Seal Dictionary

Applications Area Internal Review
20 October 2003
Stefan Roiser / CERN

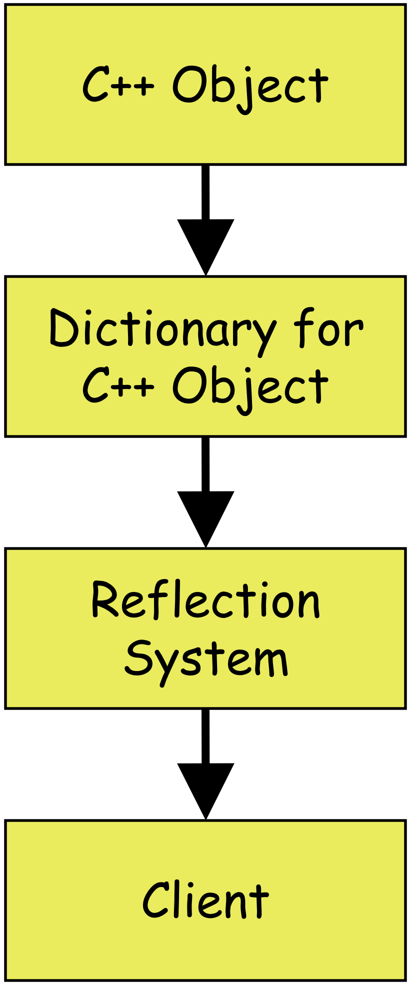
Motivation

- The Architecture Blueprint RTAG proposes the use of object dictionaries for
  - object streamers, object browsers, debuggers
  - rapid prototyping applications (e.g. Python)
  - runtime discovery of interfaces
- Modern languages provide reflection inherently
  - In C++ reflection is very limited (RTTI)
- The basic C++ concepts shall be supported
  - e.g. inheritance, methods, data members, accessibility, templates

