SEAL Reflex, the new Reflection Library

Stefan Roiser

(for the LCG/SEAL Dictionary work package)

LCG Applications Area Meeting, Oct. 6, 2004

Content

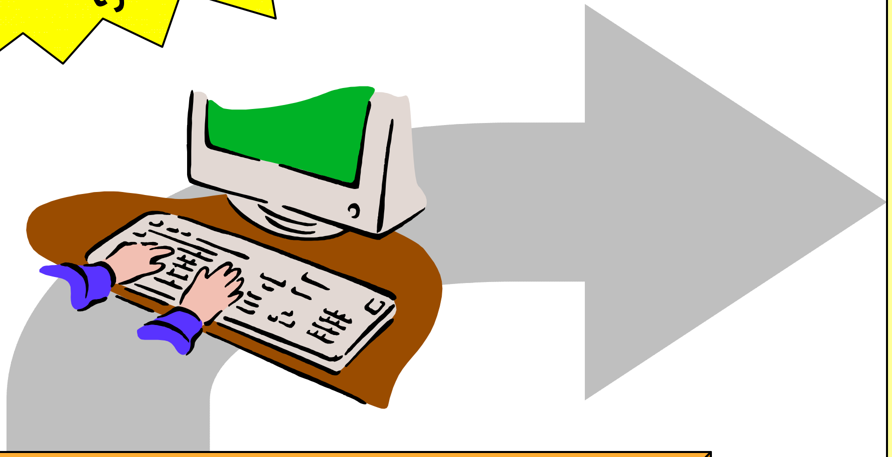
- Reflection and C++
- The Model
- Use Cases
- Producing reflection information
- Current status, future enhancements
- CHEP ‘04
- Conclusion

Definitions

- Reflection is the ability of a language to introspect it’s own structure at runtime and interact with it in a generic way
- A dictionary provides reflection information about types of a certain language to the user

What is Reflection?

Without prior knowledge about types



Dictionary

```
ClassBuilder("Particle").
 addDataMember("m_mass").
 addFunctionMember("mass");
```

```
class Particle {
 public:  double mass();
 private: double m_mass; };
```

```
const Type* t =
 Type::byName("Particle");

Object o = t->construct();

cout << *(double*)
 o.invoke("mass");

t->destruct(o);
```

