An IL Converter and Program Database for Analysis Tools

Kathleen Lindlan, Janice Cuny, Allen D. Malony, Sameer Shende

Department of Computer and Information Science
University of Oregon, Eugene, OR 97403
{klindlan, cuny, malony, sameer}@cs.uoregon.edu

Bernd Mohr

Research Centre Juelich, ZAM
D-52425 Juelich, Germany
b.mohr@fz-juelich.de

Peter Beckman

Advanced Computing Laboratory
Los Alamos National Laboratory
Los Alamos, NM 87545
beckman@acl.lanl.gov

http://www.cs.uoregon.edu/~klindlan/

C++ is a highly complex language that has changed considerably due to recent ISO/ANSI standardization efforts. Currently available tools for analysis and processing of C++ do not adequately support the latest features, such as templates and namespaces. We have developed tools that support these features. These tools are based on the latest Edison Design Group (EDG) front end, a parser that is almost up-to-date with the C++ standard.

The EDG Front End parses a C++ source file, and creates an intermediate language (IL) tree. The IL Converter processes this intermediate language tree, and creates another file. This file essentially contains the C++ interface of the original source. It consists of item descriptions that describe functions and classes, including template instantiations, as well as other types, source files, namespaces, templates, and macros. This file is in "program database" (PDB) format, and can be easily and efficiently read by a programming or a scripting language.

DUCTAPE is a C++ library that processes and merges PDB files. Four DUCTAPE applications have already been developed: "conv" converts PDB files to a more readable format, "merge" merges PDB files, "tree" prints various trees, and "HTMLizer" "htmlizes" C++ source.

Program Database (PDB) Format

all ITEMs	**source position, parent class, access**
HEADER	**<PDB 1.0>**
FILES	**header files included by source file**
ROUTINES	**signature, characteristics, functions called, template from which instantiated**
TYPES	**various characteristics, depending on type**
CLASSES	**characteristics, direct parent classes, friend classes and functions, member functions, information on other members, template from which instantiated**
TEMPLATES	**text of template**
NAMESPACES	**members of namespace or alias**
MACRO	**text of macro**

PDB ROUTINE Item

```
ro#(routineID)  <name of routine>
rloc                    <fileID>  <line>  <column>          # source location
(
rclass                  <classID>                           # parent class
racs                    <pub|prot|priv>                     # access
  OR
rnspace                 <namespaceID>                       # parent namespace
)
rsig                    <typeID>                            # signature
rlink                   <no|internal|C++|C>                 # linkage
rstore                  <ext|stat|auto|NA|asm>              # storage class
rkind                   <ctor|dtor|conv|op>                 # kind
rvirt                   <no|virt|pure>                      # virtuality
rcrvo                   <boolean>                           # is covariant return
                                                            # virtual override?
rinline                 <boolean>                           # is inline?
rcgen                   <boolean>                           # is compiler generated?
rexpl                   <boolean>                           # is explicit ctor?
rtempl                  <templateID or boolean>             # if template instance;
                                                            # boolean when specialized
rspecl                  <boolean>                           # is specialized?
rcall (...)             <routineID>  <line_number>  <column>  # callees
```

PDB CLASS Item

```
cl#(classID) <name of class>
cloc                <fileID>  <line>  <column>                  # source location
(
cclass              <classID>                                   # parent class
cacs                <pub|prot|priv>                             # access
  OR
cnspace             <namespaceID>                               # parent namespace
)
ckind               <class|struct|union>                        # kind
ctempl              <templateID or boolean>                     # if template instance;
                                                                # boolean when specialized
cspecl              <boolean>                                   # is specialized?
cbase (...)         <virt|no> <pub|prot|priv> <direct parent classID>
                                                                #  direct parent classes
cfrclass (...)      <classID>                                   # friend classes
cfrfunc (...)       <routineID>                                 # friend functions
cfunc (...)         <member routineID>                          # member functions
```

```
cmem (...)          <name of non-function member>      # other members
  cmloc             <fileID>  <line>  <column>         # source location
  cmacs             <pub|prot|priv>                   # access
  cmkind            <type|statvar|var>                # kind
  cmtype            <typeID or classID>               # type

  (
  if (cmkind == statvar):
  cmtempl           <templateID or boolean>           # if template static data member
                                                      # boolean when specialized
  cmspecl           <boolean>                         # is specialized?
  cmconst           <boolean>                         # was initializer specified
                                                      # at its declaration
                                                      # within the class definition?

  if (cmkind == var):
  cmisbit           <boolean>                         # is bit field?
  cmmut             <boolean>                         # is mutable?
  )
```

