Conditions data: Core SW issues

LCG Calibration Workshop
on the 8 & 9 Dec. 03

Martin Liendl, Ad Aerts, Frank Glege,
Vincenzo Innocente

08.12.2003

Outline:

- Classification of Conditions Data
- Databases, Databases, Databases, Databases
- Foreseen Core SW Services to handle conditions data
- POOL plans

This talk should cover some general remarks on conditions data.
It will then focus on the tasks necessary to incorporate conditions data in the reconstruction, i.e. the SW issues related with COBRA & POOL

In this talk: Core SW == COBRA & POOL integration

Conditions data: Core SW issues

COBRA =
CMS offline framework

ORCA =
CMS reconstruction SW
implemented using
COBRA

Outline:

- Classification of Conditions Data
- Databases, Databases, Databases, Databases
- Foreseen Core SW Services to handle conditions data
- POOL plans

This talk should cover some general remarks on conditions data.
It will then focus on the tasks necessary to incorporate conditions data in the reconstruction, i.e. the SW issues related with COBRA & POOL

In this talk: Core SW == COBRA & POOL integration

Conditions Data: a classification

Does not require sophisticated post-processing

Simple: comes directly from dedicated measurement devices on the detector and can be stored immediately as such, e.g. temperature, pressure

Non event data related: comes from dedicated measurements, but requires data manipulation (statistics, ...), e.g. transparency of ECAL crystals

Requires sophisticated post-processing

Event data related: are calculated from event data; needs ORCA

Indexed by time (timestamp, time of validity, expiration time)

Simple: comes directly from dedicated measurement devices on the detector and can be stored immediately as such, e.g. temperature, pressure

Non event data related: comes from dedicated measurements, but requires data manipulation (statistics, ...), e.g. transparency of ECAL crystals

Indexed by time & version

Event data related: are calculated from event data; needs ORCA

Indexed by time (timestamp, time of validity, expiration time)

Simple: comes directly from dedicated measurement devices on the detector and can be stored immediately as such, e.g. temperature, pressure

Non event data related: comes from dedicated measurements, but requires data manipulation (statistics, ...), e.g. crystal laser monitoring, laser alignment measurements

Will require some kind of meta-data such as:
what were the input data, the algorithms & their parameters?

Event data related: are calculated from event data; needs ORCA

Indexed by time & version

Databases, Databases, Databases, ...

See various talks of Frank Glege for details ...

- We will have 4 kinds of databases in CMS

  (not mentioning event & reco object & analysis objects stores):

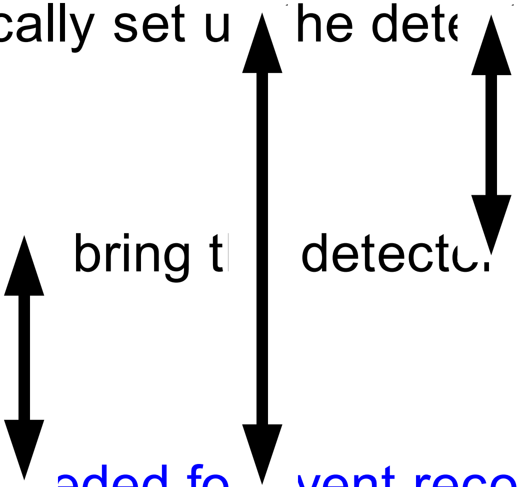
  - Construction databases
    - Holds all information about the sub detector construction up to the start of integration
  - Equipment management database
    - Holds all information to physically set up the detector and is used for asset tracking
  - Configuration database
    - Holds all information required to bring the detector in any running mode
  - **Conditions database**
    - Holds at least all information needed for event reconstruction
    - Used for logging also other conditions data not used in event reco
    - Will be used for error tracking (one example not being event reco)

In some areas, information of different DBs will be related

- We will have 4 kinds of databases in CMS

  (not mentioning event & reco object & analysis objects stores):

  - Construction databases
    - Holds all information about the sub detector construction up to the start of integration
  - Equipment management database
    - Holds all information to physically set up the detector and is used for asset tracking
  - Configuration database
    - Holds all information required to bring the detector in any running mode
  - **Conditions database**
    - Holds at least all information needed for event reconstruction
    - Used for logging also other conditions data not used in event reco
    - Will be used for error tracking (one example not being event reco)