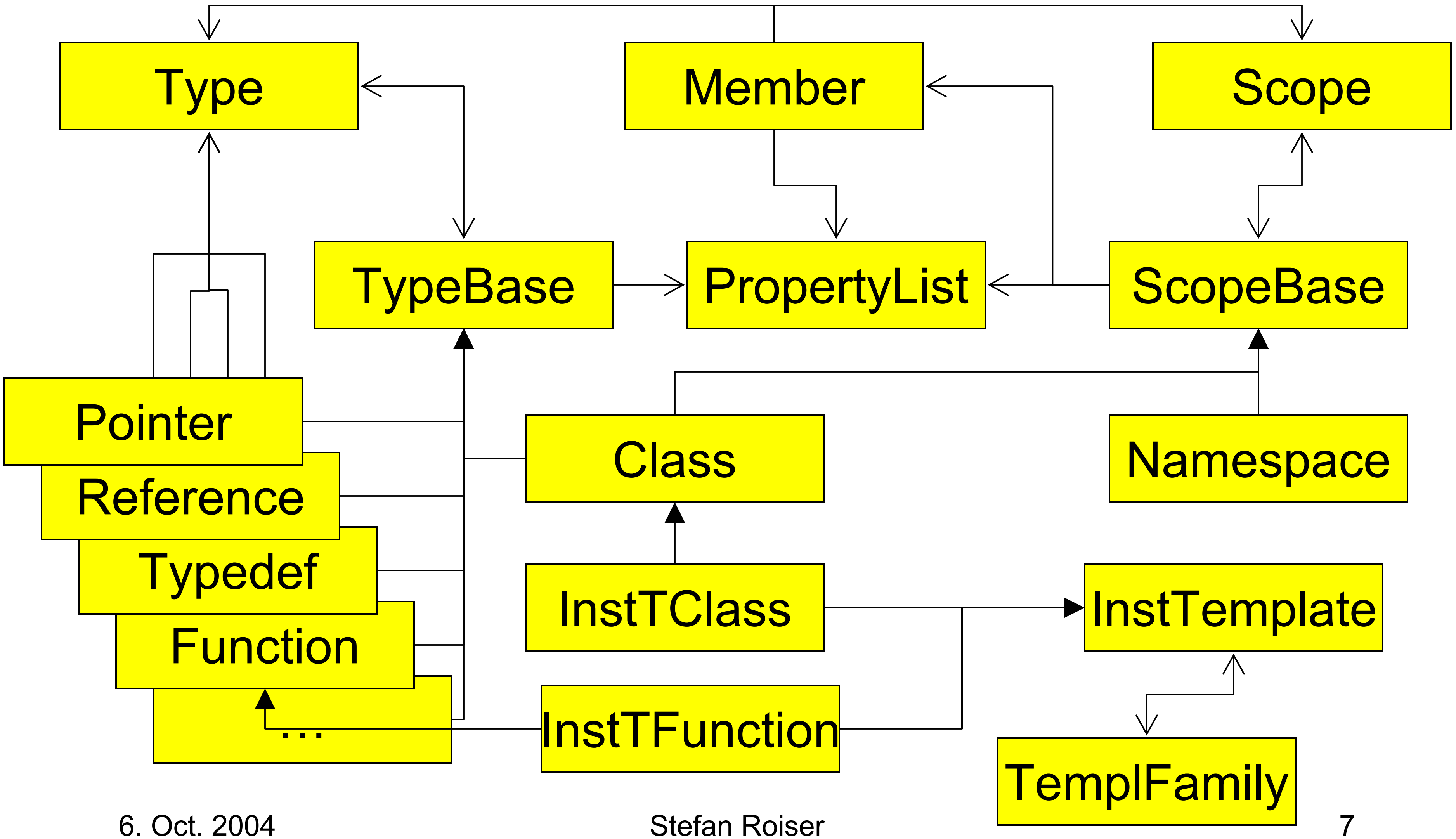
Reflection and C++

- Very limited support by the language (RTTI)
  - Other languages do (e.g. Java, Python,…)
- Several approaches exist, but …
  - Tied to a system or
  - Instrumentation of code needed
- Stroustrup proposed XTI (eXtended Type Information)

Goals

- Enhance C++ with reflection capabilities
  - non intrusive, automated
- Close to the C++ ISO 14882 standard
- Light and standalone system
- Small memory footprint
- Multi platform
  - Linux, Windows, Mac OSX, …

Reflection Model



Reflection Use Cases

- Types
  - Retrieve a specific type
  - Introspect the actual type

```
const Type * t = Type::byName(“Particle”);

size_t s = t->sizeOf();

bool t = t->isClass();
```

Reflection Use Cases (ctd.)

- Classes
  - – Inspect inheritance tree
  - – Create / delete instances

```
const Class * c = t->asClass();

const Base * b = c->base(0);

Object o = c->construct();
c->destruct(o);
```

Reflection Use Cases (ctd.)

- Data Members
  - Get / set values
  - Retrieve offset relative to beginning of class

```
const DataMember * dm = c->dataMember(0);

double d = *(double*) dm->get(v);

size_t s2 = dm->offset();
```

Reflection Use Cases (ctd.)

