Summary of the CINT/Reflex workshop 2-6 May 2005

Rene Brun

4 May 2005

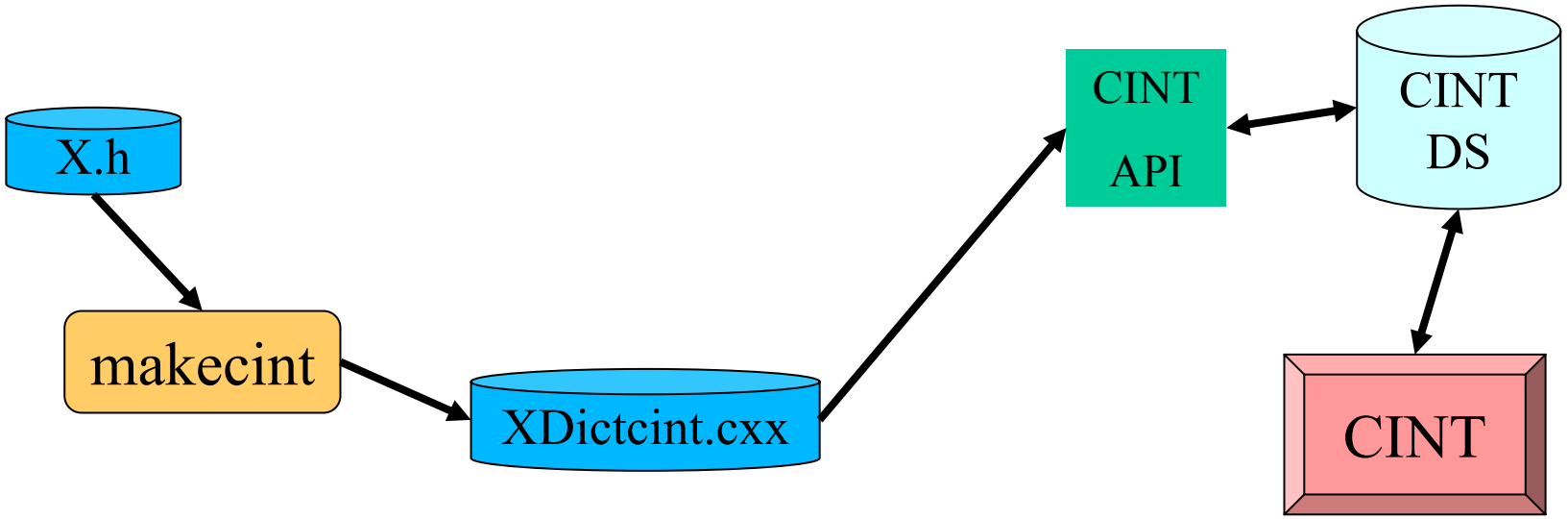
http://agenda.cern.ch/fullAgenda.php?ida=a052473

Participants

- Rene Brun
- Philippe Canal
- Markus Frank
- Masa Goto
- Giacomo Govi
- Wim Lavrijsen
- Pere Mato
- Fons Rademakers
- Stefan Roiser

Makecint (since CINT early days)

