



Data Exchange with the Columbus Schema for C++

Rudolf Ferenc, Árpád Beszédes

University of Szeged, Hungary

Research Group on Artificial Intelligence

{ferenc,beszedes}@cc.u-szeged.hu

March 11-13, 2002



Introduction

- In order to successfully perform a real-life software maintenance or reengineering task, a suitably assembled *set of tools* is required
- To accomplish these tools' interchange, one must have also an *interchange format (schema)* for these tools
- No such standard schema for C++ exists
- This became an active research topic
 - e.g. GXL, DATRIX, GUPRO, Bauhaus



Introduction (cont.)

- We propose a new C++ schema – called ***Columbus*** – as a candidate for such exchange
- It satisfies some important requirements of an exchange format. It reflects the
 - low-level (AST) structure of the code, as well as
 - higher level semantic information (e.g. semantics of types)



Introduction (cont.)

- It is described with standard *UML Class Diagram* notation
- Its implementation is straightforward
- An *API* is very simple to issue (will be available shortly)



Introduction (cont.)

- The Columbus schema is a *good candidate* for exchanging information among tools of various nature (e.g. a C++ parser front end and code-rewriters, metrics tools)
- This is already supported by several usage examples, where the Columbus schema has been successfully used for data exchange
 - e.g. GXL, RSF – rigi, MAISA, Rational Rose, Famix XMI – CodeCrawler



Columbus Schema for C++

- It has been designed in cooperation between the University of Szeged, the Nokia Research Center and FrontEndART Ltd.
- The schema models the “clean” C++ language syntax (preprocessed source code)
- It is used as the internal representation in the C++ reverse engineering tool called also *Columbus*



The structure of the Schema

- Because of the high complexity of the C++ language, we have modularized our schema
- We divided the schema into six packages:
 - base – base classes and data types
 - struc – main program elements
(according to their scoping structure)
 - type – type representation
 - templ – template parameter and argument lists
 - statm – statements
 - expr – expressions



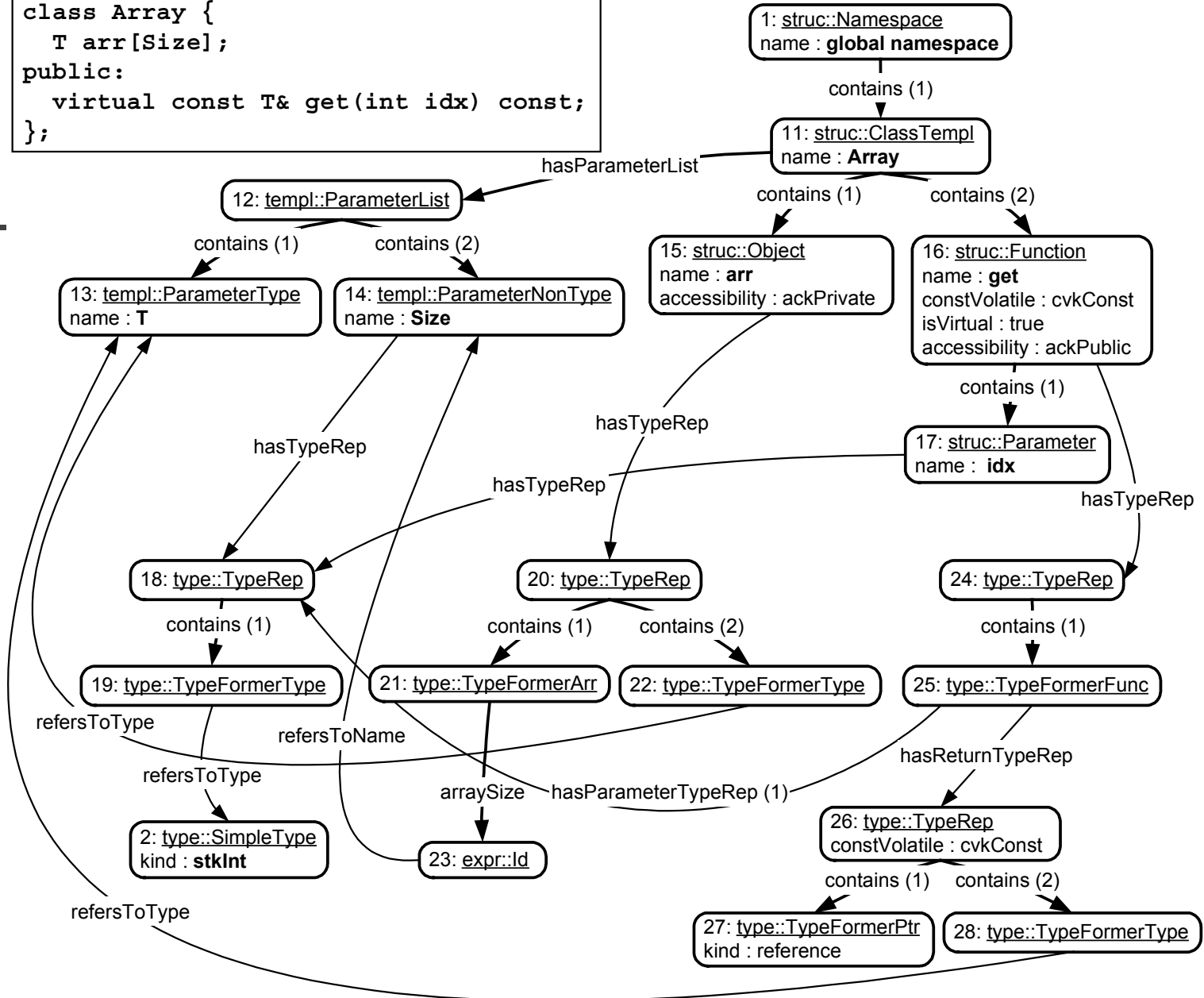
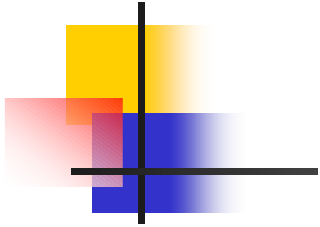
Example schema instance

- Instead of explaining the Class Diagrams that describe the schema,
- we illustrate the use of our schema through an example instance of it

```

template <class T, int Size>
class Array {
  T arr[Size];
public:
  virtual const T& get(int idx) const;
};

```





Implementation experiences

- In parallel to the schema a reverse engineering tool also called *Columbus* has been developed
 - Its C++ extractor module uses the schema as its internal representation – an API is very simple to issue
 - The implementation of the internal representation was straightforward because the schema description is in form of Class Diagrams and the schema is representing every C++ element instance in form of objects and the relations have no attributes (simple graph)
- The final result of the extraction can be presented in various formats that represent graphs, such as GXL and RSF



Summary

- This work was motivated by the observation that successful data exchange is crucial among reverse engineering tools
- This requires a common *format*, which is applicable in various reverse engineering tools, such as front ends and metrics tools
- A standard (or at least reference) schema must be found
- We propose an exchange schema for the C++ language, called the ***Columbus Schema for C++***
- *www.frontendart.com*