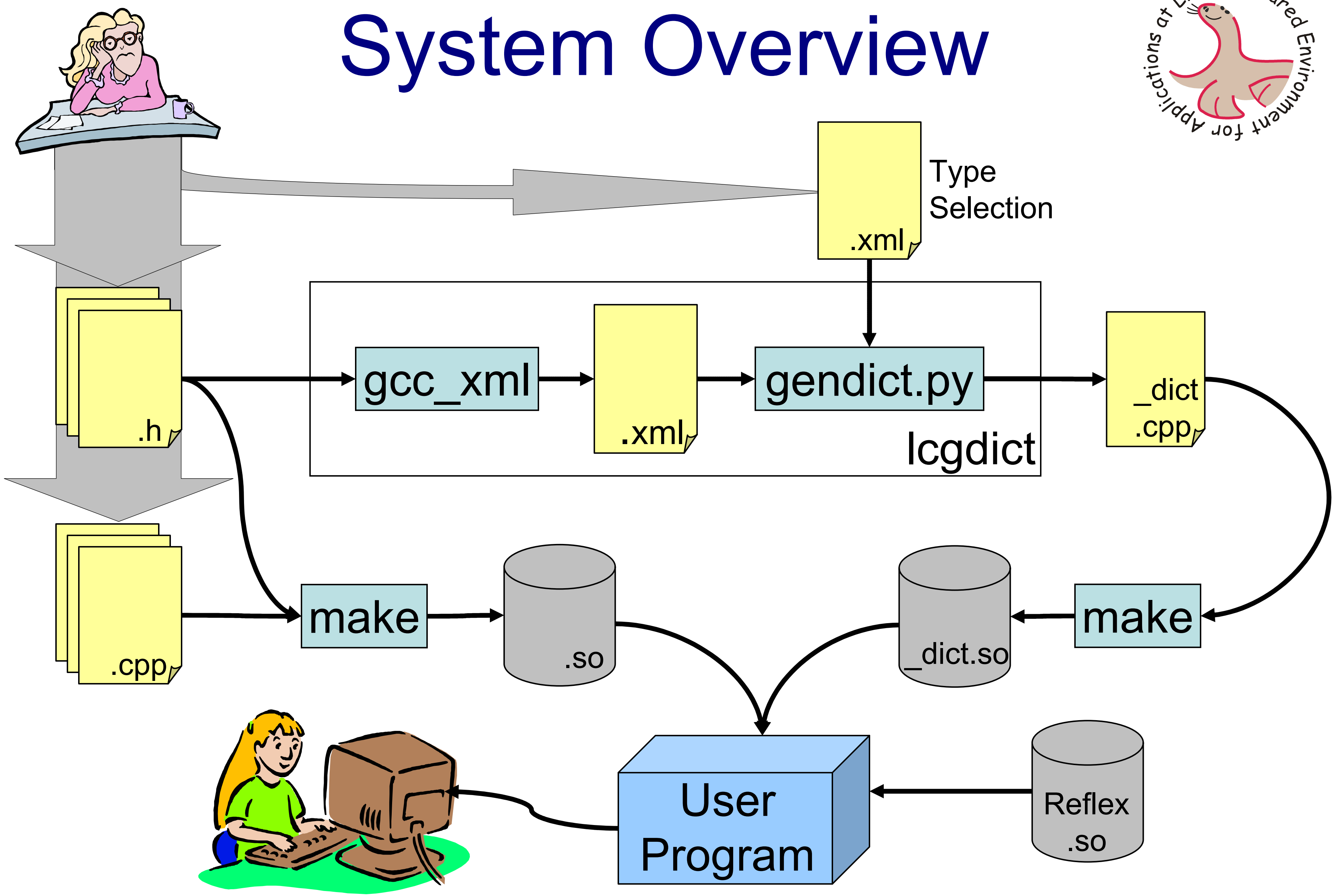
- Function Members
  - Inspect return and parameter types
  - Invoke a function and retrieve return values

```
const FunctionMember * fm = c->functionMember(0);

const Type * rt = fm->type()->asFunction()->returnType();

void * r = fm->invoke(o);
```

Reflection Use Cases (ctd.)

- All Types
  - Reference, Pointer, PointerToMember, Array, Enum, Union, Fundamental, Class, Function
- Typedefs
- Scopes
  - Search scopes
  - Retrieve scopes
- Additional Properties
  - For Types, Scopes and Members
  - Everything that does not fit ISO standard
  - Any type supported
- CV qualifiers
- Templates
  - Inspect families
  - Look for instantiations

System Overview



Dictionary Creation

```
<lcgdict>
  <class name=“Particle”/>
</lcgdict>
```

```
[lxplus]~> lcgdict Particle.h -s selection.xml --reflex -I../incl
Parsing File Particle.h with GCC_XML OK
Generating LCG Dictionary (reflex)
class Particle
[lxplus]~> ls
Particle.h Particle.xml Particle_dict.cpp
[lxplus]~>
```