Packages in Seal Dictionary

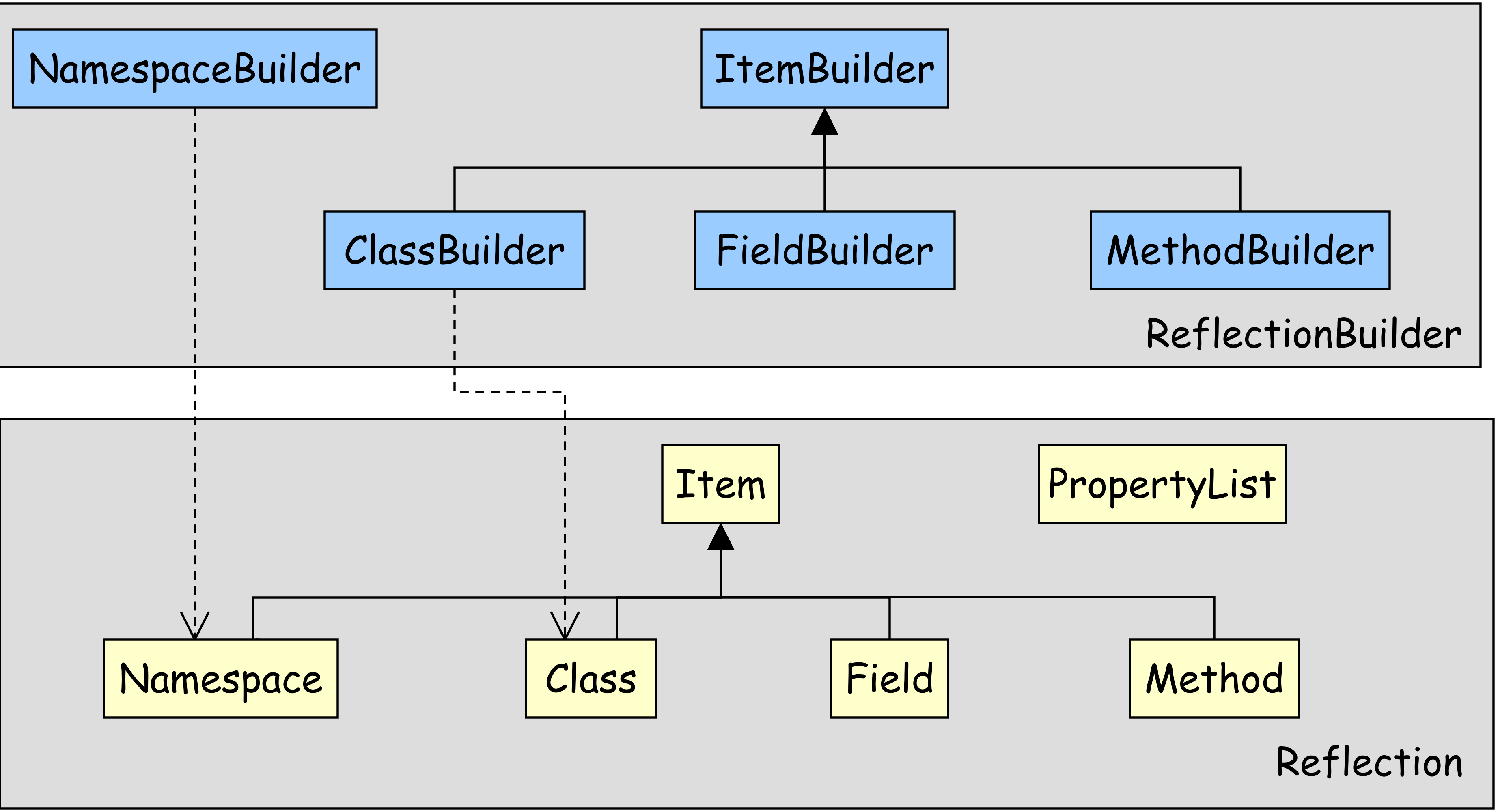
- Reflection packages
  - DictionaryGenerator for producing sources
  - ReflectionBuilder, the loading interface
  - Reflection, the user API
- Dictionary examples
  - CMS
  - ATLAS (not yet in cvs)
- Standard Dictionaries
  - CLHEP: Random, Vector
  - STL: Vector, List, String
  - Dictionary: Reflection

Overall View



- Build dictionary for C++ object
- Load information about the object into the reflection model
- Provide meta information about the object to the user