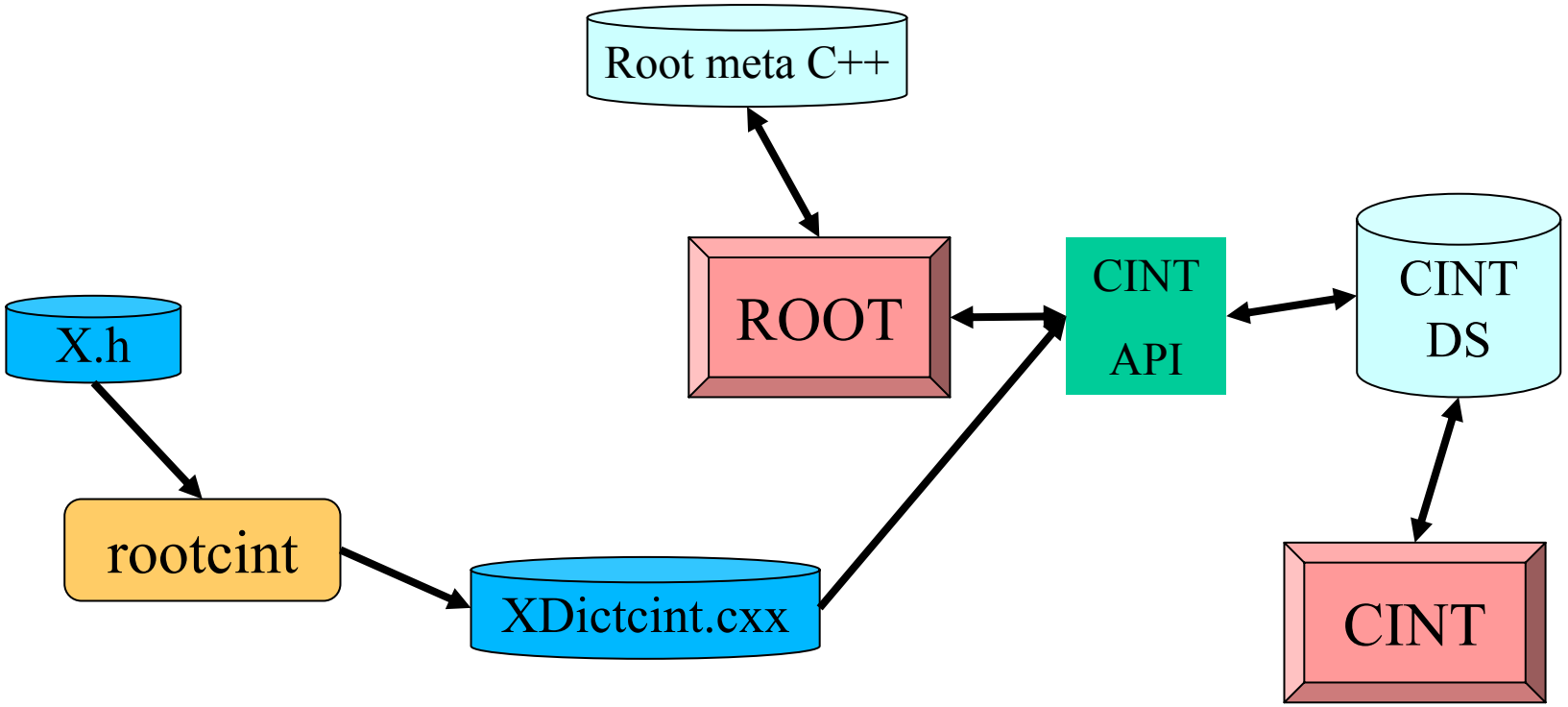
Generates stub functions such that compiled code can be called from interpreted code.



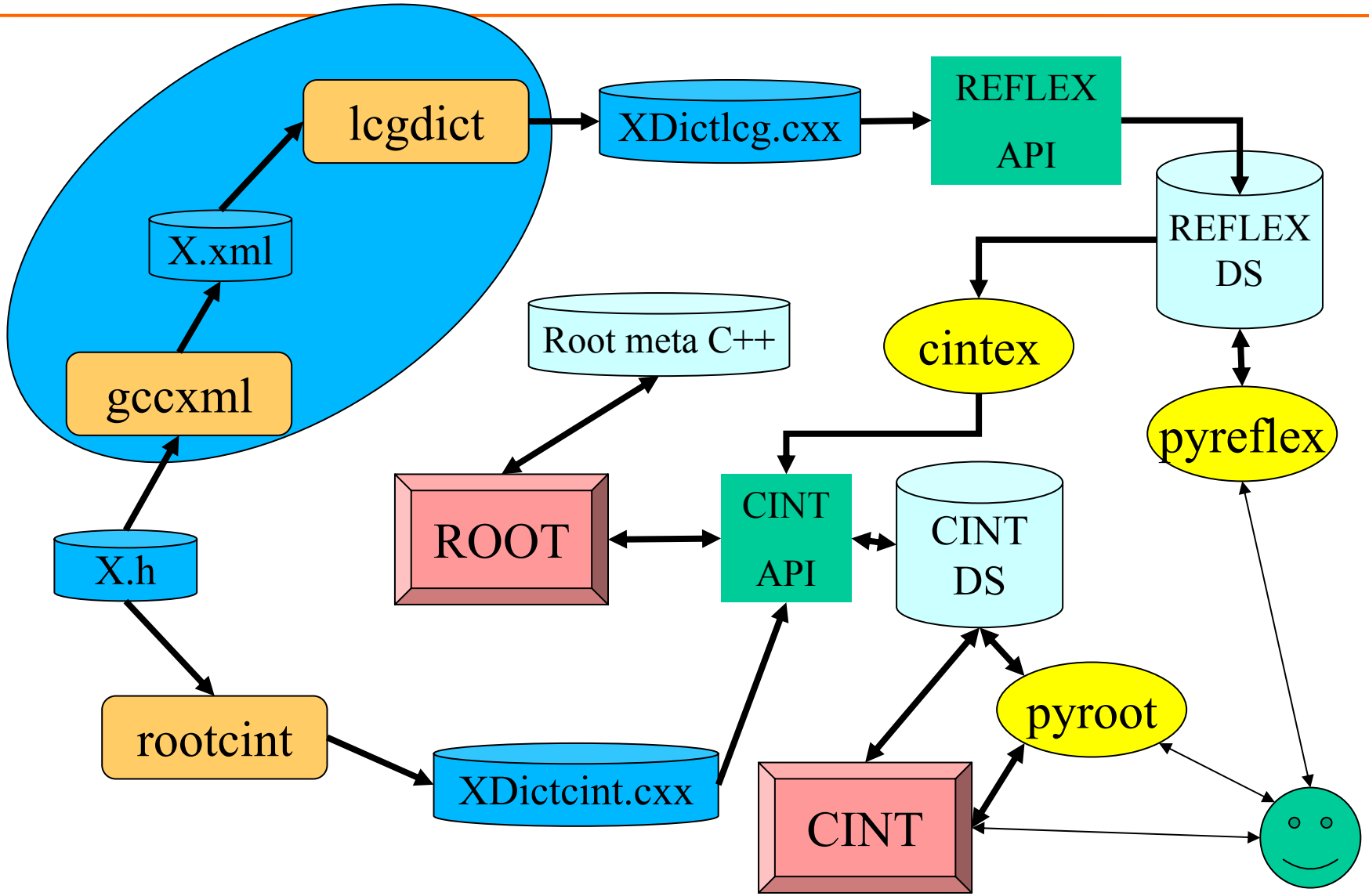
Rootcint (1996)

