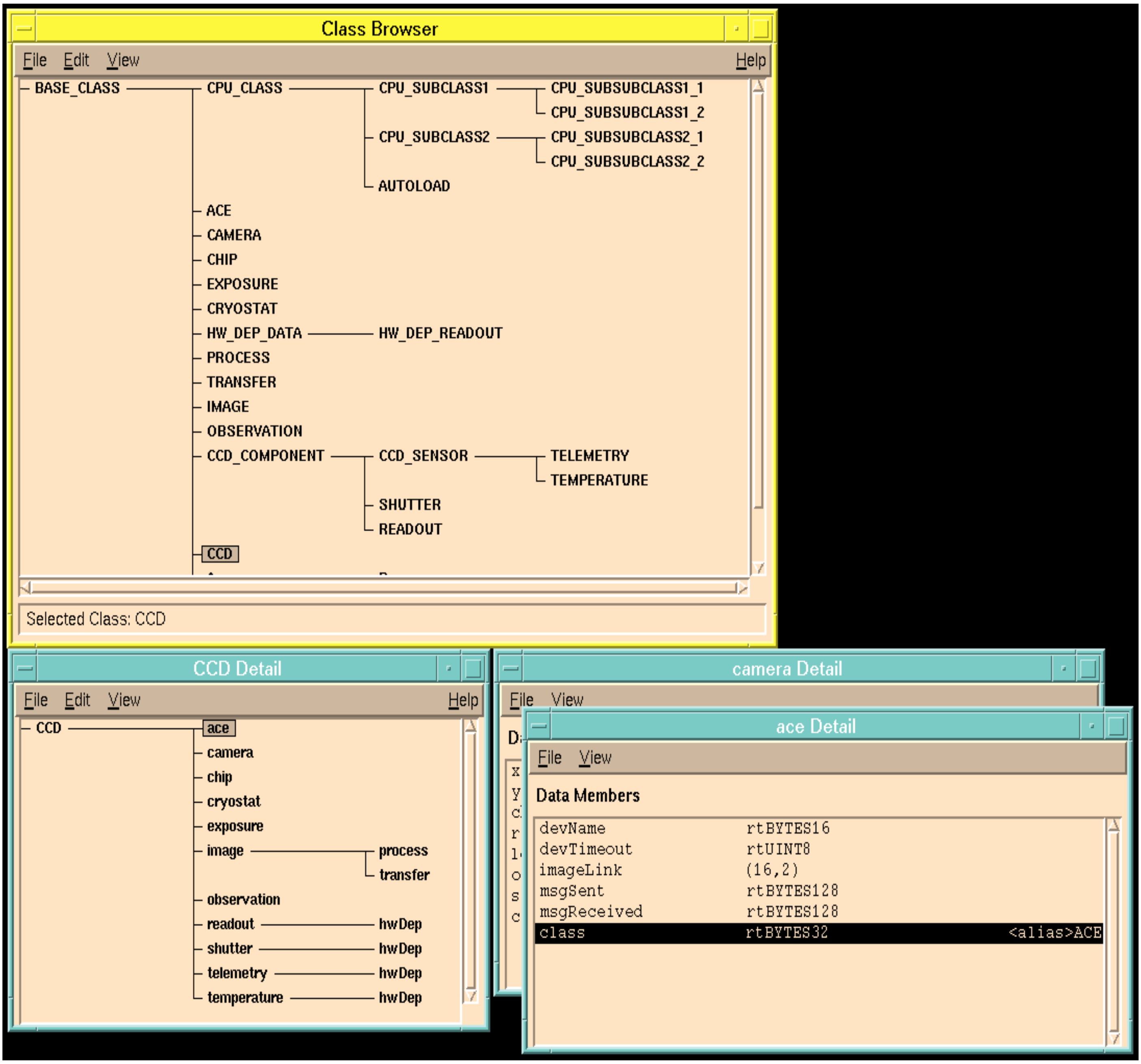


The DBL database loader

All data that can be of interest for external modules, to get a picture of the status of the system or for system's tuning and configuration are stored in the real-time database

The database is a corner-stone of the VLT software architecture and must be uniform and coherent through all the applications developed internally and by consortia, based or not on OO concepts.

- It is a real-time distributed database, based over the hierarchical model
- We have developed a database loader and a class browser that extend the semantic and syntax of the RTAP database definition language introducing OO concepts like:
    - inheritance
    - overloading
    - methods, implemented with C++ classes
    - a "database class" is a new structured data type
- The implementation is based on the commercial product RTAP, from HP. On top of it a preprocessor extend the semantic and syntax of the database definition language introducing the OO concepts.



Class Browser
File Edit View Help
BASE_CLASS
CPU_CLASS
CPU_SUBCLASS1
CPU_SUBSUBCLASS1_1
CPU_SUBSUBCLASS1_2
CPU_SUBCLASS2
CPU_SUBSUBCLASS2_1
CPU_SUBSUBCLASS2_2
AUTOLOAD
ACE
CAMERA
CHIP
EXPOSURE
CRYOSTAT
HW_DEP_DATA
HW_DEP_READOUT
PROCESS
TRANSFER
IMAGE
OBSERVATION
CCD_COMPONENT
CCD_SENSOR
TELEMETRY
TEMPERATURE
SHUTTER
READOUT
CCD
Selected Class: CCD
CCD Detail
File Edit View Help
CCD
ace
camera
chip
cryostat
exposure
image
process
transfer
observation
readout
hwDep
shutter
hwDep
telemetry
hwDep
temperature
hwDep
camera Detail
File View
ace Detail
File View
Data Members
devName rtBYTES16
devTimeout rtUINT8
imageLink (16,2)
msgSent rtBYTES128
msgReceived rtBYTES128
class rtBYTES32 <alias>ACE

Extended CCS

Extended CCS is a library written in C++ on top of CCS. It contains functions and simple classes and requires a minimal knowledge of C++ syntax.