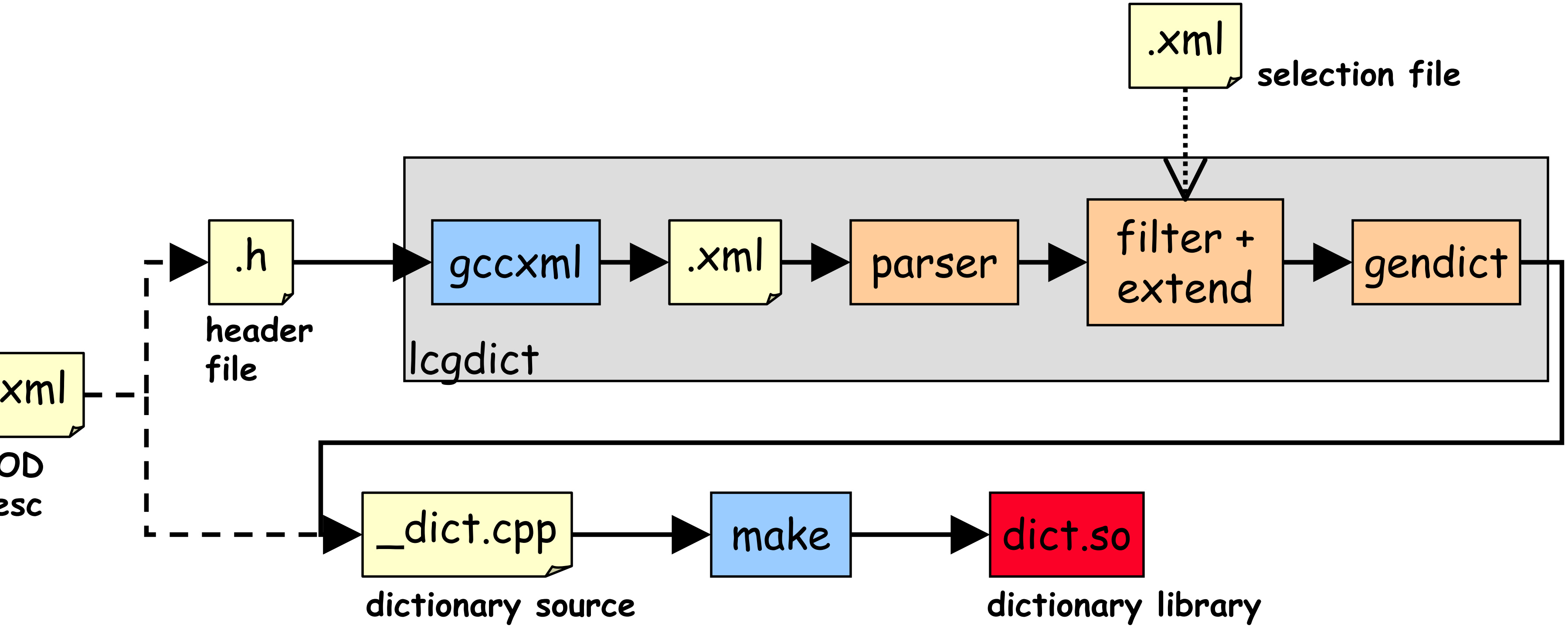
Reflection Model Class Diagram



Usage Example

```
Class mc* = Class::forName(“Foo”);
void* fooInst = mc->instantiate();

std::vector<Field*> fields = mc->fields();
std::string f0name = fields[0]->name();
std::string f0type = fields[0]->type()->name();
fields[0].set(fooInst, 4711);

Method* m0meth = mc->method(“getBar”);
int m0val = m0meth->invoke(fooInst);
```