rootcint extends makecint to generate the I/O functions and ShowMembers

ROOT meta classes get information from CINT. They contain additional info useful for the I/O and schema evolution.



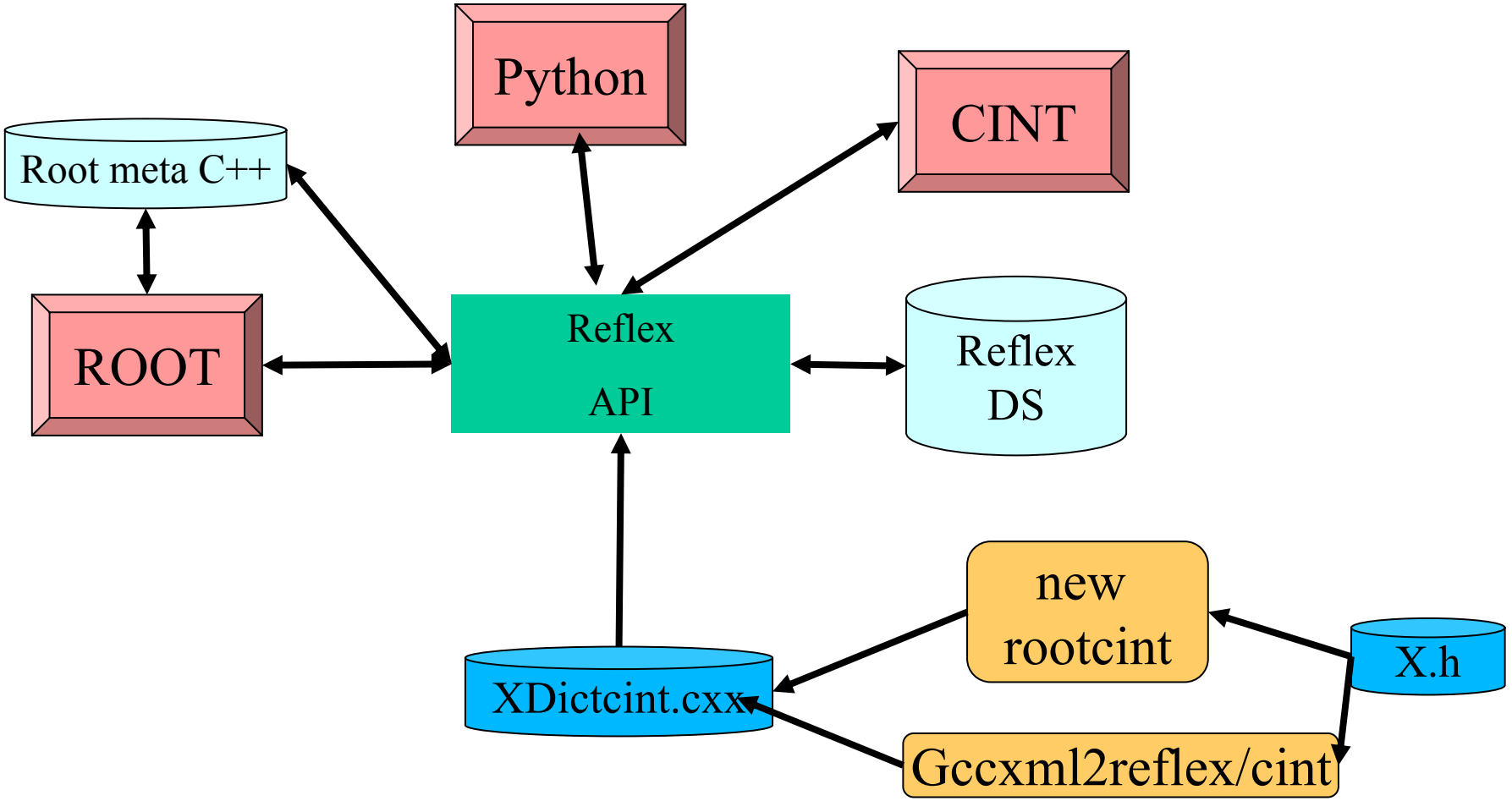
Dictionaries : situation today



Proposal

- We are proposing to cooperate with Masa for the development of a new API and data structure supporting all the features of the current C++.
- This package could be based on a merge of the existing Reflex system with the new version of CINT developed by Masa.
- This new package will be distributed with CINT and usable with non-ROOT applications.