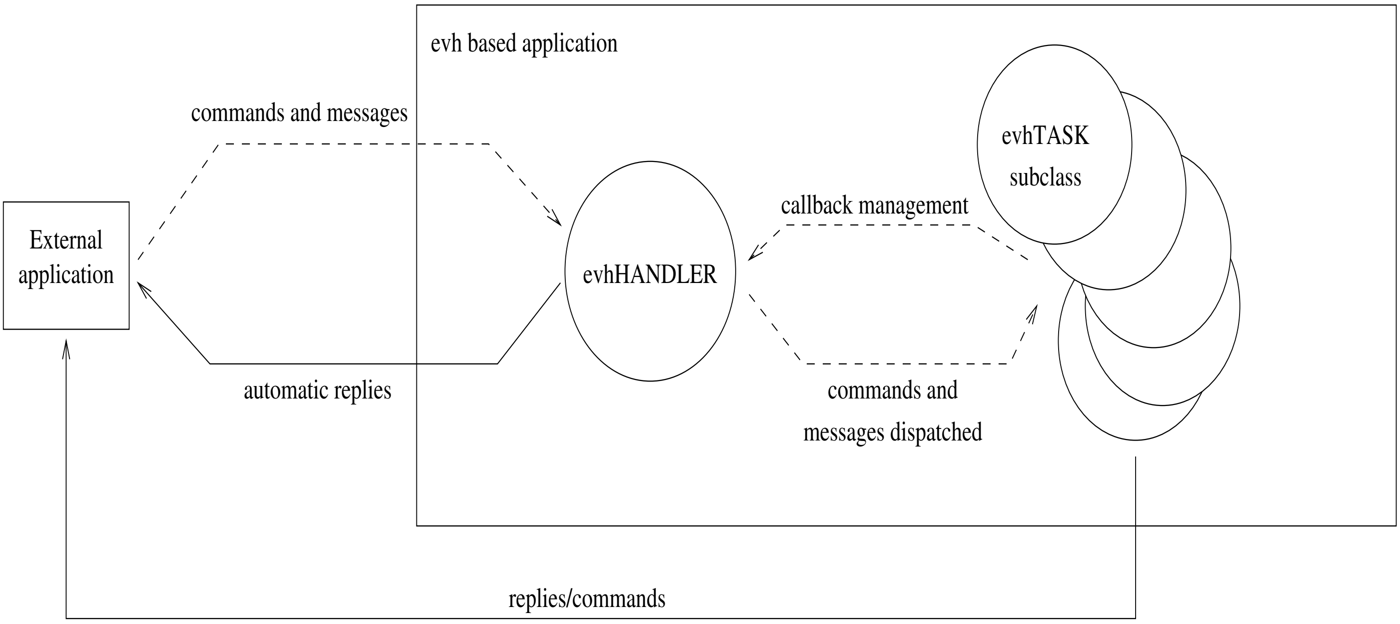
Purpose of the library is:

- Provide easier usage of common CCS features
- Embed in functions or simple classes commonly used application patterns
- Provide base CCS classes for the development of truly object oriented applications
- Prevent common errors in applications
- Standardise produced code

Only the commonly used parts of CCS are integrated in ECCS.

The Event Handling Toolkit

The Event Handling Toolkit is a set of C++ classes to develop event driven applications on workstations.



- Provide a framework to implement applications that receive and send commands and messages.
- Implement standard application patterns
- Prevent common errors in applications
- Support object-oriented design and development

Status

- The Real Time Database of all VLT applications is based on OO concepts and developed with the dbLoader.
- EVH toolkit and database preprocessor have been developed second half ’94, first half ’95 and have been extensively used in many VLT applications.
- OO based TCS part 1 (Presetting, Tracking and Mode Switching) has been released in September ’95 and tested on the NTT in December ‘95. An upgrade is under development.
- As a by-product of the development of TCS part 1, a number of general classes have been developed and added to the set provided by CCS. These classes are now used by other applications.
- Other support libraries for Instrumentation Software have been developed or are under development (OSLX, ICS classes)
- All new applications are currently under development based on these libraries and concepts:
    - TCS part 2 (Auto Guiding, ADC, Pointing)
    - ISAAC Software
    - UVES Software
    - EMMI Software
    - FORS Software

Conclusion

- The OO approach provides a number of benefits:
  - Different applications share the same architecture making easier maintenance in the future
  - Standard and conventions are not only stated "on paper" but automatically enforced by usage of standard classes
  - Important code components are better tested because extensively re-used in many applications
- It has been essential the support provided to implementation by the C++ application framework.
- The new methodology has been accepted much more easily than expected in the software development team and voluntarily adopted by Consortia.