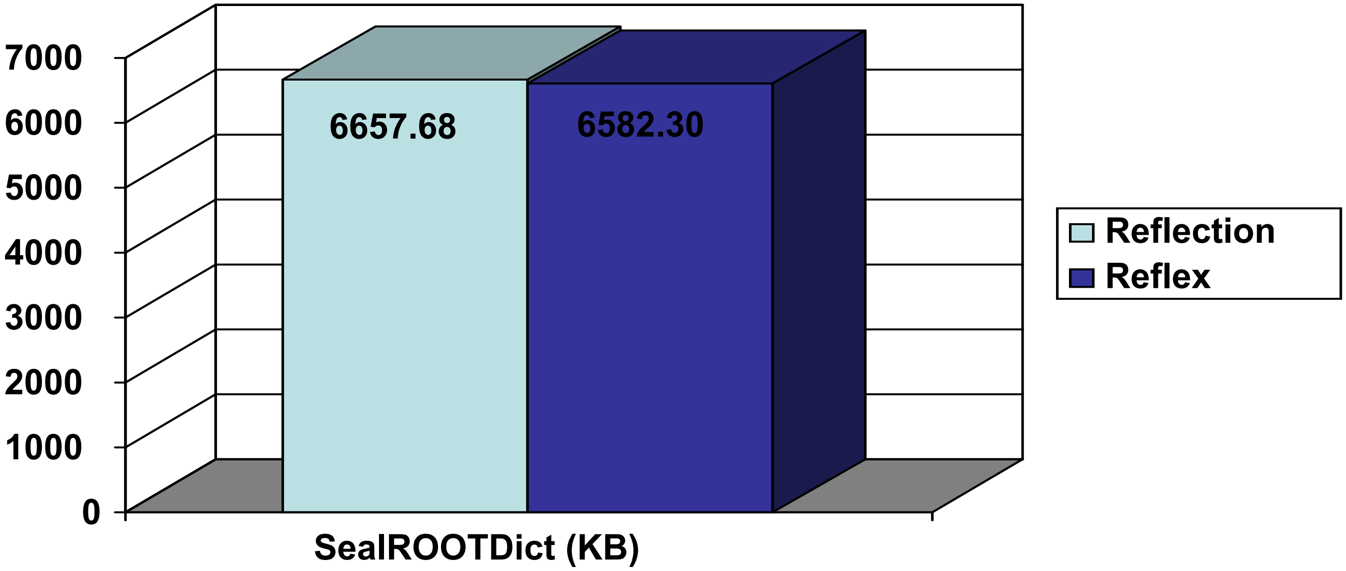
```
#include “Particle.h”
class Particle_dict { public: Particle_dict(); };
void* stubfun(void* o) { return (void*)((Particle*)o)->mass(); }
Particle_dict::Particle_dict() {
ClassBuilder<Particle>(“Particle”)
.addDataMember<double>(”m_mass”,Offset(Particle,m_mass),PRIVATE)
.addFunctionMember<double(void)>(”mass”,&stubfun,0,0,PUBLIC);
}
static Particle_dict instance;
```

gcc_xml

- “[…] generate an XML description of a C++ program from GCC's internal representation.”
- Any gcc compilable program can be used as input
- lcgdict uses the XML file to produce dictionary files

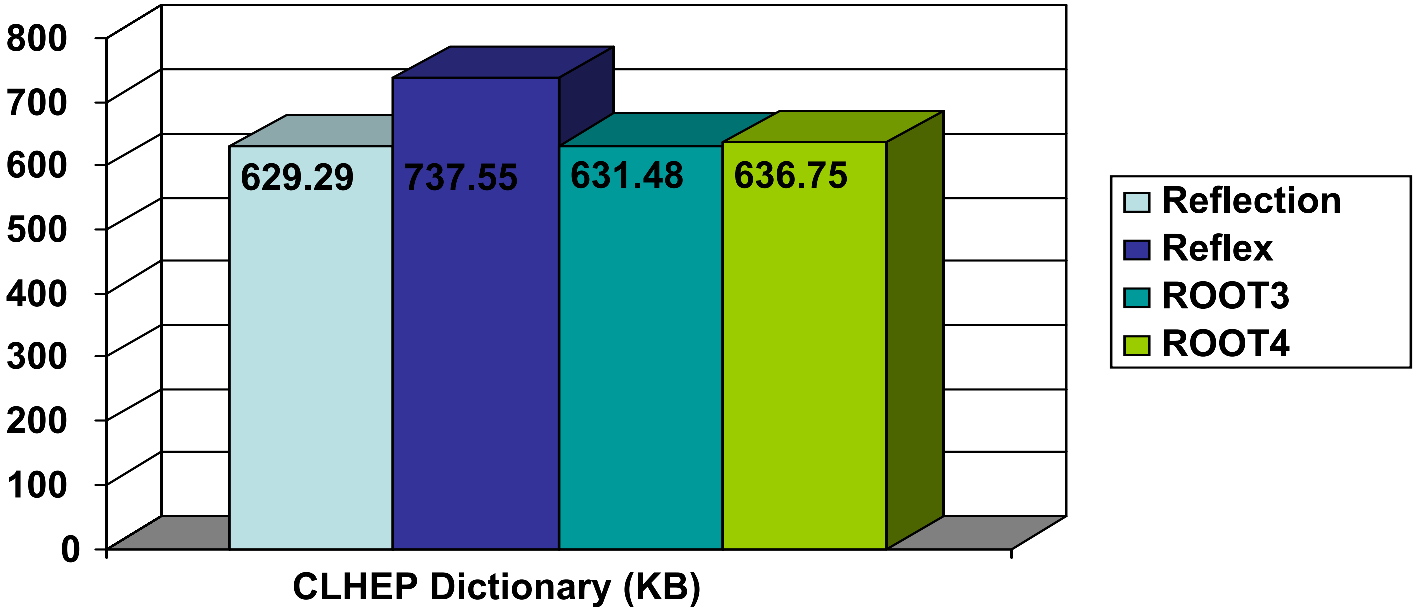
Library Sizes

- SEAL ROOT Dictionary: 405 classes



Library Sizes (ctd.)