Dictionaries : where we would like to go



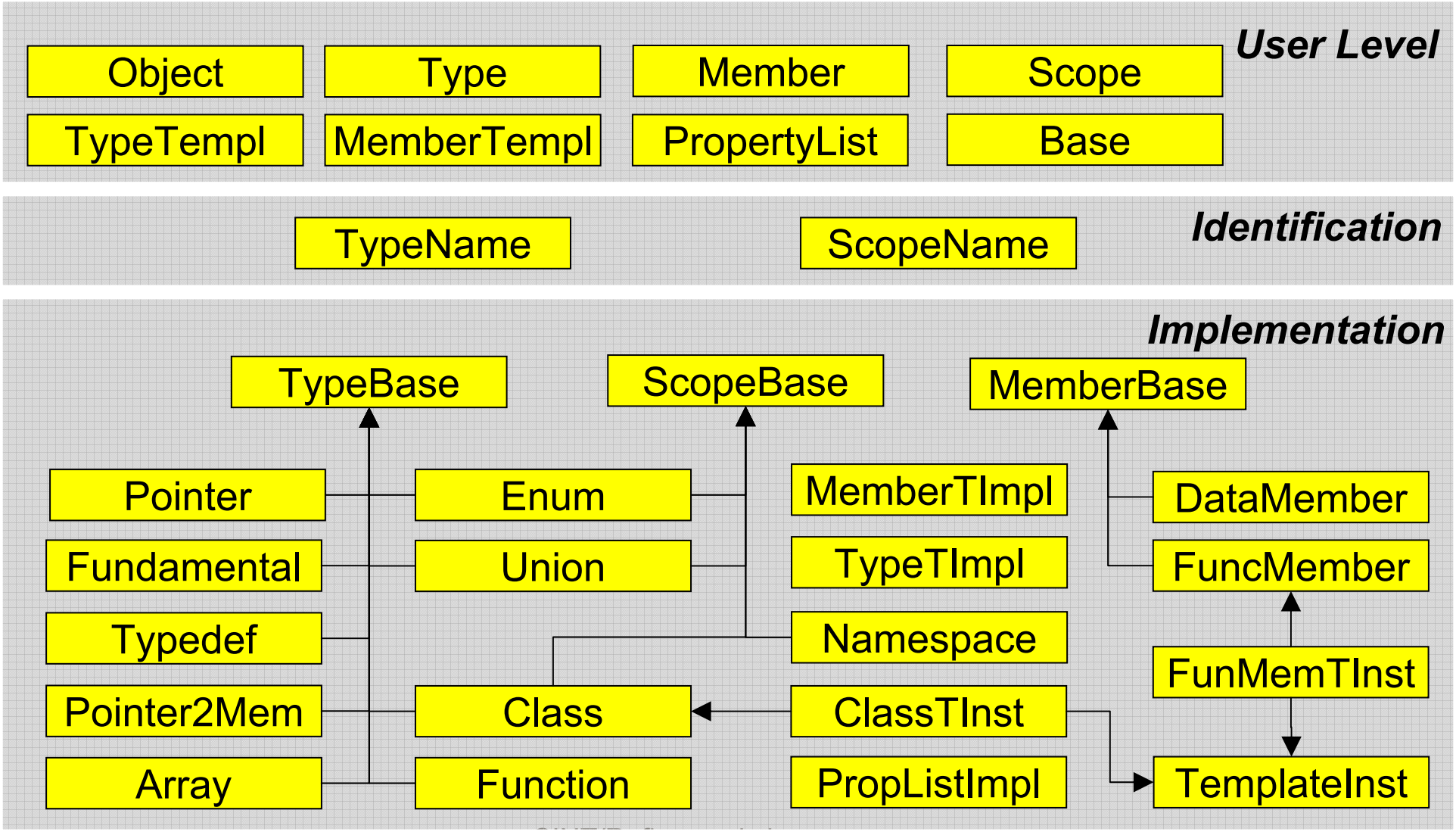
New CINT version 6 by Masa

- Masa is working since one year on a reengineering of the CINT byte-code compiler and executor.
  - Half-way done
  - Simplified architecture
  - Easier team development
- Unique occasion to introduce a new data structure.