PDB TYPE Item

```
ty#(typeID)<name of type>
yloc                <fileID>  <line>  <column>                # source location
(
yclass              <classID>                                 # parent class
yacs                <pub|prot|priv>                           # access
  OR
ynspace             <namespaceID>                             # parent namespace
)
ykind               <err|void|int|enum|wchar|bool|float
                    |ptr|func|array|tref|ptrmem|tparam|NA>

(
if (ykind == int|enum|wchar|bool):
yikind              <char|schar|uchar|short|ushort|int|uint
                    |long|ulong|longlong|ulonglong>
ysigned             <boolean>                                 # is explicitly signed?
yenum (...)         <name1> <value1>                          # if ykind == enum

if (ykind == float):
yfkind              <float|dbl|longdbl>

if (ykind == ptr):
yptr                <typeID or classID>                       # type pointed to
```

```
if (ykind == func):
yrett                   <typeID or classID>             # return type
yargt (...)             <typeID or classID>             # argument types
yellip                  <boolean>                       # has ellipsis?
yqual                   <const>                         # qualifier
yexcep (...)            <classID>                       # absence of attribute indicates
                                                        # any exception may be thrown;
                                                        # NULL for attribute value indicates
                                                        # no exceptions will be thrown

if (ykind == array):
yelem                   <typeID or classID>             # type of array element
ynelem                  <number of elements>            # -1: variable length array
                                                        # 0: () incomplete-type case
                                                        # else number of elements

if (ykind == tref):
ytref                   <typeID or classID>             # typedef type
yqual                   <const|volatile|restrict>*      # qualifiers

if (ykind == ptrmem):
ympclass                <classID>                       # type of class to which member
                                                        # pointed to belongs
ymptype                 <typeID or classID>             # type of member pointed to
)
```

C++ DUCTAPE:
C++ program Database Utilities
and
Conversion Tools APplication Environment

C++ library	**represents C++ constructs using class hierarchy analyzes and merges Program Database (PDB) files provides 4 applications:**
conv	**prints a program database file in a more readable format**
merge	**merges a set of program database files from separate compilations into one PDB file**
tree	**prints the file inclusion, class hierarchy, and call graph trees**
HTMLizer	**“htmlizes” C++ source**

Example: Source File

```
#include "AppTypes/TSVMeta.h"
#include <iostream>
using namespace std;

template<class T, unsigned D>
class Vektor
{
public:
  enum { ElemDim = 1 };

  Vektor() {
    TSV_MetaAssignScalar<Vektor<T,D>,T,OpAssign>
         ::apply(*this,T(0));
  }

  Element_t& operator[](unsigned int i)
  {
    PAssert (i<D);
    return X[i];
  }

  Message& putMessage(Message& m) const {
    m.setCopy(true);
    ::putMessage(m, X, X + D);
    return m;
  }

private:
  T X[D];
};
```

