C++ Program Information
Database for Analysis Tools

Wanghong Yuan, Xiangkui Chen, Tao Xie, Hong Mei, and Fuqing Yang

Department of Computer Sci. & Tech.
Peking University
C++ Program Information Database for Analysis Tools

1. Why employ program information database?

2. How program information database works?

3. Current status and future …
Why program information database?

Program source codes are primary information source of software systems.

Code analyzers analyze source code for different requirements.
Code analyzers: examples

• CSope *for* cross-reference of C programs
• OOTM *for* object-oriented testing
• GRASP *for* reverse engineering of structure diagrams
• COBOL/SRE *for* reengineering to recover reusable components
• ATM *for* analyzing the effect of changes to Ada code
• others, say, CIA and CIA++
Various code analyzers

- need some common information
- share same program information

therefore

- store program information into database
- avoid duplicating extraction process
JBPAS

JBPAS: Jade Bird Program Analysis System

A tool kit of code analysis for C++ programs

Three major components:

• A C++ front end
• An information manager
• A set of program analysis tools
JBPAS Architecture

C++ Front End

Information Manager

Analysis Toolset

PUS  RET  OOTS  CEX  CToC++
JBPAS Analysis Toolset

- Program Understanding System
  facilitate understanding of C++ programs
- Reverse Engineering Tool
  recover object-oriented design documents
- Object-Oriented Test Supporter
  determine test cases & support testing
- Component Extractor
  identify and extract reusable components
- C to C++ Translator
  restructure C program to equivalent C++ program
How program information database works?

• Conceptual Model
define what program information to extract

• database link
incremental parsing
Conceptual Model

Enhanced Entity Relationship (EER) model

C++ programs *viewed as*

• entities (with attributes)

• relationships (with attributes, if any)

Conceptual model *should be*

• comprehensive

• well-defined
JBPAS Conceptual Model: Entities

- Macro
- File
- Class
- Function
- Object (Attribute & Variable)
- Statement
JBPAS Conceptual Model: Relationship

• Relationship between classes
  — class A inherits class B
  — class A refer class B
  — class A is friend of class B

• Relationship between class & Object
  — object O is instance of class C
  — object O is attribute of class C

• Relationship between class & Function
  — function F is member of class C
  — function F is friend of class C
Relationship (continue)

• Relationship between Functions
  — function A call function B
  — function A overload function B

• Relationship between Function & Object
  — object O is local variable of function F
  — object O is refereed by function F

• Relationship between Object & Statement
  — object O is refereed by statement S
  — object O is modified by statement S
Database Link

Incremental parsing
parse only the modified portion

• Compilers
create a .OBJ file for each compiling unit

• JBPAS
create an incremental database for each .CPP file
Difference btw compiler & code analyzer on declaration

• compiler
  
analyze declarations and store their information in the symbol table *temporarily*

• code analyzer
  
extract declarations’ information and store into database *permanently*
Database Link (continue)

Difference btw compiler & code analyzer on declaration

• compiler linker
  
  resolve all external references

• database linker
  
  link all incremental databases to one information database
  keep only one copy of information on shared declaration
  update corresponding reference to the shared declaration
Database Link: example

// COURSE.H
class CCourse{
    char  m_sName[64];
    int   m_nCredit;
    ... ...
};
...

File COURSE.H

// TEACHER.CPP
#include "COURSE.H"
CCourse teaching;
...

File TEACHER.CPP

// STUDENT.CPP
#include "COURSE.H"
CCourse studying;
...

File STUDENT.CPP
### TEACHER.IDB

<table>
<thead>
<tr>
<th>Table</th>
<th>ID</th>
<th>Name</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>20</td>
<td>CCourse</td>
<td>...</td>
</tr>
<tr>
<td>Inst_of</td>
<td>ClassID: 20</td>
<td>ObjectID: 200</td>
<td>...</td>
</tr>
<tr>
<td>Object</td>
<td>ID: 200</td>
<td>Name: teaching</td>
<td>...</td>
</tr>
</tbody>
</table>

### STUDENT.IDB

<table>
<thead>
<tr>
<th>Table</th>
<th>ID</th>
<th>Name</th>
<th>...</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>30</td>
<td>CCourse</td>
<td>...</td>
</tr>
<tr>
<td>Inst_of</td>
<td>ClassID: 30</td>
<td>ObjectID: 300</td>
<td>...</td>
</tr>
<tr>
<td>Object</td>
<td>ID: 300</td>
<td>Name: studying</td>
<td>...</td>
</tr>
</tbody>
</table>
### Program

#### Information

#### Database

<table>
<thead>
<tr>
<th>Table <strong>Class:</strong></th>
<th><strong>ID:</strong> 20</th>
<th><strong>Name:</strong> CCourse</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Table <strong>Inst_of:</strong></th>
<th><strong>ClassID:</strong> 20</th>
<th><strong>ObjectId:</strong> 200</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>ClassID:</strong> 20</th>
<th><strong>ObjectId:</strong> 300</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th>Table <strong>Object:</strong></th>
<th><strong>ID:</strong> 200</th>
<th><strong>Name:</strong> teaching</th>
</tr>
</thead>
</table>

<table>
<thead>
<tr>
<th><strong>ID:</strong> 300</th>
<th><strong>Name:</strong> studying</th>
</tr>
</thead>
</table>
Current status

• the C++ front end

• the information manager

• prototype versions of program understanding system

• product version of reverse engineering tool to be released by Jade Bird Co.
Future . . .

• Form a more concise EER model

• Implement other analysis tools and integrate them in an integrated environment

• Construct similar systems for other object-oriented language
Thank you!