Status of Reflex by Stefan

- Nice poster with the ULM description, see simplified version in next slide.
- Reflex has been run on the ROOT TGeo classes (80 classes) and cintex used to execute successfully the stressgeom.cxx test.

Reflex Model



Stressgeom Comparison Test

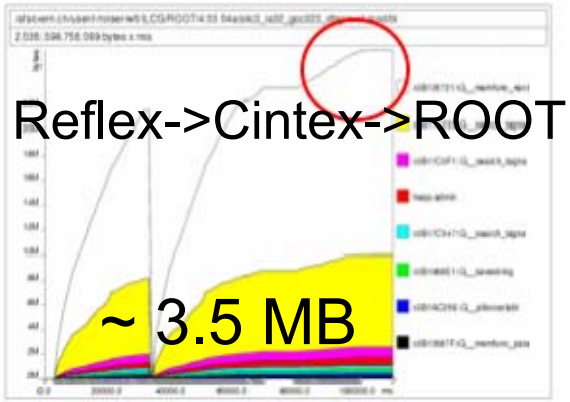
- Idea
  1. Build Reflex dictionary for ROOT geom package
  2. Load Root Geom dictionary information through Cintex into the ROOT framework
  3. Run tests/stressgeom.cxx with CINT
  4. Compare to native Root performance
     - In CPU time / Rootmarks
     - In memory allocation
- Stressgeom.cxx
  - Generates 1 Mio random points, computes volumes, tracking of 100 K random rays
  - 322 lines of code

stressgeom.cxx.diff

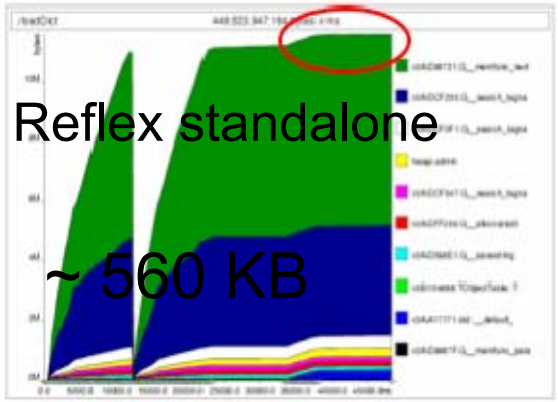
[…]	[…]
`ratio = Double_t(iin)/ Double_t(ngen);`	`ratio = iin/ngen;`
[…]	[…]
`void stressgeom()`	`void stressgeom_reflex()`
[…]	[…]
`gSystem->Load("libGeom");`	`gSystem->Load("liblcg_Cintex");` `gSystem->Load("libGeom_Rflx");`
[…]	[…]