Core SW Services:
will work “only” on parts of the Conditions DB

Read & provide it synchronized with the event being processed

Simple: comes directly from dedicated measurement devices on the detector and can be stored immideately as such, e.g. temperature, pressure

Non event data related: comes from dedicated measurements, but requires data manipulation (statistics, ...), e.g. crystal laser monitoring, laser alignment measurements

Create new versions of conditions data & store them

Event data related: are calculated from event data; needs ORCA

Core SW services will play a role in:

Storing conditions data:
- at “Tier 0”, at “Tier n”
- choice of DB technology

Accessing conditions data:
- from the reconstruction SW, i.e. ORCA
- from SW not related to reconstruction tasks

Distributing conditions data:
- transfering conditions data from “Tier 0” to “Tier n”
- transfering new versions of conditions data from “Tier n” to “Tier 0”

? ? ?

Storing conditions data:

- at “Tier 0”, at “Tier n” ?
- choice of DB technology ? **Relational vs. OO**

Accessing conditions data:

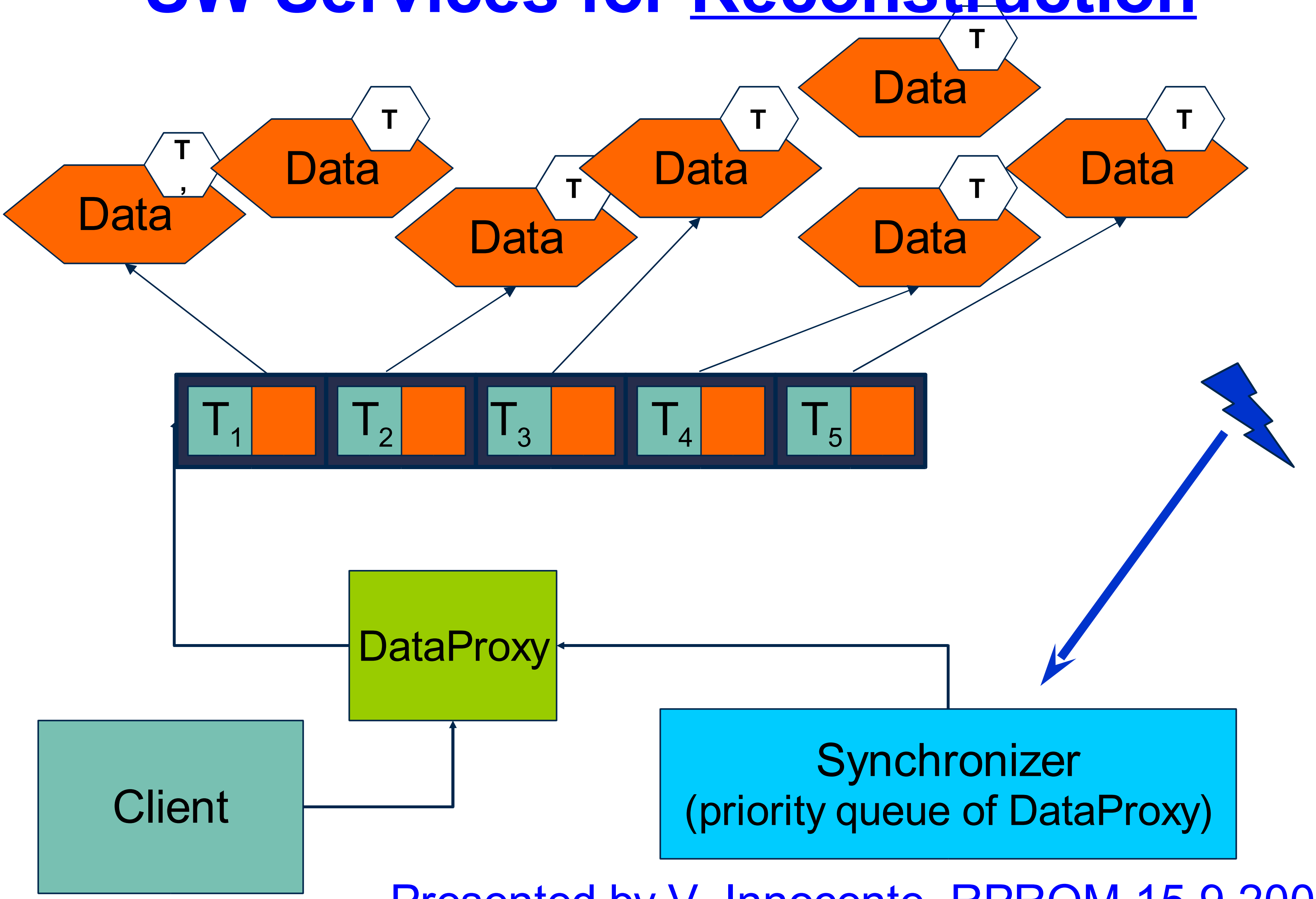
- from the reconstruction SW, i.e. ORCA **via POOL**
- from SW not related to reconstruction tasks ? **depends on DB choice**