CLHEP Dictionary : 56 classes



Status

- Reflex - new version will be released with SEAL 1_4_2 and 1_5_2
  - – In collaboration with the ROOT team
  - – Idea is common LCG & ROOT dictionary
  - – Several enhancements
    - References, Enums, Unions, Typedefs, …
  - – Closer to ISO standard

Future Enhancements

- Template types
  - – Full implementation needed
- Measuring the memory footprint
  - – Instructing new / delete
  - – Counting instances
- Unloading libraries
  - – Proper deletion of types / scopes

Future Enhancements (ctd.)

- Missing Builders
  - UnionBuilder, EnumBuilder
- STL like iterators
  - In addition to current implementation
- String parser
  - Could be used when building types

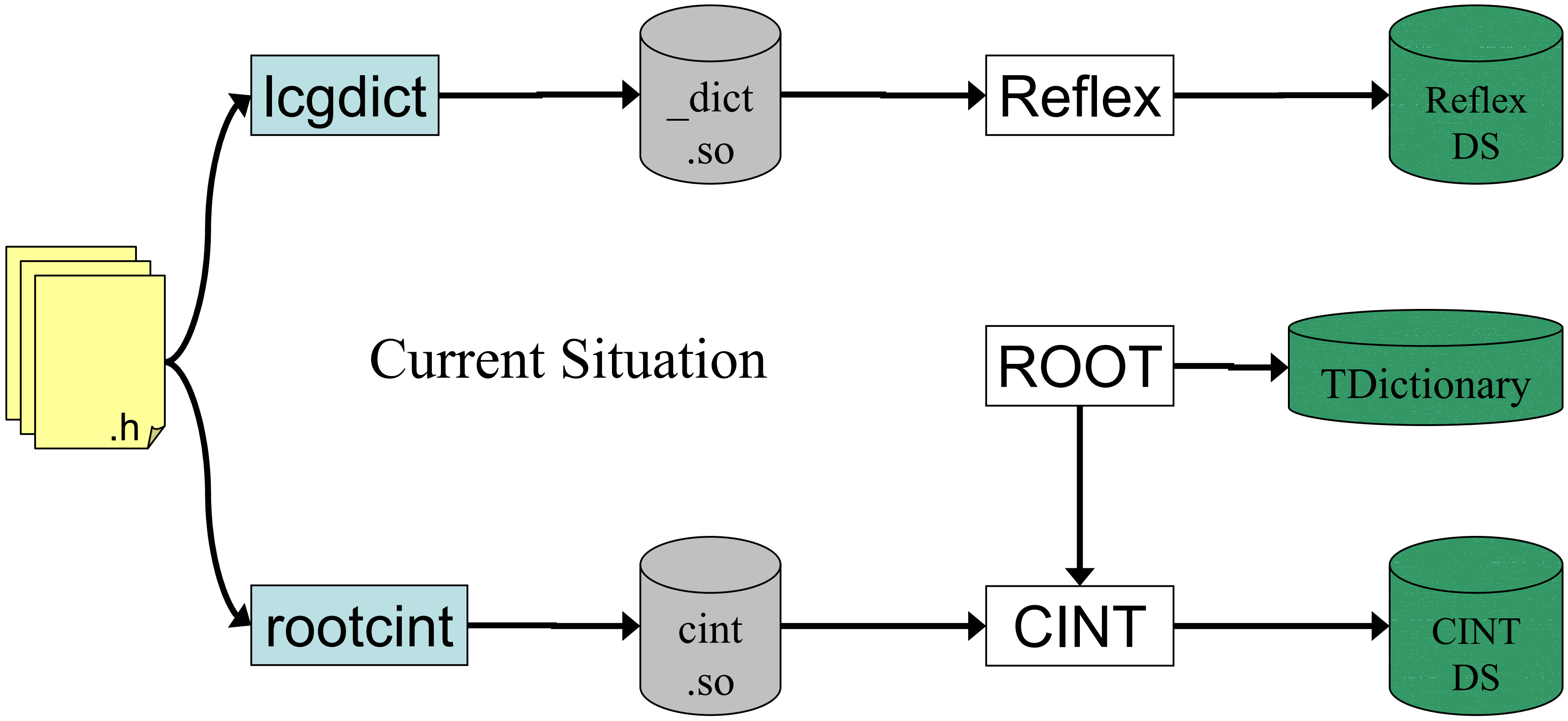
Integration with LCG SW

- POOL
  - Reflex provides sufficient functionality
  - Integration foreseen
- SEAL Scripting
  - PyLCGDict will become PyReflex
  - Work has already started