Memory allocation

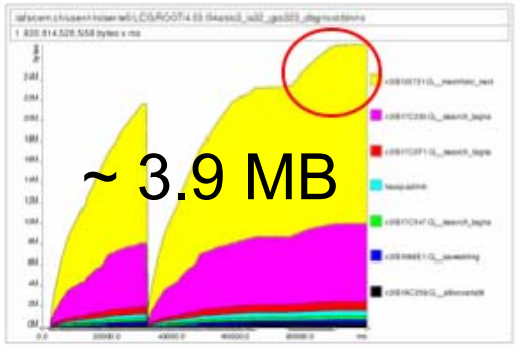
Reflex->Cintex->ROOT



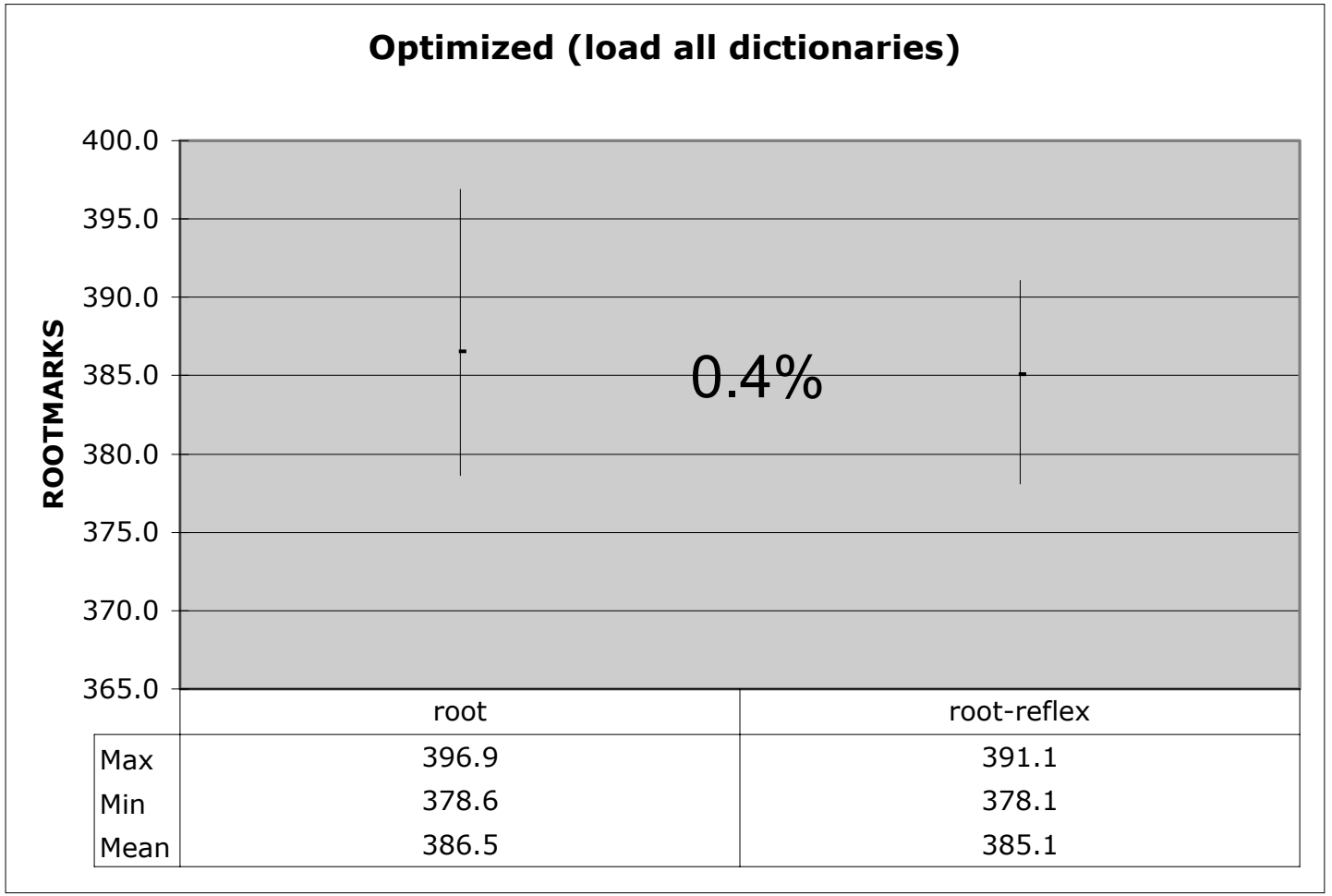
Reflex standalone



ROOT standalone



ROOTMARKS



	root	root-reflex
Max	396.9	391.1
Min	378.6	378.1
Mean	386.5	385.1

POOL requirements by Markus

- **Fast type resolution for C++ pointers (IsA)**
- **Fast access to constructor/destructor ideally compiled**
- **Check for dictionary completeness**
- **CINT as an interpreter, ROOT for I/O and Reflex need to agree on a common way to name classes, constructors and destructors.**

```
std::vector<std::pair<std::string, std::size_t> >
vs.
vector<pair<basic_string<char>, unsigned long> >
```

POOL requirements by Giacomo - 1

- **The LCG Dictionary has a very important role in the POOL architecture, in particular in the Storage Manager area.**
- **Currently both ROOT and RDBMS backends rely on the seal::reflect package for the implementation of Object streaming.**
- **In particular, RDBMS backend uses the reflection to convert data from the C++ layout to the RDBMS entities (Tables and columns).**
- **In addition, all of Storage Manager components somehow depend on the seal::reflect in their interfaces (arbitrary object are exchanged between the components as pair of void* + reflect::Class*).**
- **Some POOL functionalities (Custom Shape Transformation and Token type checking) are currently based on the dictionary lookup by Guid, which is provided in the seal::reflect package.**
- **Other functionality of the DataSvc (type chekcing, casting, memory de-allocation in Ref<T>) are strongly based on seal::reflect.**

POOL requirements by Giacomo - 2

- **From the POOL point of view, the concerns regarding the various proposals for the merge of the Reflex package in Root are the following:**
  - **The use of dictionary (reflect or reflex) is not confined in the Root backend area. It would be desirable that re-packaging required by the merge SEAL+ROOT minimizes the introduction of unnecessary new dependencies for the users.**
  - **POOL couldbenefit from the re-engeering of CINT for the use of Reflex data structure:**

    reduced memory consumption of the two dictionaries

    performance improvement from optimization of some Reflex, CINT and Root code
  - **POOL consider acceptable a phase with the system based on Reflex+Cintex+CINT, with the two dictionaries in memory (which corresponds to the reflect+CINTBridge+CINT used until now), to be used until the new CINT based on the Reflex data structure is available.**
  - **The migration from Reflect to Reflex will require some work in all the Storage Manager component, and in particular in RootStorageSvc, ObjectRelationalAccess, PersistencySvc and DataSvc.**
  - **POOL functionality requires the possibility to make a dictionary lookup by Guid. It would be desirable to maintain the functionality previously supported in Reflect.**