Example: PDB File

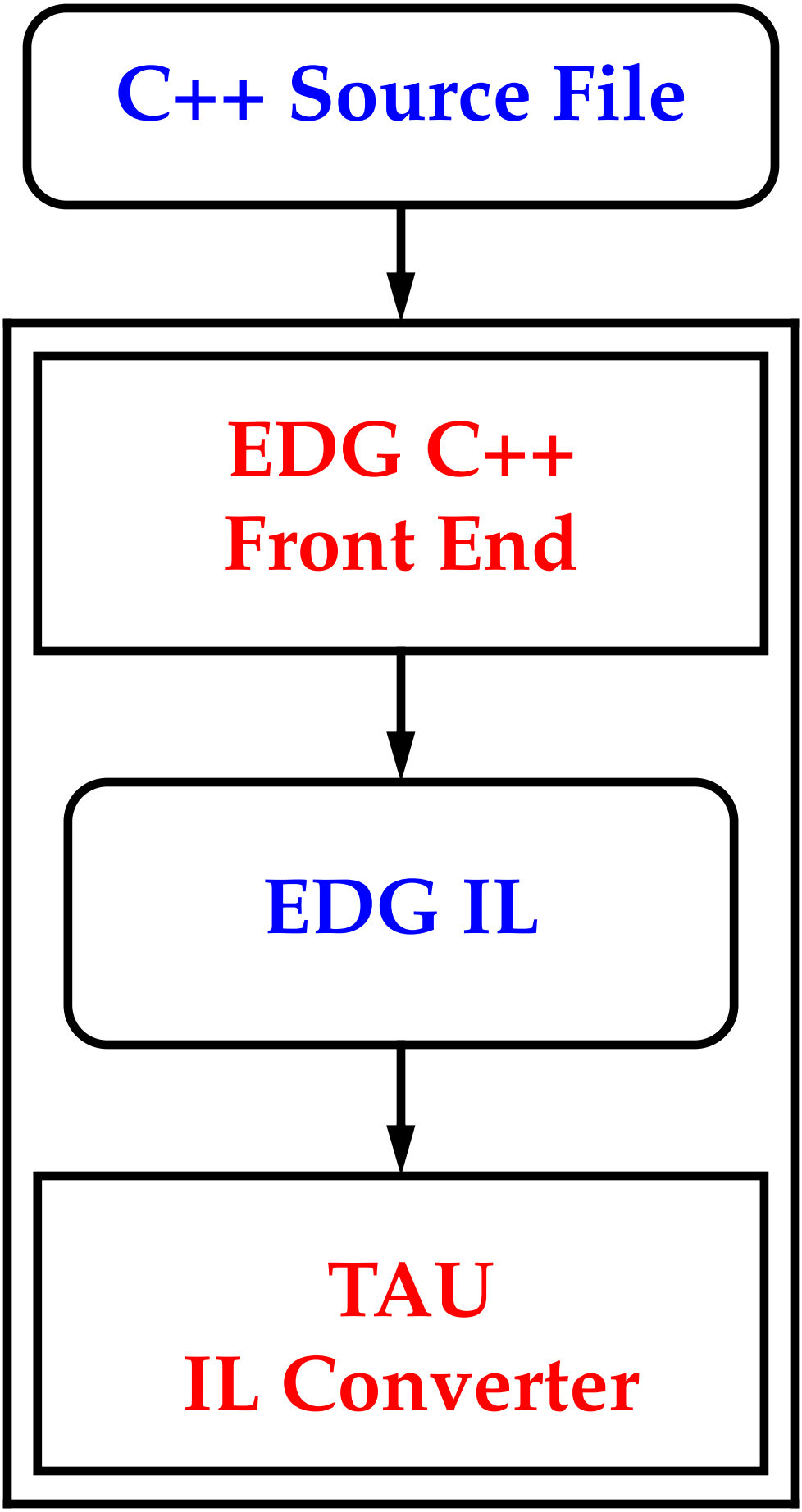
```
<PDB 1.0>

so#98 /parallel/src/AppTypes/TSVMeta.h

so#91 /parallel/src/AppTypes/Vektor.h
sinc so#98

te#1615 Vektor
tloc so#91 42 1
tkind class
ttext template < class T,
         unsigned D >\012
         class Vektor\012{ ... };

ro#2053 Vektor
rloc so#91 52 3
rclass cl#3746
racs pub
rsig ty#3757
rlink C++
rstore ext
rkind ctor
rvirt no
rtempl te#1615

ty#3755 enum Vektor<double, 2U>
          ::<unnamed@1017f020>
yloc so#91 49 8
yclass cl#3746
yacs pub
ykind enum
yikind int
yenum ElemDim 1

cl#3756 Vektor<double, 2U>::DontInitialize
cloc so#91 57 9
cclass cl#3746
cacs pub
ckind class
ctempl te#1615

ty#3757 Vektor<double, 2U> &()
ykind func
yrett ty#3751

ty#3770 Vektor<double, 2U>::Element_t &(
          unsigned int)
ykind func
yrett ty#3753
yargt ty#4
```