Integration with ROOT

- For ROOT I/O
  - ROOT Meta Classes (e.g. TClass, TMethod) may be implemented using Reflex
  - Check of sufficient functionality needed
- For interactive ROOT (CINT)
  - Creation of CINT data structures needed
  - Equivalent to existing CINT/Lcgdict gateway in POOL
    - Easy for data members
    - Function members not so straight forward

Integration with ROOT



.h

lcgdict

_dict .so

Reflex

Reflex DS

Current Situation

ROOT

TDictionary

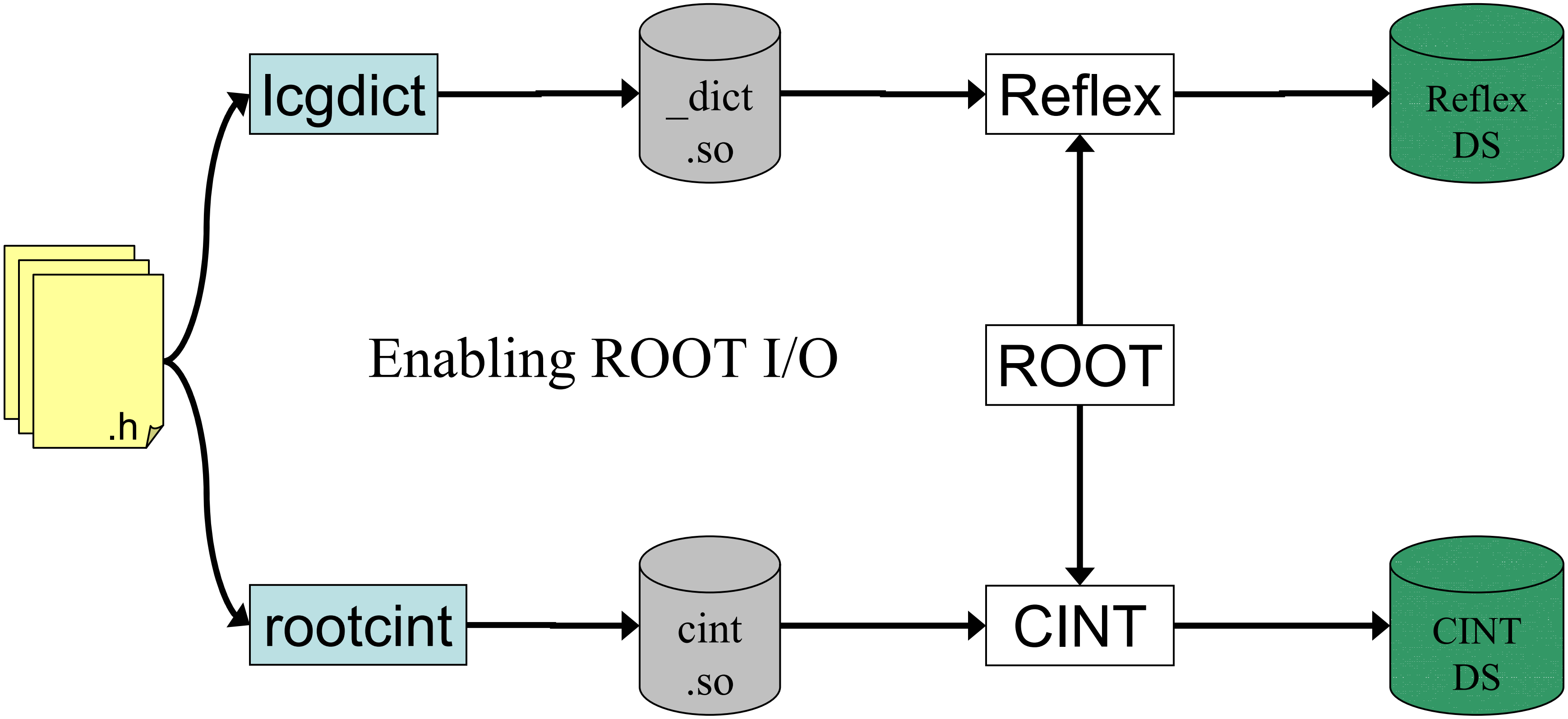
rootcint

cint .so

CINT

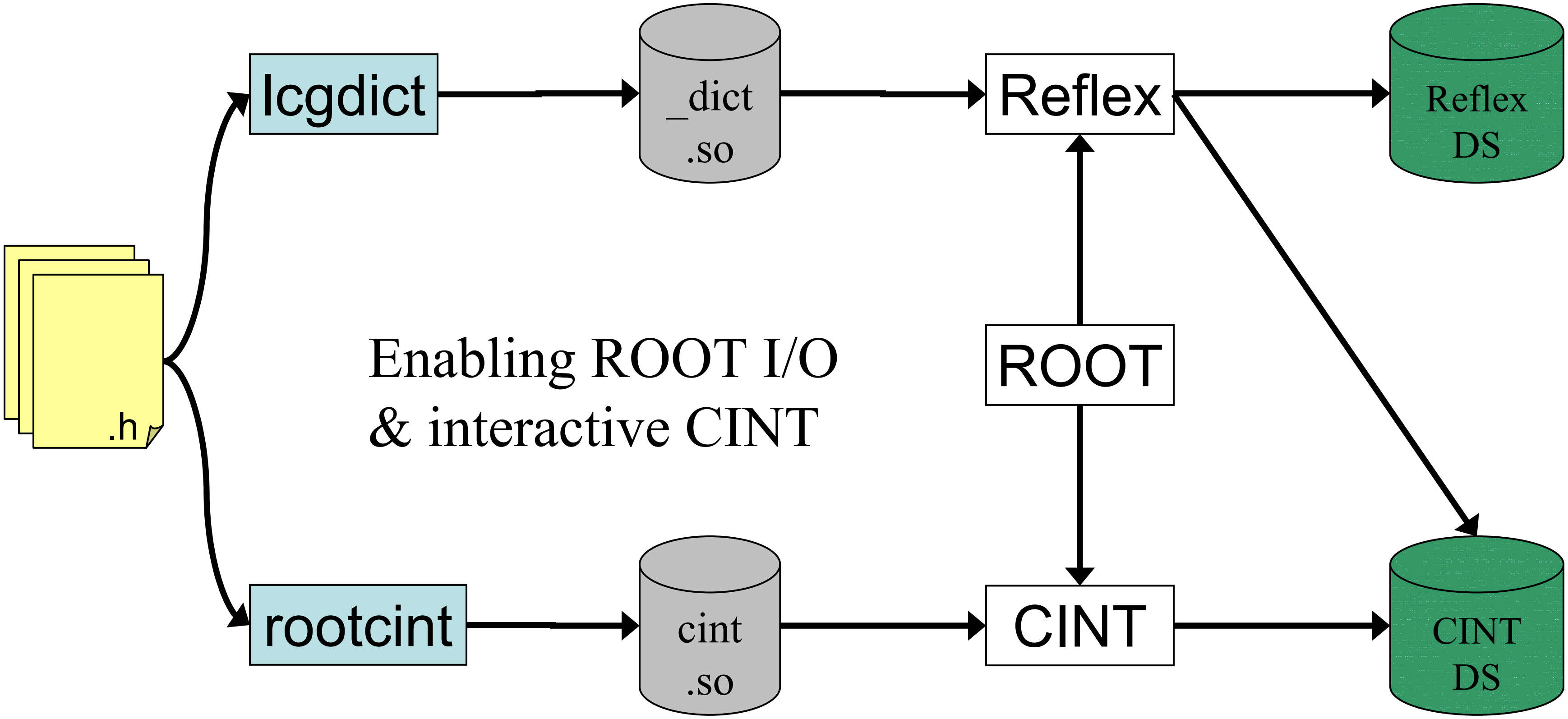
CINT DS

Integration with ROOT