Python aspects by Wim Lavrijsen

Dictionary completeness (C++)

- API for modifying dictionaries
- Fill Reflex dictionary from Python class
- Access to auto generated methods
- Access to global namespace
- Call backs with user parameters
- Call back for dictionary changes
- Remove distinction between interpreted and compiled classes.
- Memory management support
- Error messages & exceptions

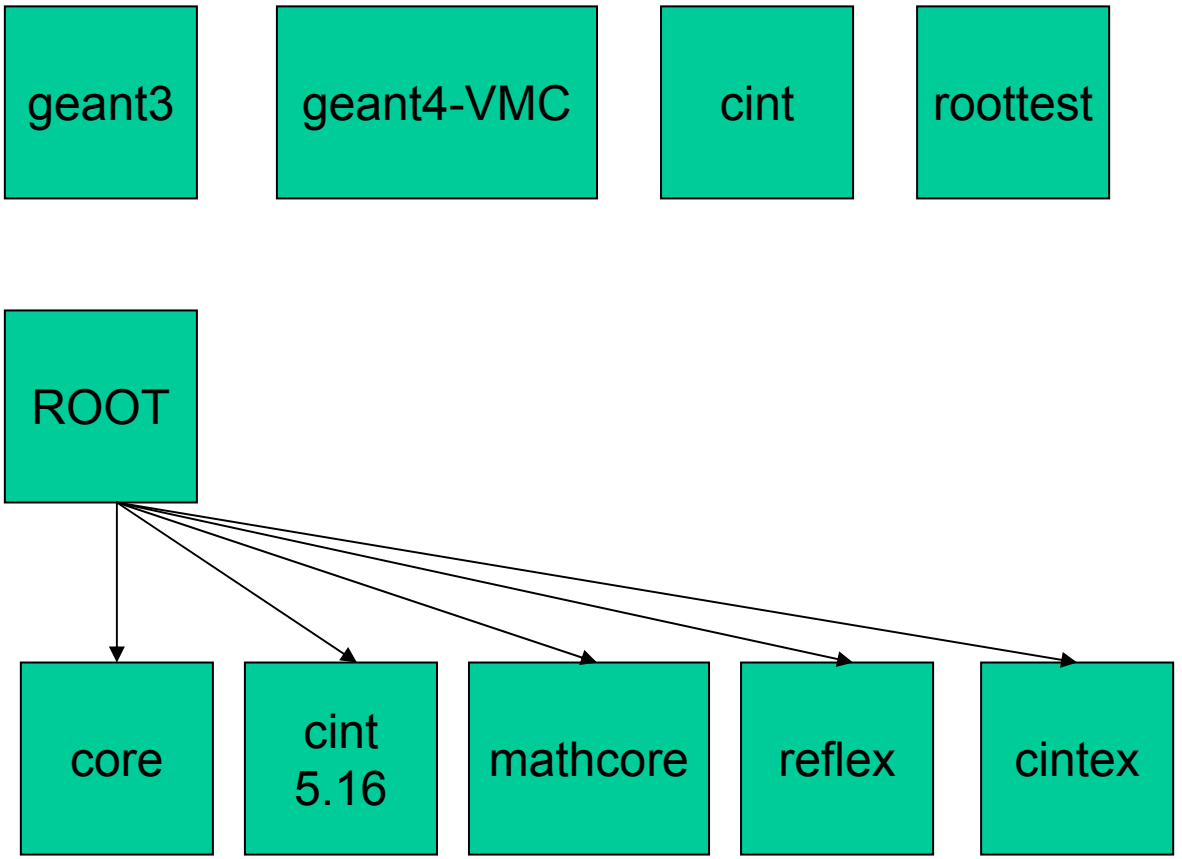
CINT & Reflex

- CINT 6 will use Reflex as the in memory Type database
- Fermilab and CERN will be responsible for the code modifications needed to adapt CINT to use Reflex.
- In parallel, Masa Goto will continue the development of the new bytecode compiler.