Distributing conditions data:

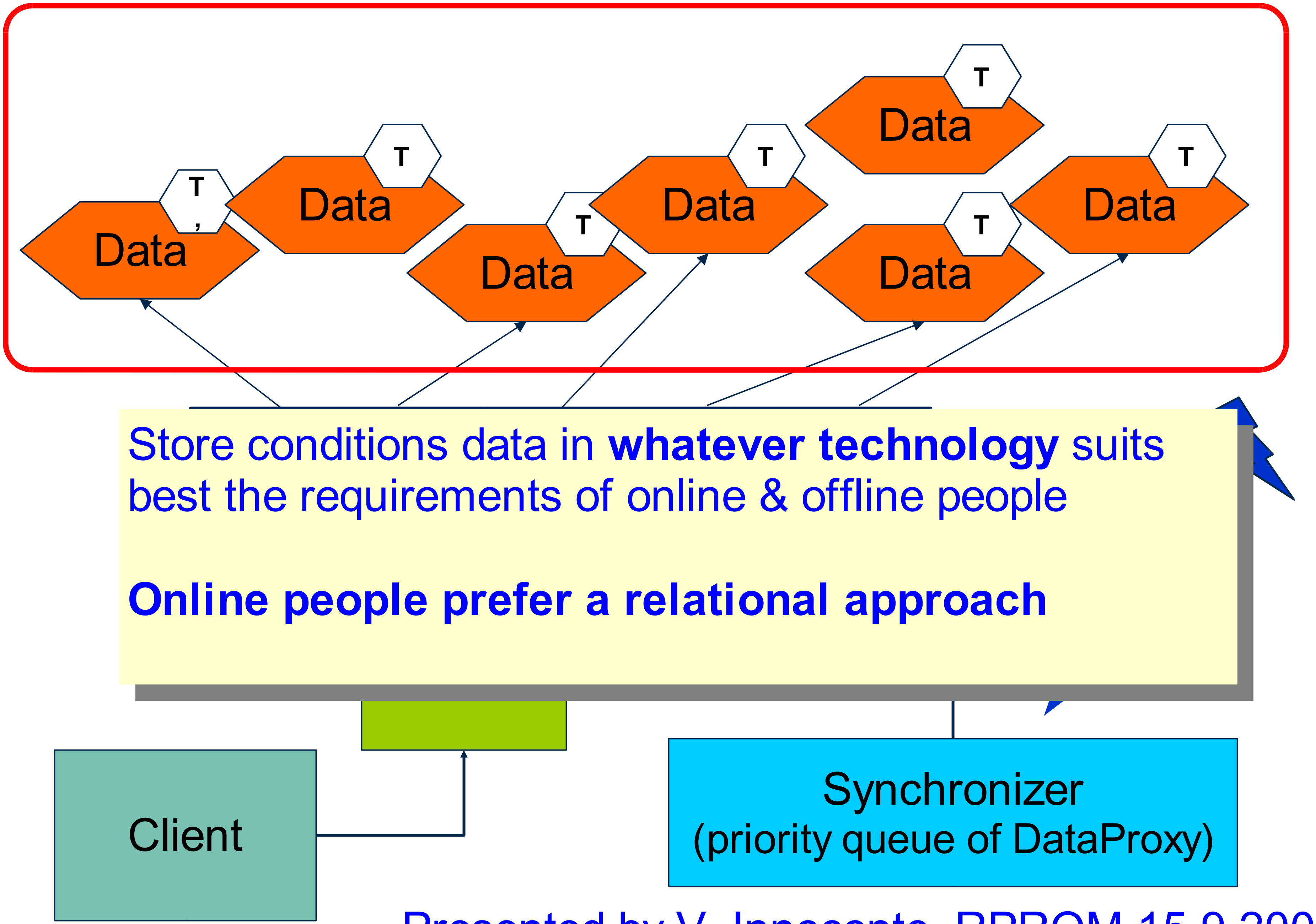
- transfering conditions data from “Tier 0” to “Tier n” ?
- transfering new versions of conditions data from “Tier n” to “Tier 0” ?