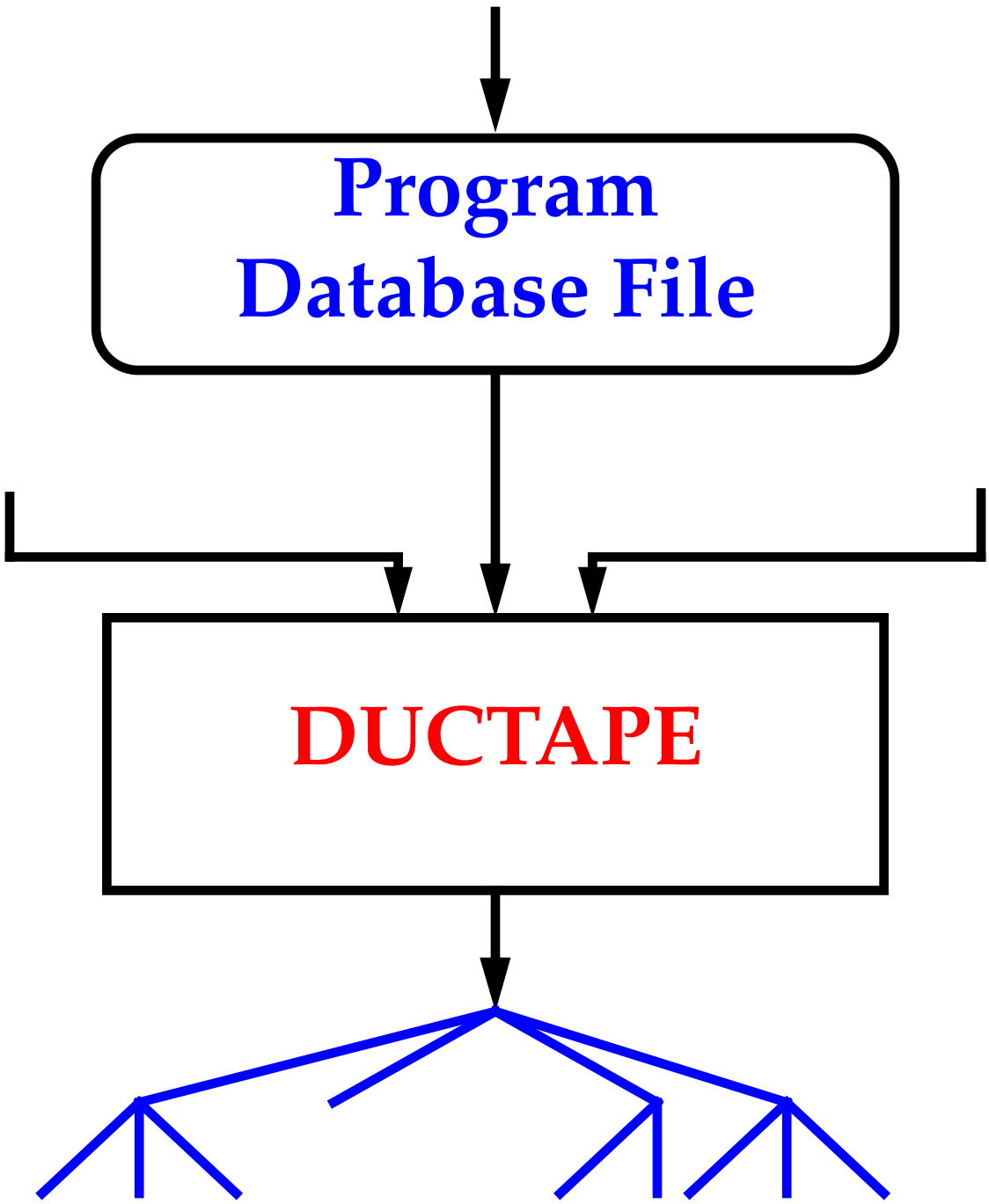
```
ro#2066 operator[]
rloc so#91 163 14
rclass cl#3746
racs pub
rsig ty#3770
rlink C++
rstore ext
rkind op
rvirt no
rtempl te#1615

ro#2072 putMessage
rloc so#91 194 12
rclass cl#3746
racs pub
rsig ty#3776
rlink C++
rstore ext
rvirt no
rtempl te#1615

ty#92 double
ykind float
yfkind dbl

ty#3751 Vektor<double, 2U> &
ykind ptr
yptr cl#3746

ty#3752 double [2]
ykind array
yelem ty#92
ynelem 2

ty#3776 Message &(Message &) const
ykind func
yrett ty#944
yargt ty#944
yqual const

cl#3746 Vektor<double, 2U>
cloc so#91 43 7
ckind class
ctempl te#1615
cfunc ro#2053
cfunc ro#2066
cfunc ro#2072
cmem enum Vektor<double, 2U>
        ::<unnamed@1017f020>
cmloc so#91 49 8
cmacs pub
cmkind type
cmtype ty#3755
cmem Vektor<double, 2U>::DontInitialize
cmloc so#91 57 9
cmacs pub
cmkind type
cmtype cl#3756
cmem X
cmloc so#91 208 5
cmacs priv
cmkind var
cmtype ty#3752

na#1 std
nloc so#40 12 11
```