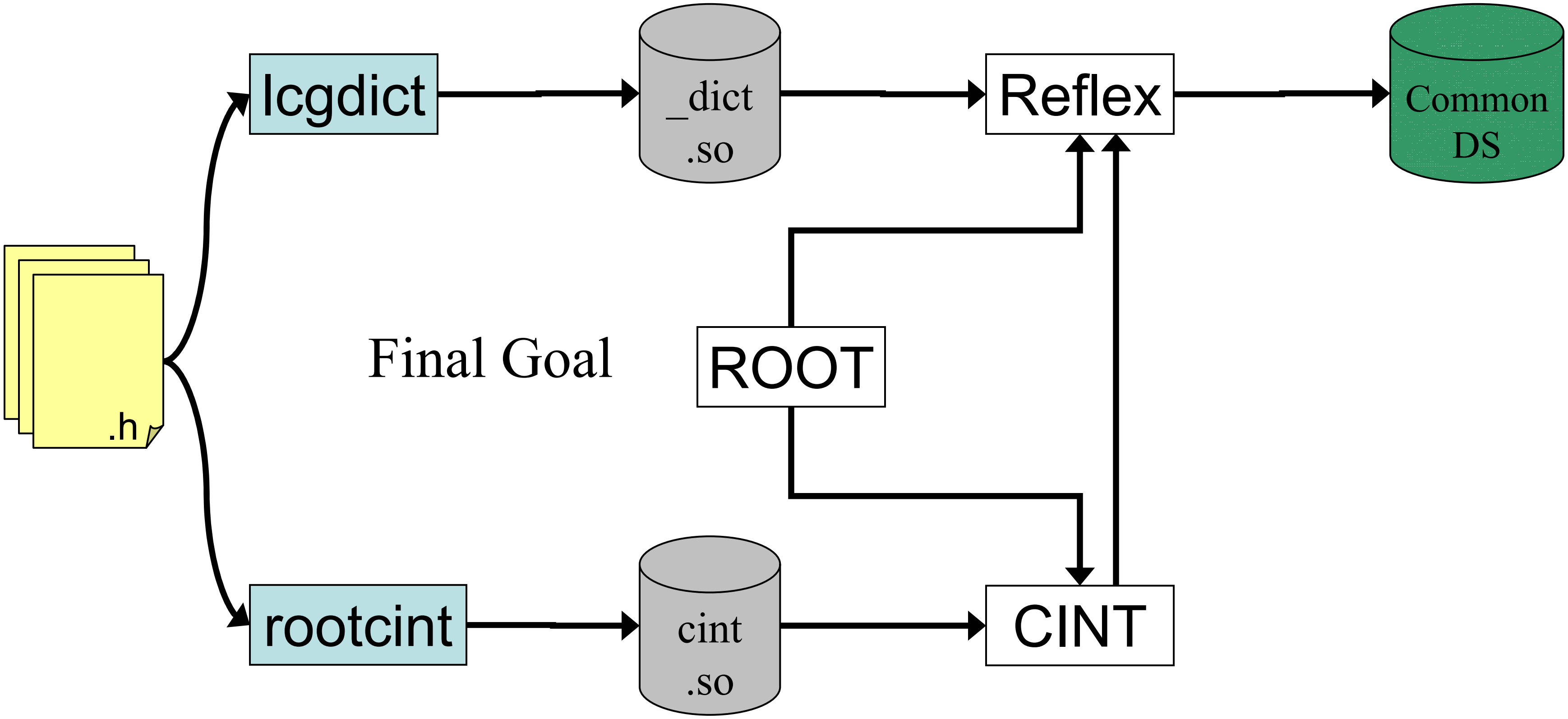
Enabling ROOT I/O

Integration with ROOT



Enabling ROOT I/O
& interactive CINT

Integration with ROOT



CHEP ‘04

- Other projects showed interest
  - FreeHEP Java Library
  - Panoramix
- Improving Standard C++
  - Compile time reflection wanted feature
  - No proposal for the time being
- Summary talk of Core Software
  - Propose reflection as part of ISO standard

SW packages

- Reflection library
  - Current implementation
    - Reflection, ReflectionBuilder
  - New model
    - Reflex
- DictionaryGenerator
  - Produce dictionary files for both versions
- Dictionaries CLHEP, STL, ROOT
  - Standard dictionaries ready to use

Conclusion

- SEAL Dictionaries
  - Light, standalone system
  - Non intrusive, automated code production
  - Dictionaries for any gcc compileable program
- New system - Reflex - ready for integration
  - In collaboration with ROOT
  - Common LCG/ROOT dictionary envisaged
  - Several enhancements, closer to ISO std
  - Positive feedback from CHEP’04

Pointers

- The SEAL project
  - – http://cern.ch/Seal/
- CVS repository
  - – SEAL.cvs.cern.ch:/cvs/SEAL/Dictionary
    - access: kserver, ssh, (anonymous) pserver
- gcc_xml
  - – http://www.gccxml.org