Parse C++ Header Files



.xml

selection file

.h

header file

gccxml

.xml

parser

filter + extend

gendict

lcgdict

xml

OD esc

_dict.cpp

dictionary source

make

dict.so

dictionary library

Example: lcgdict Foo.h –-select=FooSel.xml --deep –I/home/lcg/include

How to build Dictionaries

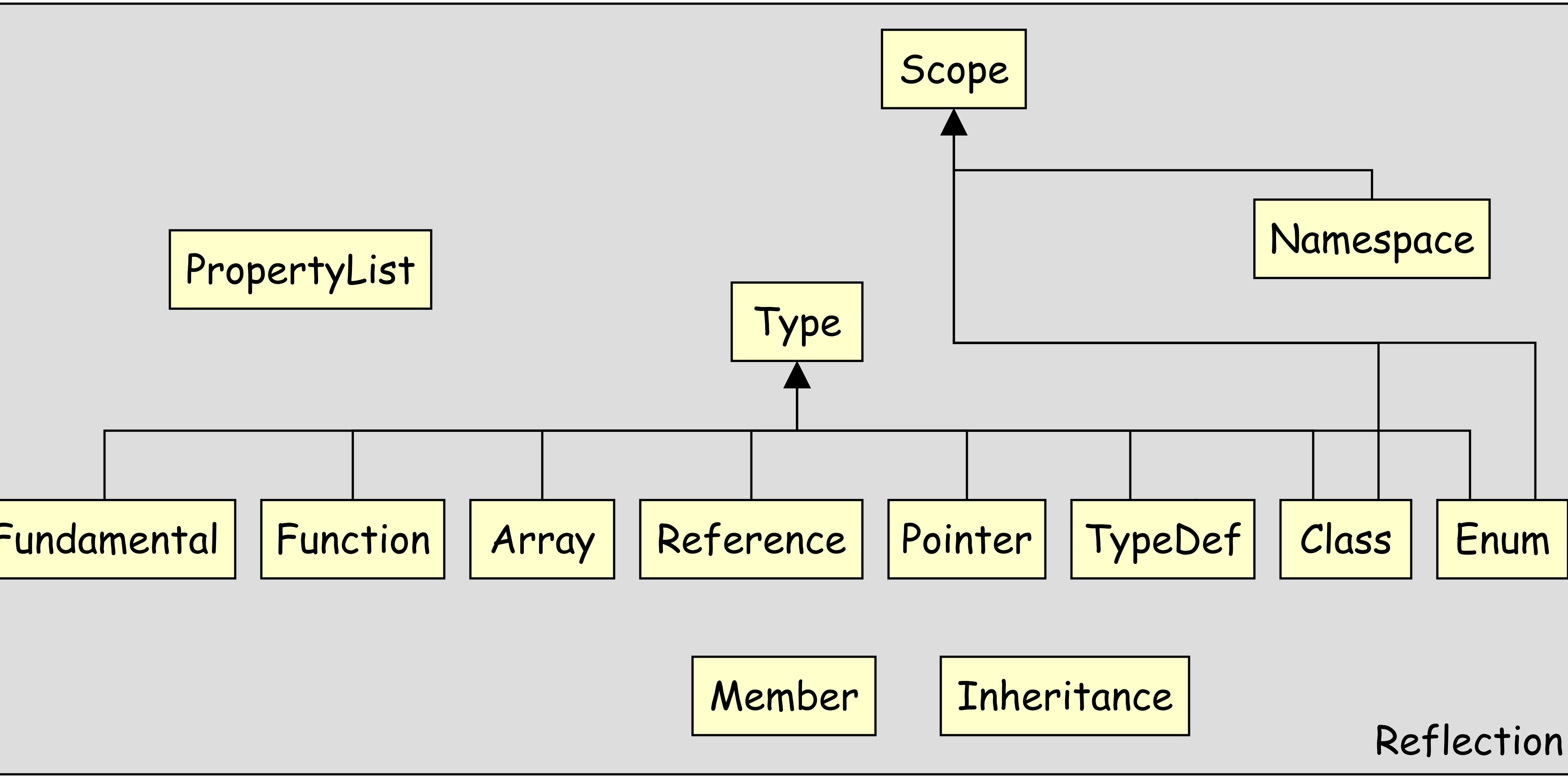
- Parse C++ header files
  - lcgdict
    - » i.e. gccxml + python script
- Other means
  - Derive from XML descriptions
    - » LHCb approach
  - Write by hand
    - » design was made with simple syntax in mind

```
#include “Foo.h”

class Foo_dict {
public:
  Foo_dict();
};

Foo_dict::Foo_dict() {
ClassBuilder(“Foo”,”desc of Foo”,
    typeid(Foo),sizeof(Foo)).
addField(“bar”,”int”,”desc of bar”,
    OffsetOf(Foo,bar)).
addMethod(“getBar”,“desc of getBar”,
    “int”, Foo_getBar).
build();
}

static Foo_dict inst;
```

Who is using Dictionaries

- POOL
  - DataService
    - » Casting, type-checking and deleting of objects
  - StorageService
    - » Provide meta-information of objects when writing
    - » Instantiating objects when reading, type-checking
- SEAL
  - PyLCGDict
    - » Python gateway to dictionaries
  - PluginManager (work in progress)
    - » Loading dictionaries into the application

New Reflection Model



Outlook

- Implementation of new model
- Extending and creating new dictionaries on demand
- eXtended Type Information (XTI)
  - Stroustrup approach
  - Still very preliminary, solution for far future
- New language gateways
  - Java gateway under investigation

Summary

- Seal
  - Provides standard dictionaries (STL, CLHEP, …)
  - Provides python gateway (PyLCGDict)
  - Feasibility studies for ATLAS and CMS
  - PluginManager will load dictionaries
  - New model will be implemented soon
- Pool
  - Has been using dictionary system from the beginning
- LHCb
  - Will use dictionaries as well