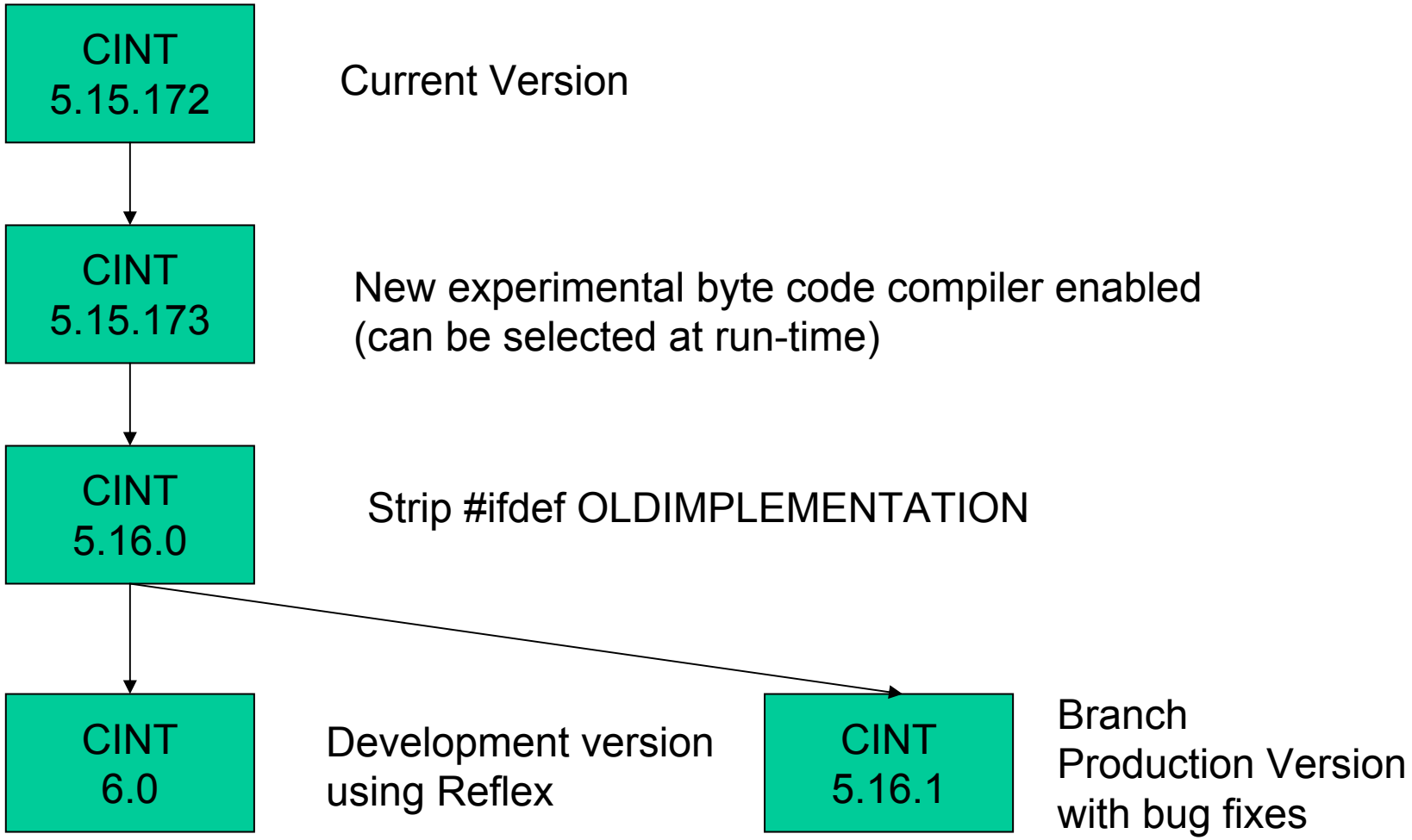
CINT & Reflex

- ROOT coding conventions have been updated to allow for namespace
- Reflex will be adapted to the ROOT new coding conventions and uploaded in the ROOT CVS.
- CINT and Cintex will be uploaded in their own CVS modules.

CVS Structure



CINT Cvs


CINT
5.15.172
Current Version
CINT
5.15.173
New experimental byte code compiler enabled
(can be selected at run-time)
CINT
5.16.0
Strip #ifdef OLDIMPLEMENTATION
CINT
6.0
Development version
using Reflex
CINT
5.16.1
Branch
Production Version
with bug fixes

Add Coding Convention (ROOT v5)

- All ROOT classes will be in the namespace ROOT
- Class name and Namespace name have to start with an uppercase letter
- Classes starting with T can be in ROOT namespace
- Classes within a sub-namespace (like Reflex) do not have to start with T
- In code class starting with T can be used directly
- In code class not starting with T, having to be qualified by their sub-namespace.
  - Example:
    - allowed: Reflex::Object myobject; TObject *obj;
    - not allowed: using namespace Reflex; Object *obj;

Advantages of the merge

- Advantage for ROOT/POOL users:
- Better C++ standard compliance
- Better header files parsing
- Better performance
- More reliable system

- Advantage for CINT development:
- Easier maintenance
- Enhanced/simpler Dictionary API
- Increase support team

plans

- cint , reflex & cintex in ROOT CVS immediately (cint already done)
- June dev release will have
  - Reflex
  - Cintex
  - MathCore
- First visible results of the CINT/Reflex merge at the time of the ROOT workshop (28,29,30 september)
- Pro release in December

Summary of Summary

- The workshop has been a big success and an important opportunity for people to exchange their views and know each other a bit better.
- Important decisions have been taken
- A lot of work in front of us.
- First positive results should be visible very soon.