In the GRID; task similar to distribution of event & reconstructed data or analysis objects

SW Services for Reconstruction



Presented by V. Innocente, RPROM 15.9.2003

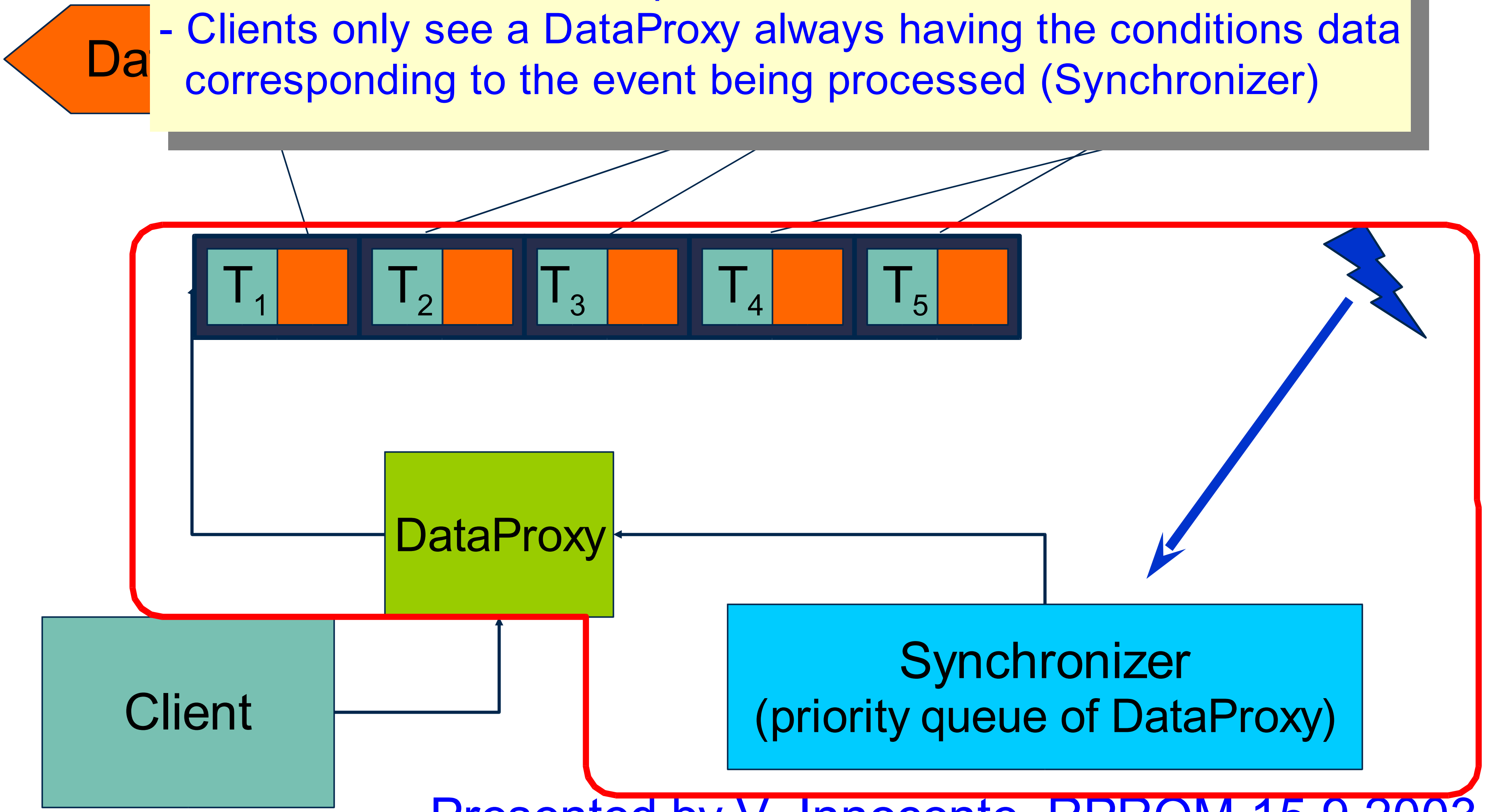


Store conditions data in **whatever technology** suits best the requirements of online & offline people

Online people prefer a relational approach

Presented by V. Innocente, RPROM 15.9.2003

- The COBRA framework then manages a time ordered collections of “pointers” to conditions data. “Dereferencing” the “pointers” depends on the choice of the DB technology.
- One such a collection represents a version.
- Clients only see a DataProxy always having the conditions data corresponding to the event being processed (Synchronizer)