Example: DUCTAPE Tree Output

```
void vector<DataSourceObject *, alloc>::~vector()
`--> void destroy(vector<DataSourceObject *,
                          alloc>::value_type *,
                          vector<DataSourceObject *,
                          alloc>::value_type *)
    `--> void __destroy(vector<DataSourceObject *,
                          alloc>::value_type *,
                          vector<DataSourceObject *,
                          alloc>::value_type *,
                          vector<DataSourceObject *,
                          alloc>::value_type *)
        `--> void __destroy_aux(vector<DataSourceObject *,
                          alloc>::value_type *,
                          vector<DataSourceObject *,
                          alloc>::value_type *,
                          __false_type)
            `--> void destroy(vector<DataSourceObject *,
                          alloc>::value_type *)
    `--> vector<DataSourceObject *, alloc>::value_type *value_type(
                          const vector<DataSourceObject *,
                          alloc>::value_type *)
```

```
UserList
^--- private FieldLayout<2U>
     `--> (HAS-A) FieldLayout<2U>::ac_id_vnodes
     `--> (HAS-A) FieldLayout<2U>::ac_gc_domain_vnodes
     `--> (HAS-A) NDIndex<2U>
^--- private Mesh<2U>
     ^--- public (IS-A) UniformCartesian<2U, double>
          `--> (HAS-A) Vektor<double, 2U>
          `--> (HAS-A) FieldLayout<2U> *
          `--> (HAS-A) FieldLayout<2U> *
          `--> (HAS-A) Vektor<double, 2U> [4]
          `--> (HAS-A) BareField<Vektor<double, 2U>, 2U> *
          `--> (HAS-A) BareField<Vektor<double, 2U>, 2U> *
`--> (HAS-A) UserList::UserList_t


`--> /parallel/src/Meshes/UniformCartesian.h
     `--> /parallel/src/Meshes/Mesh.h
     `--> /parallel/src/Meshes/Centering.h
     `--> /parallel/src/Meshes/CartesianCentering.h
     `--> /parallel/src/AppTypes/Vektor.h
          `--> /parallel/src/AppTypes/TSVMeta.h
               `--> /parallel/src/AppTypes/TSVMetaAssign.h
               `--> /parallel/src/AppTypes/TSVMetaUnary.h
               `--> /parallel/src/AppTypes/TSVMetaBinary.h
               `--> /parallel/src/AppTypes/TSVMetaDot.h
               `--> /parallel/src/AppTypes/TSVMetaDotDot.h
               `--> /parallel/src/AppTypes/TSVMetaCompare.h
     `--> /parallel/src/Meshes/CartesianStencilSetup.h
```


C++ Source File
EDG C++
Front End
EDG IL
TAU
IL Converter


Program
Database File
DUCTAPE