Program Analysis

Extracting static and dynamic information from a software system

Program Analysis

- Extracting information, in order to present abstractions of, or answer questions about, a software system
- Static Analysis: Examines the source code
- Dynamic Analysis: Examines the system as it is executing

Static Analysis

- Involves parsing the source code
- Usually creates an Abstract Syntax Tree
- Borrows heavily from compiler technology but stops before code generation
- Requires a grammar for the programming language
- Can be very difficult to get right

CppETS

- CppETS is a benchmark for C++ extractors
- It consists of a collection of C++ programs that pose various problems commonly found in parsing and reverse engineering
- Static analysis research tools typically get about 60% of the problems right

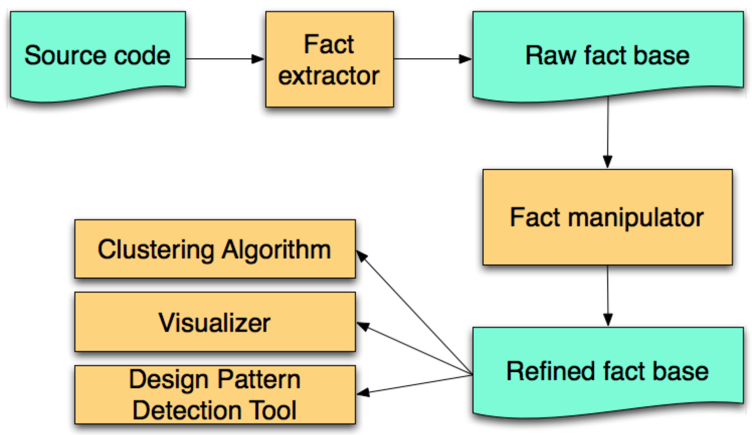
Example program

```
#include <iostream.h>
class Hello {
public: Hello(); ~Hello();
};
Hello::Hello()
{ cout << "Hello, world.\n"; }
Hello::~Hello()
{ cout << "Goodbye, cruel world.\n"; }
main() {
  Hello h;
  return 0;
}
```

Example Q&A

- How many member methods are in the Hello class?
  Two, the constructor `Hello::Hello()` and destructor `Hello::~Hello()`
- Where are these member methods used?
  The constructor is called implicitly when an instance of the class is created. The destructor is called implicitly when the execution leaves the scope of the instance.

Static analysis pipeline



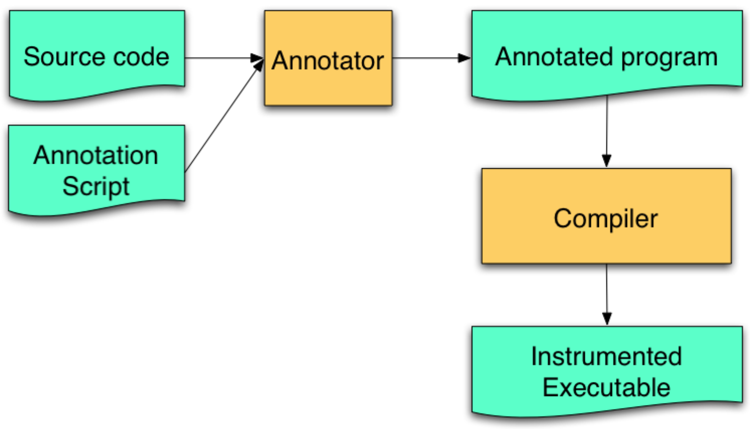
Dynamic Analysis

- Provides information about the run-time behaviour of software systems, e.g.
  - Component interactions
  - Event traces
  - Concurrent behaviour
  - Code coverage
  - Memory management
- Can be done with a profiler or a debugger

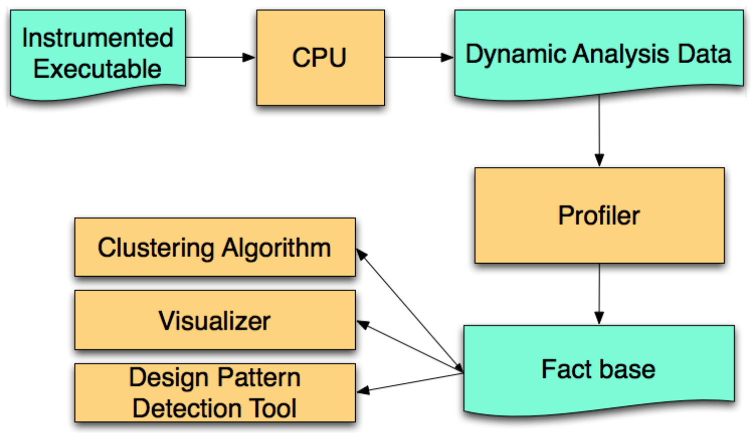
Instrumentation

- Augments the subject program with code that transmits events to a monitoring application, or writes relevant information to an output file
- A profiler can be used to examine the output file and extract relevant facts from it
- Instrumentation affects the execution speed and storage space requirements of the system

Instrumentation process



Dynamic analysis pipeline



Non-instrumented approach

- One can also use debugger log files to obtain dynamic information
- Disadvantage: Limited amount of information provided
- Advantage: Less intrusive approach, more accurate performance measurements

Dynamic analysis issues

- Ensuring good code coverage is a key concern
- A comprehensive test suite is required to ensure that all paths in the code will be exercised
- Results may not generalize to future executions

Static vs. Dynamic

- Reasons over all possible behaviours (general results)
- Conservative
- Challenge: Choose good abstractions

- Observes a small number of behaviours (specific results)
- Precise and fast
- Challenge: Select representative test cases