Da

T_1 T_2 T_3 T_4 T_5

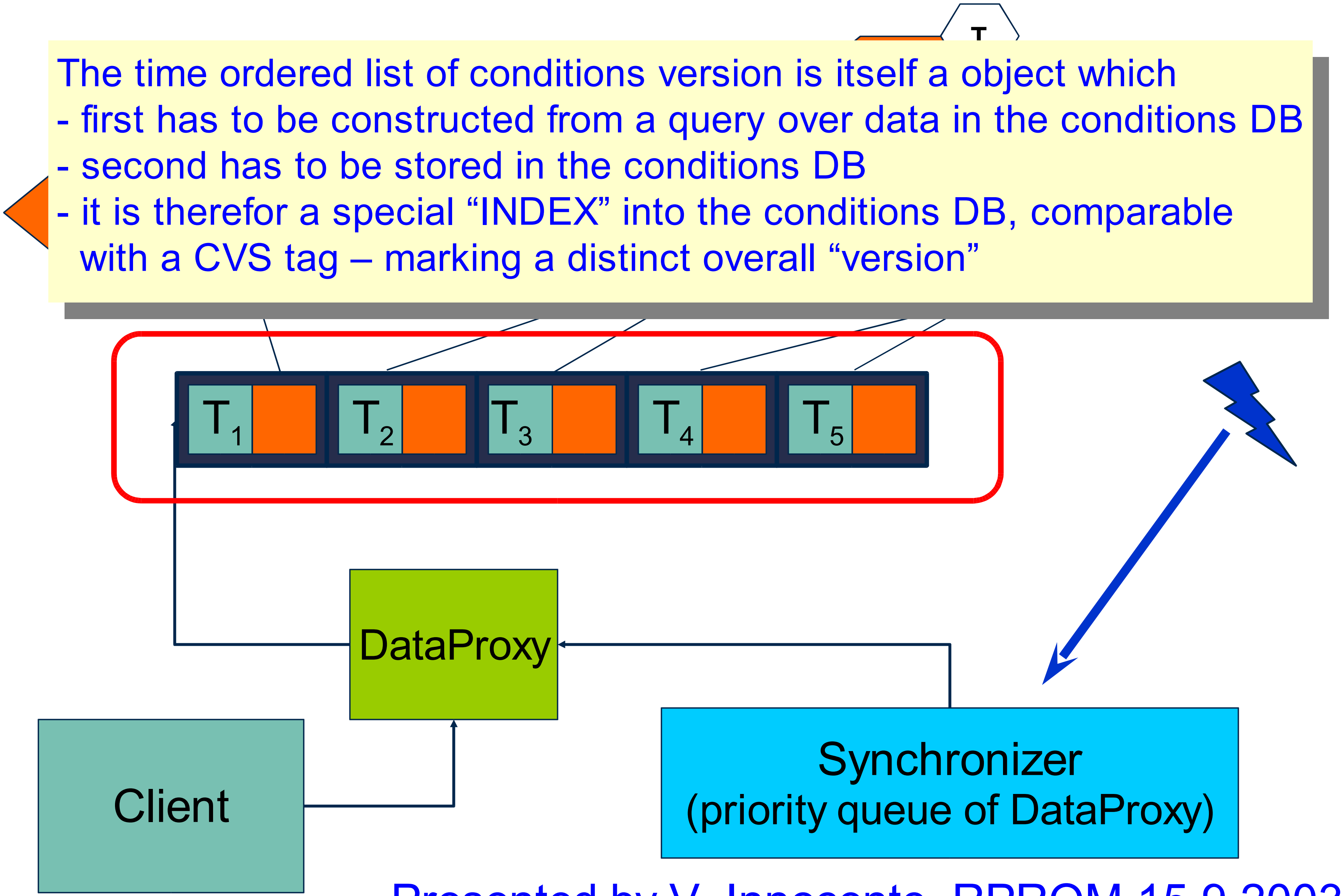
DataProxy

Client

Synchronizer
(priority queue of DataProxy)

Presented by V. Innocente, RPROM 15.9.2003

The time ordered list of conditions version is itself a object which
- first has to be constructed from a query over data in the conditions DB
- second has to be stored in the conditions DB
- it is therefor a special “INDEX” into the conditions DB, comparable with a CVS tag – marking a distinct overall “version”



DataProxy

Client

Synchronizer
(priority queue of DataProxy)

Presented by V. Innocente, RPROM 15.9.2003

SW services for writing conditions

Only affects the 3rd kind of conditions data – event data related

Simple: comes directly from dedicated measurement devices on the detector and can be stored immediately as such, e.g. temperature, pressure

Non event data related: comes from dedicated measurements, but requires data manipulation (statistics, ...), e.g. transparency of ECAL crystals

Conditions object is a POOL object;
POOL stores it in the Conditions DB

Event data ata; needs ORCA

Remarks on data models & storage technology

From the Offline people's point of view:

- only conditions data needed for reconstruction is of interest
- acessed (read/write) via COBRA/POOL & therefore independent of the storage backend
- Data models are class models; conditions data is more struct-like
- Vincenzo has allready prototype for conditions data using POOL,ROOT

From the Online people's point of view:

- broader scope of what to store in the Conditions DB (logging, error tracking, ...)
- have/use applications not built on COBRA
- are more data oriented vs. object-oriented
- want to benefit from RDBMS features such as built in ensurance of referential integrity
- want to define data models **without** defining classes

From the COBRA framework point of view:

- COBRA uses POOL with ROOT as storage backend
- all persistent objects, no matter how they are physically stored, w
- V OOT
- fa e P

Fr

- h
- a

The way we want to go is:

- allow for a relational storage backend, so that relational models for conditions can be specified
- have a POOL which is able to access data stored using a relational schema & not a class definition; must be non intrusive to the DB design!

- want to benefit from RDBMS features such as built in ensurance of referential integrity.
- want to define data models without defining classes
- more concrete: they want ORACLE

First discussions with POOL team

(Thanks to Ioannis & Dirk!)

- POOL will provide a non-intrusive solution to access relational data through C++ objects
- In the beginning support for simple tables related to each other via primary keys, foreign keys
- In order to tell POOL whether an object A has a pointer to another object B, whether the pointer is owned by A, or whether A aggregates B, or whether B inherits from A (or vice verse), additional information has to be provided on how to interpret foreign keys.This information will be stored non-intrusively in seperate tables in the DB. Reasonable default behavior should be provided.

CMS Requirements for LCG:

o) have the POOL suport for relational DB access dicussed before

o) Of course, we have the notion of 'time of validity'/ time-stamps, versions, tags;
A common solution tackling these issues would help us in implementing conditions in our framework COBRA;

o) Apart from that: we want to be “free” to design a relational model for conditions (req. mainly from our online community)

o) Present stage: CMS started to collect requirements only after the LCG Conditions DB project was kicked off;
Currently, we don't have yet much expierence or many prototypes ...; More requirements will be clearer once we have gained more insight by developing/using better prototypes!